Analytics and Location Engine 2.0.0.11

aruba

a Hewlett Packard Enterprise company

API Guide

Copyright Information

Related Documents

The following documents are part of the complete documentation set for the Analytics and Location Engine.

- *Analytics and Location Engine 2.0 User Guide*
- *Analytics and Location Engine 2.0.0.x Release Notes*
- *Aruba Instant 6.4.3.x-4.2 User Guide*
- *ArubaOS 6.4 Quick Start Guide*
- *ArubaOS 6.4.2.x or 6.4.3.x User Guide*
- *ArubaOS 6.4.2.x or 6.4.3.x Command-Line Reference Guide*
- *AirWave 8.0.8 User Guide*

Contact Support

Table 1: *Contact Information*

Main Site	arubanetworks.com
Support Site	support.arubanetworks.com
Airheads Social Forums and Knowledge Base	community.arubanetworks.com
North American Telephone	1-800-943-4526 (Toll Free) 1-408-754-1200
International Telephone	arubanetworks.com/support-services/contact-support/
Software Licensing Site	hpe.com/networking/support
End-of-life Information	arubanetworks.com/support-services/end-of-life/
Security Incident Response Team	Site: arubanetworks.com/support-services/security-bulletins/ Email: sirt@arubanetworks.com

Conventions

The following conventions are used throughout this manual to emphasize important concepts:

Table 2: *Typographical Conventions*

Type Style	Description
Italics	This style is used to emphasize important terms and to mark the titles of books.
`System items`	This fixed-width font depicts the following: ● Sample screen output ● System prompts ● Filenames, software devices, and specific commands when mentioned in the text
`Commands`	In the command examples, this bold font depicts text that you must type exactly as shown.
<*Arguments*>	In the command examples, italicized text within angle brackets represents items that you should replace with information appropriate to your specific situation. For example: # **`send`** <*text message*> In this example, you would type “send” at the system prompt exactly as shown, followed by the text of the message you wish to send. Do not type the angle brackets.
`[Optional]`	Command examples enclosed in brackets are optional. Do not type the brackets.
`{Item A | Item B}`	In the command examples, items within curled braces and separated by a vertical bar represent the available choices. Enter only one choice. Do not type the braces or bars.

The following informational icons are used throughout this guide:

NOTE

Indicates helpful suggestions, pertinent information, and important things to remember.

CAUTION

Indicates a risk of damage to your hardware or loss of data.

WARNING

Indicates a risk of personal injury or death.

About This Guide

The Analytics and Location Engine supports two types of APIs: a polling-based REST API, and a publish/subscribe API based on Google Protobuf and ZeroMQ. This guide describes the format of information included in the API, the types of data each API returns, and the steps required to use these APIs to view ALE data.

- The REST-based APIs support HTTP GET operations by providing a specific URL for each query. This information is returned in the JSON format. For more information on ALE Polling APIs, see Polling APIs
- The publish/subscribe API is based on the ØMQ transport. A subscriber uses ØMQ client libraries to connect to ALE and receive information from ALE asynchronously. This information is delivered in the Google Protobuf format. For more information on this group of APIs, see Publish/Subscribe APIs.

When the Analytics and Location Engine integrates with a third-party analytics partner, secure communication between the ALE server and the analytics application may be required. ALE uses a WebSocket Tunnel to secure polling and publish/subscribe APIs and retrieve important context information through a secure channel. For more information, see *Configuring WebSocket Tunnel* in the *Analytics and Location Engine 2.0 User Guide*.

What's New in this Release

The following features are introduced in ALE 2.0.0.11:

Table 3: *API Features Introduced in ALE 2.0.0.11*

Name	Description
Client URL API Enhancements	The following fields have been added to the Client URL API: • **ap_host_name** • **client_mac** • **http_method** • **last_hit_timestamp**
Radio API Enhancements	The **radio_num** field has been added to the Radio API.
Radio Statistics API Enhancements	The **sta_number** field has been added to the Radio Statistics API.
Virtual Access Point Statistics API Enhancements	The **sta_number** field has been added to the Virtual Access Point Statistics API.

There are no new features in ALE 2.0.0.10.

There are no new features in ALE 2.0.0.9.

The following features are introduced in ALE 2.0.0.8:

Table 4: *API Features Introduced in ALE 2.0.0.8*

Name	Description
Radio Statistics API Enhancements	Output values have been added to the **radio_band** field in the Radio Statistics API.
Visibility Record API Enhancements	Starting with ALE 2.0.0.8, the Visibility Record API is available in Branch Office Controller (BoC) deployments. The following fields are available in BoC deployments: • **client_ip** • **dest_ip** • **ip_proto** • **app_id** • **tx_pkts** • **tx_bytes** • **rx_pkts** • **rx_bytes** • **hashed_client_ip** • **ap_mac** • **cc_url_prefix** • **cc_md5** The **ap_mac** field is also available in controller deployments.
Uplink Info API Enhancements	Starting with ALE 2.0.0.8, the Uplink Info API is available in Branch Office Controller (BoC) deployments. The following fields are available in BoC deployments: • **device_mac** • **link_id** • **desc** • **link_status**

Name	Description
	The following fields have been added for BoC deployments: • **vlan_id** • **port_tunnel_desc** • **link_prio** • **wan_status**
Uplink IP Probe Statistics API Enhancements	Starting with ALE 2.0.0.8, the Uplink IP Probe Statistics API is available in Branch Office Controller (BoC) deployments. The following fields are available in BoC deployments: • **device_mac** • **link_id** • **probe_status** • **ip_probe_pkt_loss_pct** • **probe_ip_addr** The following fields have been added for BoC deployments: • **vlan_id** • **avg_rtt** • max_**rtt** • **min_rtt** • **avg_jitter** • **max_jitter** • **min_jitter** • **mos_quality**
Uplink Statistics API Enhancements	Starting with ALE 2.0.0.8, the Uplink Statistics API is available in Branch Office Controller (BoC) deployments. The following fields are available in BoC deployments: • **device_mac** • **link_id** • **rx_pkts** • **tx_pkts** • **rx_bytes** • **tx_bytes** The **vlan_id** field has been added for BoC deployments.
Uplink WAN Compression API	The Uplink WAN Compression message returns information on uplink data compression, which compresses packets to increase data-tranfser efficiency.

The following features are introduced in ALE 2.0.0.7:

Table 5: *API Features Introduced in ALE 2.0.0.7*

Name	Description
Uplink Info API Enhancements	The **tunnel_name** field has been added to the Uplink Info API.
Uplink IP Probe Statistics API Enhancements	The **device_mac** and **tunnel_name** fields have been added to the Uplink IP Probe Statistics API.
Uplink Statistics API Enhancements	The **tunnel_name** field has been added to the Uplink Statistics API.

The following features are introduced in ALE 2.0.0.6:

Table 6: *API Features Introduced in ALE 2.0.0.6*

Name	Description
Modem Statistics API	The Modem Statistics message returns modem statistics data from a Virtual Controller (VC) if a modem is plugged in and the AP uplink type is set to **3G/4G modem**.
Uplink Info API	The Uplink Info message returns general information about uplink ports and tunnels.
Uplink Statistics API	The Uplink Statistics message returns performance information for uplink ports and tunnels.
Uplink IP Probe Statistics API	The Uplink IP Probe Statistics message returns monitoring information for traffic sent through an uplink port or tunnel.
IAP Web Category Summary API	The IAP Web Category Summary message returns IAP performance information based on the type of websites clients visit when they are connected to the network.
IAP Application ID Summary API	The IAP Application ID Summary message returns IAP performance information based on the type of applications being used by clients.
IAP Web Reputation Summary API	The IAP Web Reputation Summary message returns IAP performance information based on the reputation of websites clients visit when they are connected to the network.
IAP Role Statistics API	The IAP Role Statistics message returns IAP performance information based on the user role of a client associated with the IAP.
IAP VLAN Statistics API	The IAP VLAN Statistics message returns IAP performance information based on the VLAN to which the IAP is assigned.
IAP SSID Statistics API	The IAP SSID Statistics message returns IAP performance information based on the SSID of the IAP.
Access Point State API Enhancements	The **current_uplink_inuse** field has been added to the Access Point State API.
Station RSSI API Enhancements	The **classification_type** field has been added to the Station RSSI API.

The following features are introduced in ALE 2.0.0.5:

Table 7: *API Features Introduced in ALE 2.0.0.5*

Name	Description
Access Point State API	The Access Point State message returns (among other information) details on modem status when a modem connects to or disconnects from an IAP.
Topology API Enhancements	AP location fields have been added to the Topology API as part of the Access Point message.

The following features are introduced in ALE 2.0.0.4:

Table 8: *API Features Introduced in ALE 2.0.0.4*

Name	Description
Air Monitor Info API	The Air Monitor Info message returns information on devices that are discovered by Aruba Air Monitor, which analyzes channels to detect potential wireless attacks from neighboring APs and clients.
Spectrum Info API	The Spectrum Info message provides visilibility into network RF interference, which can cause connectivity and performance issues in a deployment.

The following features are introduced in ALE 2.0.0.3:

Table 9: *API Features Introduced in ALE 2.0.0.3*

Name	Description
Client URL API	The Client URL message returns information on URL visibility support.
Rogue Info API	The Rogue Info message returns information on unauthorized rogue devices that can potentially disrupt network operations.
Access Point API Enhancements	The **is_master** and **reboot_reason** fields have been added to the Access Point API for IAP deployments.
Visibility Record API Enhancements	The **session_flags** field has been added to the Visibility Record API for IAP deployments.

The following features are introduced in ALE 2.0.0.2:

Table 10: *API Features Introduced in ALE 2.0.0.2*

Name	Description
Station Statistics API Enhancements	The following fields have been added to the Station Statistics API for IAP deployments: ● **max_tx_rate** ● **tx_data_bytes_transmitted** ● **tx_time_data** ● **rx_time_data** ● **sta_client_health** ● **rx_retries** ● **tx_retries**
Uplink Bandwidth API Enhancements	The following fields have been added to the Uplink Bandwidth API for IAP deployments: ● **ap_mac** ● **ap_name** ● **downstream_jitter** ● **upstream_lost_packets** ● **downstream_lost_packets** ● **hashed_ap_eth_mac** The following fields have been modified from existing fields in the Uplink Bandwidth API: ● **upstream_bytes** ● **upstream_bandwidth**

Name	Description
	● upstream_retries ● downstream_bytes ● downstream_bandwidth ● downstream_retries ● upstream_datagrams ● downstream_datagrams ● upstream_jitter

The following features are introduced in ALE 2.0.0.1:

Table 11: *API Features Introduced in ALE 2.0.0.1*

Name	Description
Uplink Bandwidth API	The Uplink Bandwidth API returns information about the capacity and quality of the connection between users and the Iperf server.

The Representational State Transfer (REST) polling-based API supports HTTPS GET operations by providing a specific URL for each query. The following sections describe each of the Polling APIs supported by ALE. Outputs are displayed in JSON format.

NOTE

Some output parameter fields are optional; certain fields are absent under different modes of operation.

Access Point API

The Access Point (AP) API displays information about APs that terminate on the controllers and IAPs configured to send information to ALE. This API is available in controller and IAP deployments.

AP API queries use the following URL syntax:

`https://1.2.3.4/api/v1/access_point`

The JSON response to this query type displays the following information about the AP:

Table 12: *AP API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	✓
ap_name	Name of the AP. If the AP does not have a name, the API returns the IP address of the AP.	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_group	Name of the AP group	✓	✓
ap_model	Model number of the AP	✓	✓
depl_mode	The AP's deployment mode: ● DEPLOYMENT_MODE_CAMPUS ● DEPLOYMENT_MODE_REMOTE	✓	✓
ap_ip_address	IP address of the AP	✓	✓
reboots	Number of reboots	✓	x
rebootstraps	Number of rebootstraps	✓	x
managed_by	IP address of the controller to which the AP is associated	✓	✓
managed_by_key	Key for the controller IP address	✓	✓
radios	List of radios on the AP	✓	✓
is_master	Indicates if the IAP is an AP master.If **true**, the IAP is an AP master. If **false**, the IAP is not an AP master.	x	✓
ap_location	Location of the AP **NOTE:** Only available in context mode with device location (estimation) as part of the Topology API. AP location is NOT sent when the Access Point REST API is invoked.	✓	x
ap_eth_mac	MAC address of the AP	✓	x
campus_id	ID number identifying a specific campus	✓	x
building_id	ID number identifying a specific campus building	✓	x
floor_id	ID number identifying a specific building floor	✓	x
longitude	Longitude coordinate of the top left corner of the floor (defined in VisualRF)	✓	x
latitude	Latitude coordinate of the top left corner of the floor (defined in VisualRF)	✓	x
ap_x	X coordinate used to determine AP location on a map. This value is based on the AP's distance from the top left corner of the floor.	✓	x
ap_y	Y coordinate used to determine AP location on a map. This value is based on the AP's distance from the top left corner of the floor.	✓	x

```
https://1.2.3.4/api/v1/access_point
```

This query displays output similar to the example below:

```
{
   "Access_point_result": [
      {
         "msg": {
            "ap_eth_mac": {
               "addr": "D8C7C8C0C7BE"
            },
            "ap_name": "1344-1-AL5",
            "ap_group": "1344-hq",
            "ap_model": "135",
            "depl_mode": "DEPLOYMENT_MODE_CAMPUS",
            "ap_ip_address": {
               "af": "ADDR_FAMILY_INET",
               "addr": "10.6.66.67"
            "reboots": 1,
            "rebootstraps": 2,
            "managed_by":{
               af: ADDR_FAMILY_INET
               addr: 0.0.0.0
            }
            "managed_by_key": 2e302bee0164cc154d1d266d8567ada44d49e77af82f4b5ccb
            "radios": {
               "radio_bssid.addr":"D8.C7.C8.46.D8.10"
            },
            "is_master": true
            "ap_location": {
               "ap_eth_mac": "D8C7C8C0C7BE",
               "campus_id": "6F9DEC79839D458B9F148D16A46A353E",
               "building_id": "83393A922FB249C1929B95393A2AAFDA",
               "floor_id": "260BE76B0DD13E7AAF18EB3B47DD7F7B",
               "longitude": -122.008,
               "latitude": 37.4129,
               "ap_x": 22.15,
               "ap_y": 99.18
            }
         },
         "ts": 1382046667
      }
   ]
}
```

Virtual Access Point API

The Virtual Access Point API displays information about virtual APs (VAP) configured for WLAN connections on an AP. This API is available in controller and IAP deployments.

VAP API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/virtual_access_point
```

Table 13: *VAP API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
bssid	BSSID of the virtual AP	✓	✓
ssid	SSID of the virtual AP	✓	✓
radio_bssid	MAC address of a radio on the virtual AP	✓	✓

```
https://1.2.3.4/api/v1/virtual_access_point
```

This query displays output similar to the example below:

```
{
   "Virtual_access_point_result":[
      {
         "msg":{
            "bssid":
            {
               "addr": "94B40FF11080"
            },
            "ssid": "instant",
            "radio_bssid":
            {
               "addr": "94B40FF11080"
            }
         },
         "ts": 1434644130
      }
   ]
}
```

Station API

The Station API displays information about clients associated to the WLAN that sends information to ALE. Every associated and authenticated user on the wireless network is represented by a station object. This API is available in controller and IAP deployments. The JSON response to this query type displays the following types of information about a station:

Station API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/station
```

Table 14: *Station API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client station	✓	✓
username	Corresponding username from the user table on the WLAN controller or IAP	✓	✓
role	Name of the user role currently assigned to the client. This is applicable to only authenticated users	✓	✓
bssid	BSSID that the client is connecting to	✓	✓
device_type	Type of device used by the client ● Windows 7 ● iOS devices	✓	✓
sta_ip_address	IP address of the client	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	x
hashed_sta_ip_address	Anonymized value of the client IP address	✓	x
ht	Type of high-throughput traffic sent by the AP: ● **HTT_NONE**: No high-throughput traffic ● **HTT_20MZ**: High-throughput traffic sent through a 20 MHz channel ● **HTT_40MZ**: High-throughput traffic sent through a 40 MHz channel ● **HTT_VHT_20MZ**: Very high-throughput traffic sent through a 20 MHz channel ● **HTT_VHT_40MZ**: Very high-throughput traffic sent through a 40 MHz channel ● **HTT_VHT_80MZ**: Very high-throughput traffic sent through an 80 MHz channel ● **HTT_VHT_160MZ**: Very high-throughput traffic sent through a 160 MHz channel ● **HTT_VHT_80PLUS80MZ**: Very high-throughput traffic sent through an 80+80 MHz channel ● **HTT_INVALID**: Invalid high-throughput traffic	x	✓
ap_name	Name of the AP	✓	x

```
https://1.2.3.4/api/v1/station
```

This query displays output similar to the example below:

```
{
   "Station_result": [
      {
         "msg": {
            "sta_eth_mac": {
               "addr": "F4F15AA2B8E0"
            },
            "username": "Vjammula",
            "role": "Aruba-Employee",
            "bssid": {
               "addr": "D8C7C888D0D0"
            },
            "device_type": "iPhone",
            "sta_ip_address": {
               "af": "ADDR_FAMILY_INET",
               "addr": "10.11.9.248"
            },
            "hashed_sta_eth_mac": "09097EA8F4DACD5A55F3A9D2F456EFE557D35F09",
            "hashed_sta_ip_address": "436738A08110E88906F8A14CCEF66949A3DBAE01"
            },
            "ht": "HTT_20MZ",
            "ap_name": "ac:a3:1e:c1:0b:f0"
         },
         "ts": 1381977108
      },
   ]
}
```

Presence API

The Presence API enumerates all associated and unassociated devices detected or "sighted" by ALE. There is only one presence entry per device.

NOTE

The **associated** field of the Presence API is not a reliable indication of whether a client is associated, as this is based on a heuristic over the type of frames used to record the RSSI. For an accurate client association status, use the Station API.

In distributed IAP deployments, the Presence API enables presence analytics to detect devices. In controller deployments with many access points in a single location, the Proximity or Location APIs provide a more granular position of the devices.

Presence API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/presence
```

The Presence API can focus on one or more individual presence objects by including the following (optional) input parameter in your polling request. Queries that include multiple parameters must format the query with an ampersand (&) symbol between each parameter.

Table 15: *Presence API Query Parameter*

Query Parameter	Definition
associated	Returns a boolean value of **true** or **false**. If **true**, the MAC address is associated. If **false**, the MAC address is not associated anymore.

The output is filtered to only include information on the records that match your request.

Table 16: *Presence API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
stat_eth_mac	MAC address of the client	✓	✓
associated	Indicates whether the client is associated with an AP on the network. If **true**, the client is associated with an AP. If **false**, the client is no longer associated with an AP.	✓	✓
hash_stat_eth_mac	Anonymized value of the client MAC address	✓	✓
ap_name	Name of the AP	✓	✓
radio_mac	MAC address of the radio that hears the client	✓	✓

```
https://1.2.3.4/api/v1/presence
```

This query displays output similar to the example below:

```
{
  "Presence_result": [
    {
      "msg": {
        "sta_eth_mac": {
          "addr": "00FF00043DFF"
        },
        "associated": true,
        "hashed_sta_eth_mac": "B9081DE2290A1670307F127D07935F788C7614DE"
        "ap_name": "9c:1c:12:c0:19:5a"
        "radio_mac": {
          "addr": "1864720B8460"
        },
      },
      "ts": 1381871586
    }
  ]
}
```

Proximity API

The Proximity API reports which AP hears the station at the highest RSSI, indicating which AP is closest to the station. This API, which is available in controller and IAP deployments, provides a rough location estimation of the client when ALE runs in context mode without maps.

The Proximity API is particularly useful in distributed deployments, such as IAP deployments, in which mapped locations may not be practical for calculating client location or the number of APs may be too low to determine client location.

Proximity API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/proximity
```

Table 17: *Proximity API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
radio_mac	MAC address of the corresponding radio	✓	✓
rssi_val	RSSI value at which the AP hears the station **NOTE:** Attenuation (dBm) = RSSI - 96 - client Tx power. If client power is unknown, a value of 10 is used.	✓	✓
ap_name	Name of the AP	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓

```
https://1.2.3.4/api/v1/proximity
```

This query displays output similar to the example below:

```
{
   "Proximity_result": [
      {
         "msg": {
            "sta_eth_mac": {
               "addr": "FEF50B25A59C"
            },
            "radio_mac": {
               "addr": "1864720B8460"
            },
            "rssi_val": -49,
            "ap_name": "9c:1c:12:c0:19:5a"
            "hashed_sta_eth_mac": "D170F995BA2BADA23B27F5412C997F3864BF6FEA"
         }
      }
   ]
}
```

Campus API

Campuses contain buildings with individual floor maps. This API is available in controller and IAP deployments under context modes with device location (estimation or calibration).

Campus API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/campus
```

Table 18: *Campus API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
campus_id	ID number identifying a specific campus	✓	✓
campus_name	Name of the campus location	✓	✓

```
https://1.2.3.4/api/v1/campus
```

This query displays output similar to the example below:

```
{
   "Campus_result": [
      {
         "msg": {
            "campus_id": "6F9DEC79839D458B9F148D16A46A353E",
            "campus_name": "GAP"
         },
         "ts": 1382046667
      },
   ]
}
```

Building API

The Building API provides information about the buildings within each campus structure. Though each building name must be unique within a campus, other campuses can use the same building name. This API is available in controller and IAP deployments under context modes with device location (estimation or calibration).

Building API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/building
```

Table 19: *Building API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
building_id	ID number identifying a specific campus building	✓	✓
building_name	Name of the campus building	✓	✓
campus_id	ID number identifying a specific campus location	✓	✓

```
https://1.2.3.4/api/v1/building
```

This query displays output similar to the example below:

```
{
   "Building_result": [
      {
         "msg": {
            "building_id": "83393A922FB249C1929B95393A2AAFDA",
            "building_name": "3600-RFBOX",
            "campus_id": "ECDDE4535C8E4723B8AF849B3F86E7BF"
         },
         "ts": 1382046667
      }
   ]
}
```

Floor API

The Floor API retrieves floor definitions for each building in a campus. Campuses contain buildings with individual floor maps. Though each floor name must be unique within a building, other buildings can use the same floor name. This API is available in controller and IAP deployments under context modes with device location (estimation or calibration) and can be used to retrieve the URL of the floor plan image.

Floor API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/floor
```

Table 20: *Floor API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
floor_id	ID number identifying a specific building floor	✓	✓
floor_name	Name of the building floor	✓	✓
floor_latitude	Latitude coordinate of the top left corner of this floor (defined in VisualRF) **NOTE:** Only available in context mode with device location (calibration)	✓	✓
floor_longitude	Longitude coordinate of the top left corner of this floor (defined in VisualRF) **NOTE:** Only available in context mode with device location (calibration)	✓	✓
floor_img_path	URL path to retrieve the background image for this floor	✓	✓
floor_img_width	Floor width, in the configured units	✓	✓
floor_img_length	Floor length, in the configured units	✓	✓
building_id	ID number identifying the building containing this floor	✓	✓
floor_level	Floor level within a building	✓	✓
units	Unit of measurement used for floor specifications (for example, width and length): ● **METERS** ● **FEET**	✓	✓
grid_size	VisualRF grid cell size, in the configured units	✓	✓

```
https://1.2.3.4/api/v1/floor
```

This query displays output similar to the example below:

```
{
   "Floor_result": [
      {
         "msg": {
            "floor_id": "1C48EF7D78DC4B948F1A15D7CD14FDED",
            "floor_name": "Floor 1",
            "floor_latitude": 0,
            "floor_longitude": 0,
            "floor_img_path": "/images/plan/img_1c48ef7d-78dc-4b94-8f1a-15d7cd14fded.jpg",
            "floor_img_width": 246.33,
            "floor_img_length": 249.92,
            "building_id": "DAD3A7092AC04AA4B2F5DAF266EBD81B",
            "floor_level": 2,
            "units": FEET,
            "grid_size": 20
         },
         "ts": 1382046667
```

```
        }
    ]
}
```

Location API

The Location API retrieves the last known location for a specific MAC client. If historical locations are important to your use-case, they can be stored in an external database or filesystem by listening to and saving the publish/subscribe Location API.

The Location API is available in controller and IAP deployments under context modes with device location (estimation or calibration). If AP density is insufficient but location information is desired, "single AP location" can be enabled to ensure that a rough location is still calculated, though at a higher uncertainty.

NOTE

The **associated** field of the Location API is not a reliable indication of whether a client is associated, as this is based on a heuristic over the type of frames used to record the RSSI. For an accurate client association status, use the Station API.

Location API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/location?sta_eth_mac=AA:BB:CC:DD:EE:FF
```

The Location API only supports querying using MAC addresses.

Table 21: *Location API Query Parameter*

Query Parameter	Definition
stat_eth_mac	Returns presence objects in context of a specific MAC address. For example, AA:BB:CC:DD:EE:FF.

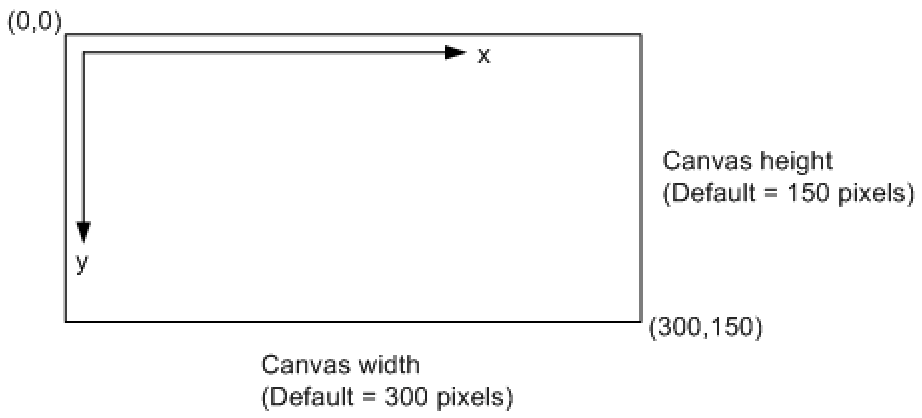
The output is filtered to only include information on the records that match your request.

Table 22: *Location API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
sta_location_x	X coordinate used to determine client location on a map. This value is based on the number of feet, or meters if configured, the station is from the top left corner of the floor	✓	✓
sta_location_y	Y coordinate used to determine client location on a map. This value is based on the number of feet, or meters if configured, the station is from the top left corner of the floor	✓	✓
error_level	Indicates the radius of horizontal uncertainty, computed at 95%. This means the sum of the probability of potential locations contained in this uncertainty circle represents 95% of the whole venue probability. The unit of this radius is configured and published in the unit field.	✓	✓
associated	Indicates whether the client is associated with an AP on the network. If **true**, the client is associated with an AP. If **false**, the client is no longer associated with an AP.	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
campus_id	ID number identifying a specific campus	✓	✓
building_id	ID number identifying a specific campus building	✓	✓
floor_id	ID number identifying a specific building floor	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓
geofence_ids	Indicates whether the client is located in a GeoFence region. If **true**, the client is located in a GeoFence region. If **false**, the client is no longer located in a GeoFence region.	✓	✓
loc_algorithm	Indicates how the (X,Y) coordinates are populated: ● **ALGORITHM_TRIANGULATION**: Triangulation for ALE 1.3.x and earlier ● **ALGORITHM_AP_PLACEMENT**: Single AP location for ALE 1.3.x and earlier ● **ALGORITHM_CALIBRATION**: Calibration mode ● **ALGORITHM_ESTIMATION**: Estimation mode ● **ALGORITHM_LOW_DENSITY**: Indication that location was calculated under low AP density conditions or mostly low RSSI values in the input data set	✓	✓
longitude	Longitude coordinate of the top left corner of the floor (defined in VisualRF)	✓	✓
latitude	Latitude coordinate of the top left corner of the floor (defined in VisualRF)	✓	✓
altitude	Altitude of the floor (defined in VisualRF)	✓	✓
unit	Unit of measurement used for floor specifications (for example, width and length): ● **METERS** ● **FEET**	✓	✓

Figure 1 *Determining the X, Y Coordinates of a Client*



This query displays output similar to the example below:

```
{
   Location_result": [
      {
         "msg":{
            "sta_eth_mac": {
               "addr": "c0:bd:d1:56:81:f3"
            },
            "sta_location_x": 17.033,
            "sta_location_y": 16.5164,
            "error_level": 9,
            "associated": true,
            "campus_id": "08FBBBBF81D937759B5DAC4963DFBC1A",
            "building_id": "24C73B58A1F33C3ABE427485A9977BFF",
            "floor_id": "D635A61B06673775ADFF61D70B55785C",
            "hashed_sta_eth_mac": "A09B5D8F99F9BB8034A8ADBBEC11B24494981096",
            "geofence_ids": true,
            "loc_algorithm": "ALGORITHM_CALIBRATION",
            "longitude": -122.008,
            "latitude": 37.4129,
            "altitude": 5,
            "unit": METERS
         }
         "ts": 1434750262
      }
   ]
}
```

Application API

The Application API maps an application ID, which is specified in a Visibility Record, to an application name string. This API is available in controller and IAP deployments. See Publish/Subscribe APIs for more information on the Visibility Record.

Client Application API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/application
```

The response to this query type displays the following information:

Table 23: *Application API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
app_id	ID number identifying a specific application	✓	✓
app_name	Name of the application	✓	✓
app_family	The application category	x	✓
app_long_name	Name of the application or the domain name	x	✓

```
https://1.2.3.4/api/v1/application
```

This query displays output similar to the example below in a controller deployment:

```
{
   "Application_result": [
      {
        "msg": {
          "app_id": 50331801,
          "app_name": "Smarter Balanced Testing"
        }
      }
    ]
}
```

This query displays output similar to the example below in an IAP deployment:

```
{
   "Application_result": [
      {
        "msg": {
          "app_id": 999,
          "app_name": "foxnews",
          "app_family": "Web",
          "app_long_name": "FoxNews.com"
        }
      }
   ]
}
```

Destination API

The Destination API maps the destination IP address from the Visibility Record to a domain name. This API is only available in controller deployments. See Publish/Subscribe APIs for more information.

Destination API queries use the following URL syntax:

`https://1.2.3.4/api/v1/destination`

Table 24: *Destination API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
dest_ip	IP address of the client receiving traffic through the network	✓	x
dest_name	Name of the client destination	✓	x
dest_alias_name	(Optional) Alternate name for the client destination	✓	x

`https://1.2.3.4/api/v1/destination`

This query displays output similar to the example below:

```
{
   "Destination_result": [
      {
        "msg": {
          "dest_ip": {
             "af": "ADDR_FAMILY_INET",
             "addr": "98.175.77.106"
          },
          "dest_name": "adserving.autotrader.com",
          "dest_alias_name": "autotrader"
        }
      }
   ]
}
```

GeoFence API

GeoFencing allows a network administrator to designate regions and leverage client location information to monitor client traffic through those designated regions. Once ALE downloads the region information from AirWave or the NaoCloud tool, it uses this information to identify the relationship between the devices and the regions. The Geo_Fence API describes the shape and coordinates of the polygon defining the region and is available in both controller and IAP deployments.

Geo_Fence API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/geo_fence
```

Table 25: *GeoFence API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
floor_id	ID number identifying a specific building floor	✓	✓
geofence_id	ID number identifying a specific GeoFence region	✓	✓
geofence_name	Name of the GeoFence region	✓	✓
type	Type of GeoFence region	✓	✓
point_list	The (X,Y) coordinates of the points designating the GeoFence region	✓	✓

```
https://1.2.3.4/api/v1/geo_fence
```

This query displays output similar to the example below:

```
{
   "Geofence_result":[
      {
         "msg":{
            "floor_id": "260BE76B0DD13E7AAF18EB3B47DD7F7B",
            "geofence_id": "51C67C0A356F35C39089664022AB4BED",
            "geofence_name": "b8b87ef6-da7b-4d51-b7dc-d901547189fd",
            "type": planning
            "point_list":[
               {
               "x": 146.52,
               "y": 207.9
               },
               {
               "x": 137.65,
               "y": 207.55
               },
               {
               "x": 138.01,
               "y": 213.22
               },
               {
               "x": 147.94,
               "y": 213.58
               }
            ]
         }
      }
   ]
}
```

System Information API

The System Information API displays general information about your ALE configuration. This API is available in controller and IAP deployments.

System Information API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/info
```

Table 26: *System Information API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
current_mode	The current deployment mode: ● **CONTEXT**: Context mode ● **CONTEXT_AND_ESTIMATED_LOCATION**: Context mode with device location (estimation) ● **CONTEXT_AND_LOCATION_WITH_CALIBRATION**: Context mode with device location (calibration)	✓	✓
license_valid	Returns a boolean value of **true** or **false**. If **true**, a valid license is installed on your system. If **false**, a valid license is not installed.	✓	✓
key_value	String value	✓	✓

```
https://1.2.3.4/api/v1/info
```

This query displays output similar to the example below:

```
{
   "Info_result":[
      {
         "msg":{
            "current_mode": "CONTEXT_AND_LOCATION_WITH_CALIBRATION",
            "license_valid": true,
            "key_value":[
               {
               "key": "geofence_enabled",
               "value": "false"
               }
            ]
         },
         "ts": 1435017107
      }
   ]
```

WebCC Category API

The WebCC Category API retrieves information about the types of websites clients visit when they are connected to the network. This API is available in controller and IAP deployments. The optional `webcc_cat_id` input parameter, as shown in the example below, restricts the response to a specific `webcc_cat_id`. If this parameter is not supplied, the entire table is returned.

WebCC Category API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/webcc_category
```

Table 27: *WebCC Category API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
cat_id	ID number specifying the website category	✓	✓
category	The website category	✓	✓

```
https://1.2.3.4/api/v1/webcc_category?webcc_cat_id=2
```

This query displays output corresponding to the category ID 2 (building):

```
{
   "WebCCCategory_result":[
      {
         "msg":{
            "cat_id": 2,
            "category": "computer/internet-security"
            "ts": 1428011789
         }
      }
   ]
}
```

Topology API

The Topology API describes the hierarchy between networking components in both controller and IAP deployments. For example, a controller deployment runs on the following network hierarchy:

Controllers > APs connected to the controller > radios in the AP > virtual APs connected to the radios.

Topology API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/topology
```

The default API query returns a compressed network topology. For a more detailed network topology, use the following optional query parameters:

Table 28: *Topology API Query Parameters*

Query Parameter	Definition
expand	Returns a boolean value of **true** or **false**. If **true**, the API displays an expanded network topology tree. If **false**, the API displays the compressed network topology tree.
managed_by	Allows you to request for a specific controller IP address.

The Topology API displays the following output parameters, but not all fields are present in both controller and IAP deployments. Refer to and Table 30 and the example ouputs to determine which fields are present in each mode.

Controller

The **controller** message filter returns information about the network hierarchy for a controller deployment.

Table 29: *Topology API Output Parameters - controller*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
controller_ip_address	IP address of the controller	✓	x
access_points	List of APs associated with the controller	✓	x
radios	List of radios on the AP	✓	x
virtual_access_points	List of virtual APs	✓	x

Cluster Info

The **cluster_info** message filter returns information about the network hierarchy for an IAP deployment.

Table 30: *Topology API Output Parameters - cluster_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
cluster_key	Unique key identifying the cluster deployment	x	✓
cluster_ip	IP address of the cluster	x	✓
access_points	List of APs in the cluster	x	✓
radios	List of radios on the AP	x	✓
virtual_access_points	List of virtual APs	x	✓

```
https://1.2.3.4/api/v1/topology
```

The following query displays the topology output for a controller deployment:

```
{
   "Topology_result":[
      {
         "msg":{
            "controller":{
               "controller_ip_address_addr":"10.11.0.10",
               "access_points":[
                  {"ap_eth_mac_addr":"D8.C7.C8.CC.6D.80",
                  "ap_name":"1344-1-AL59 (AM)"
                  "ap_location":{
                     "ap_eth_mac": "D8.C7.C8.CC.6D.80",
                     "floor_id": "260BE76B0DD13E7AAF18EB3B47DD7F7B",
                     "ap_x": 22.15,
                     "ap_y": 99.18
                  "radios":[
                     {"radio_bssid_addr":"D8.C7.C8.46.D8.10"},
                     {"radio_bssid_addr":"D8.C7.C8.46.D8.00"}]},
                  {"ap_eth_mac_addr":"D8.C7.C8.CC.6D.7C",
                "ap_name":"1344-1-AL56 (AM)",
                "radios":[
                   {"radio_bssid_addr":"D8.C7.C8.46.D7.D0"},
                   {"radio_bssid_addr":"D8.C7.C8.46.D7.C0"}]},
                   "virtual_access_points":[
                      {"bssid_addr":"18.64.72.D7.69.21"},
                      {"bssid_addr":"18.64.72.D7.69.20"}]}]},
                   "virtual_access_points":[
                      {"bssid_addr":"18.64.72.D7.5F.01"},
                      {"bssid_addr":"18.64.72.D7.5F.00"}]}]}]}},
         ,"ts":1437496948
      }
   ]
}
```

The following query displays the topology output for an IAP deployment:

```
{
   "Topology_result":[
      {
         "msg":{
            "cluster_info":{
               "cluster_key":"c80a4ace01ea6b8ed5908826d2afc391c1aeb2355e2d7155ba",
               "cluster_ip.addr":"10.5.166.57",
               "access_points":[
                  {"ap_eth_mac.addr":"94.B4.0F.C7.11.08",
                  "ap_name":"94:b4:0f:c7:11:08"
                  "radios":[
                     {"radio_bssid.addr":"94.B4.0F.F1.10.90"},
                     {"radio_bssid.addr":"94.B4.0F.F1.10.80"},
                     "virtual_access_points":[
                        {"bssid.addr":"94.B4.0F.F1.10.80"}]}]},
         "ts":1437763117
      }
   ]
}
```

Controller API

The Controller API provides the list of controllers within a network.

Controller API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/controller
```

Table 31: *Controller API Output Parameter*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
controller_ip_address	IP address of the controller	✓	x

```
https://1.2.3.4/api/v1/controller
```

This query displays output similar to the example below:

```
{
   "Controller_result": [
      {
         "msg": {
            "controller_ip_address": {
               "af": "ADDR_FAMILY_INET",
                "addr": "10.11.0.10"
            }
        },
        "ts": 1437499995
    }
```

Cluster Info API

The Cluster Info API displays information about clusters in an IAP deployment.

Cluster Info API queries use the following URL syntax:

```
https://1.2.3.4/api/v1/cluster_info
```

Table 32: *Cluster Info API Output Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
cluster_key	Unique key identifying the cluster deployment	x	✓
cluster_name	Name of the cluster	x	✓
organization	Name of the organization	x	✓
cluster_ip	IP address of the cluster	x	✓

```
https://1.2.3.4/api/v1/cluster_info
```

This query displays output similar to the example below:

```
{
    "Cluster_result": [
        {
            "msg": {
                "cluster_key": "c80a4ace01ea6b8ed5908826d2afc391c1aeb2355e2d7155ba",
                "cluster_name": "VC-VTHAKKAR",
                "organization": "CORPORATE",
                "cluster_ip": {
                    "af": "ADDR_FAMILY_INET",
                    "addr": "10.5.166.57"
                }
            },
            "ts": 1437678800
        }
    ]
}
```

The ALE publish/subscribe API uses ØMQ client libraries to allow network administrators to connect to ALE and subscribe to selected topics. Once a user has subscribed to a topic, ALE begins publishing messages and sends them to the subscribers.

All messages are encoded using Google Protocol Buffer and specified using a .proto file. Network application developers, who create applications to process this data, use the .proto file and protocol buffer compiler (protoc) to generate a message parsing code in the desired programming language (for example, C++, Java, or Python).

The Northbound API (NBAPI) publishes messages as events. An event message is the only message type NBAPI will publish. The NBAPI embeds other message types depending on the event.

NOTE

The station is removed from the ALE table if the controllers also remove the station from it's user table. The user idle-timeout for removal is 5 minutes by default and is configurable by using "aaa timers idle-timeout x" where x is the number of minutes for the user to be idled out of the system.

The event protobuf schema is as follows:

```
// Event message definition

message nb_event {
   enum event_operation {
      OP_ADD = 0;
      OP_UPDATE = 1;
      OP_DELETE = 2;
   }
   optional uint64 seq = 1;
   optional uint32 timestamp = 2;
   optional event_operation op = 3;
   optional uint64 topic_seq = 4;
   optional bytes source_id = 5;

   // One of the following is populated depending on the topic
   optional location location = 500;
   optional presence presence = 501;
   optional rssi rssi = 502;
   optional station station = 503;
   optional radio radio = 505;
   optional destination destination = 507;
   optional application application = 509;
   optional visibility_rec visibility_rec = 510;
   optional campus campus = 511;
   optional building building = 512;
   optional floor floor = 513;
   optional access_point access_point = 514;
   optional virtual_access_point virtual_access_point = 515;
   optional geofence geofence = 516;
   optional geofence_notify geofence_notify = 517;
   optional stats_radio stats_radio = 518;
   optional stats_vap stats_vap = 519;
   optional stats_station stats_station = 520;
   optional ap_neighbor_list ap_neighbor_list = 521;
   optional utilization_stats_radio utilization_stats_radio = 522;
   optional sta_rssi sta_rssi = 523;
   optional ap_rssi ap_rssi = 524;
   optional proximity proximity = 525;
   optional webcc_category webcc_category = 526;
```

```
    optional webcc_info webcc_info = 527;
    optional security_message security_message = 528;
    optional spectrum_info spectrum_info = 529;
    optional state_station state_station = 530;
    optional controller_info controller = 531;
    optional cluster_info cluster = 532;
    optional uplink_bandwidth uplink_bandwidth = 533;
    optional client_url client_url = 544;
    optional rogue_info rogue_info = 545;
    optional air_monitor_info air_monitor_info = 546;
    optional state_access_point state_access_point = 548;
    optional stats_modem stats_modem = 549;
    optional uplink_info uplink_info = 550;
    optional stats_uplink stats_uplink = 551;
    optional wan_comp_uplink wan_comp_uplink = 552;
    optional stats_ip_probe_uplink stats_ip_probe_uplink = 553;
    optional summary_webcat_iap summary_webcat = 554;
    optional summary_appid_iap  summary_appid = 555;
    optional summary_webrep_iap summary_webrep = 556;
    optional stats_role_iap stats_role = 557;
    optional stats_vlan_iap stats_vlan = 558;
    optional stats_ssid_iap stats_ssid = 559;
}
```

The global field definitions are as follows:

- **seq**: Uniquely assigned global sequence number
- **timestamp**: Time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds when this event occurred
- **op**: Event operation code. Reflects the new object state. Possible values are: OP_ADD, OP_UPDATE, OP_DELETE
- **topic_seq**: Per topic uniquely assigned sequence number
- **source_id**: Random number of bytes used as a unique ID for the source ALE. This number is unique per ALE instance. The unique source_ids are persisted across reboots.

Access Point

The Access Point message is sent when a new access point (AP) is deployed into the network. This message is sent as soon as the AP joins the network. It is also sent when an AP goes down, comes back up, and then joins a controller. The Access Point API is available in both controller and IAP deployments.

NOTE

When ALE is initiated, this information is bootstrapped from the controller and published immediately. In IAP deployments, this is sent periodically.

The output for this message type displays the following information:

Table 33: *Access Point API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	✓
ap_name	Name of the AP	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_group	Name of the AP group	✓	✓
ap_model	AP model type	✓	✓
depl_mode	AP's deployment mode strings: • **0 = DEPLOYMENT_MODE_CAMPUS**: Campus deployment • **1 = DEPLOYMENT_MODE_REMOTE**: Remote deployment	✓	✓
ap_ip_address	IP address of the AP	✓	✓
reboots	Number of reboots	✓	x
rebootstraps	Number of rebootstraps	✓	x
managed_by	IP address of the controller to which the AP is associated	✓	✓
managed_by_key	Key for the controller IP address	✓	✓
radios	List of radios on the AP	✓	✓
is_master	Indicates if the IAP is an AP master. A value of **1** indicates that the IAP is an AP master. A value of **0** indicates that the IAP is not an AP master.	x	✓
reboot_reason	Reason for IAP reboot	x	✓

The Access Point API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "access_point"
Protobuf schema

message access_point {
   optional mac_address ap_eth_mac = 1;
   optional string ap_name = 2;
   optional string ap_group = 3;
   optional string ap_model = 4;
   optional deployment_mode depl_mode = 5;
   optional ip_address ap_ip_address = 6;
   optional uint32 reboots = 7;
   optional uint32 rebootstraps = 8;
   optional ip_address managed_by = 9;
   optional string managed_by_key = 10;
   repeated radio radios = 11;
   optional bool is_master = 7;
   optional string reboot_reason = 8;
}
```

Application

The Application message is sent when a new application is classified or otherwise recognized, mapping an application ID from the Visibility Record message to an application name. This API is available in both controller and IAP deployments. In IAP deployments, this message is only sent once during ALE startup.

The output for this message type displays the following information:

Table 34: *Application API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
app_id	ID number identifying a specific application	✓	✓
app_name	Name of the application	✓	✓
app_family	The application category	x	✓
app_long_name	Name of the application or the domain name	x	✓

The Application API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "application"
Protobuf schema

message application {
   optional uint32 app_id = 1;
   optional string app_name = 2;
   optional string app_family = 3;
   optional string app_long_name = 4;
}
```

Campus

The Campus message announces a campus. The Campus message is sent only once during ALE startup. This API is available in both controller and IAP deployments under context modes with device location (estimation or calibration).

The output for this message type displays the following information:

Table 35: *Campus API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
campus_id	ID number identifying a specific campus	✓	✓
campus_name	Name of the campus	✓	✓

The Campus API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "campus"
Protobuf schema
```

```
message campus {
   optional bytes campus_id = 1; // 16 bytes id
   optional string campus_name = 2;
}
```

Building

The Building message announces a campus building. Each campus can have multiple buildings, but a building can be located on only one campus. The Building message is sent only once during ALE startup. This API is available in both controller and IAP deployments and only sends messages in context modes with device location (estimation or calibration).

The output for this message type displays the following information:

Table 36: *Building API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
building_id	ID number identifying a specific campus building	✓	✓
building_name	Name of the campus building	✓	✓
campus_id	ID number identifying a specific campus	✓	✓

The Building API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "building"
Protobuf schema

message building {
   optional bytes building_id = 1;
   optional string building_name = 2;
   optional bytes campus_id = 3;
}
```

Floor

The Floor message announces a building floor. Each building can have multiple floors, but a floor can be located on only one building. The Floor message is sent only once during ALE startup. This API is available in both controller and IAP deployments under context modes with device location (estimation or calibration).

The output for this message type displays the following information:

Table 37: *Floor API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
floor_id	ID number identifying a specific building floor	✓	✓
floor_name	Name of the floor	✓	✓
floor_latitude	Latitude coordinate of the top left corner of this floor (defined in Visual RF)	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
	NOTE: Only available in context mode with device location (calibration)		
floor_longitude	Longitude coordinate of the top left corner of this floor (defined in Visual RF) **NOTE:** Only available in context mode with device location (calibration)	✓	✓
floor_img_path	URL path to retrieve the background image for this floor	✓	✓
floor_img_width	Floor width, in the configured units	✓	✓
floor_img_length	Floor length, in the configured units	✓	✓
building_id	ID number identifying a specific campus buliding	✓	✓
floor_level	Floor level within a building	✓	✓
units	Unit of measurement used for floor specifications (for example, width and length): ● **METERS** ● **FEET**	✓	✓
grid_size	VisualRF grid cell size, in the configured units	✓	✓

The Floor API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "floor"
Protobuf schema

message floor {
   optional bytes floor_id = 1;
   optional string floor_name = 2;
   optional float floor_latitude = 3;
   optional float floor_longitude = 4;
   optional string floor_img_path = 5;
   optional float floor_img_width = 6;
   optional float floor_img_length = 7;
   optional bytes building_id = 8;
   optional float floor_level = 9;
   optional string units = 10;
   optional float grid_size = 11;
}
```

Destination

The Destination message is sent when a new destination is classified. This API can be used to map a destination IP address from the Visibility Records to a destination domain name. This API is only available in controller deployments.

The output for this message type displays the following information:

Table 38: *Destination API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
dest_ip	IP address of the client receiving traffic through the network	✓	x
dest_name	Name of the client destination	✓	x
dest_alias_name	Alternate name for the client destination	✓	x

The Destination API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "destination"
Protobuf schema

message destination {
   optional ip_address dest_ip = 1;
   optional string dest_name = 2;
   optional string dest_alias_name = 3;
}
```

Location

The Location message sends location updates for a specific station. This message is sent as soon as ALE calculates the location of an associated or unassociated client with a specified MAC address. The X and Y location values indicate the number of feet, or meters if configured, the station is from the top left corner of the floor map. This API is available in controller and IAP deployments under context modes with device location (estimation or calibration).

NOTE

The **associated** field of the Location API is not a reliable indication of whether a client is associated, as this is based on a heuristic over the type of frames used to record the RSSI. For an accurate client association status, use the Station API.

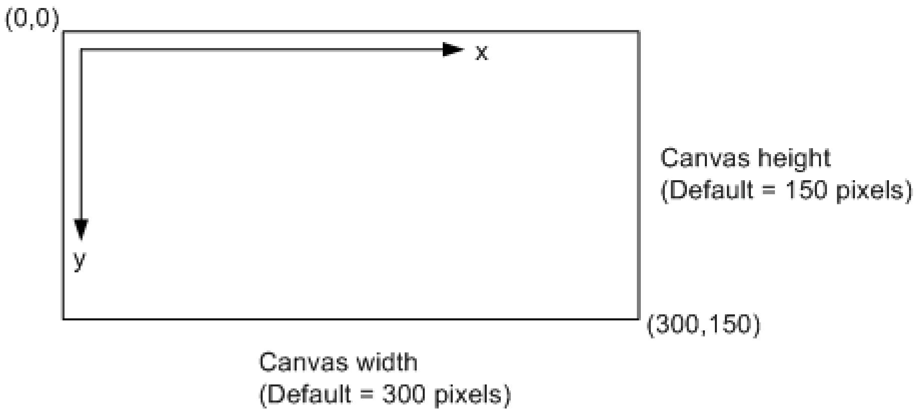
The output for this message type displays the following information:

Table 39: *Location API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
sta_location_x	X coordinate used to determine client location on a map. This value is based on the number of feet, or meters if configured, the station is from the top left corner of the floor	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_location_y	Y coordinate used to determine client location on a map. This value is based on the number of feet, or meters if configured, the station is from the top left corner of the floor	✓	✓
error_level	Indicates the radius of horizontal uncertainty, computed at 95%. This means the sum of the probability of potential locations contained in this uncertainty circle represents 95% of the whole venue probability. The unit of this radius is configured and published in the unit field.	✓	✓
associated	Indicates whether the client is associated with an AP on the network. A value of **1** indicates that the client is associated with an AP. A value of **0** indicates that the client is no longer associated with an AP.	✓	✓
campus_id	ID number identifying a specific campus	✓	✓
building_id	ID number identifying a specific campus building	✓	✓
floor_id	ID number identifying a specific building floor	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓
geofence_ids	Indicates whether the client is located in a GeoFence region. A value of **1** indicates that the client is located in a GeoFence region. A value of **0** indicates that the client is not located in a GeoFence region.	✓	✓
loc_algorithm	Algorithm indicating how the (X,Y) coordinates are populated: ● **0 = ALGORITHM_TRIANGULATION**: Triangulation for ALE 1.3.x and earlier ● **1 = ALGORITHM_AP_PLACEMENT**: Single AP location for ALE 1.3.x and earlier ● **2 = ALGORITHM_CALIBRATION**: Calibration mode ● **3 = ALGORITHM_ESTIMATION**: Estimation mode ● **4 = ALGORITHM_LOW_DENSITY**: Indication that location was calculated under low AP density conditions or mostly low RSSI values in the input data set	✓	✓
rssi_val	RSSI value at which the AP hears the station **NOTE:** Attenuation (dBm) = RSSI - 96 - client Tx power. If client power is unknown, a value of 10 is used.	✓	✓
longitude	Longitude coordinate of the client (defined in Visual RF)	✓	✓
latitude	Latitude coordinate of the client (defined in Visual RF)	✓	✓
altitude	Altitude of the client (defined in VisualRF)	✓	✓
unit	Unit of measurement used for floor specifications (for example, width and length): ● **METERS** ● **FEET**	✓	✓

Figure 2 *Determining the X,Y Coordinates of a Client*



The Location API output schema is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “location”
Protobuf schema

message location {
   optional mac_address sta_eth_mac = 1;
   optional float sta_location_x = 2;
   optional float sta_location_y = 3;
   optional uint32 error_level = 7;
   optional bool associated = 8;
   optional bytes campus_id = 9;
   optional bytes building_id = 10;
   optional bytes floor_id = 11;
   optional bytes hashed_sta_eth_mac = 12;
   repeated bytes geofence_ids = 13;
   optional algorithm loc_algorithm = 14;
   optional uint32 rssi_val = 15;
   optional double longitude = 16;
   optional double latitude = 17;
   optional double altitude = 18;
   optional measurement_unit unit = 19;
}
```

Presence

The Presence message is sent as soon as an AP radio hears a client MAC address. If the client associates to the network, then the associated field is set to **1**; if the client is not associated, this is set to **0**. Removal of a client (when silent for a long time interval) and changes in the associated state are indicated by the OP_DELETE and OP_UPDATE fields. This API is available in both controller and IAP deployments.

NOTE

The **associated** field of the Presence API is not a reliable indication of whether a client is associated, as this is based on a heuristic over the type of frames used to record the RSSI. For an accurate client association status, use the Station API.

The output for this message type displays the following information:

Table 40: *Presence API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
associated	Indicates whether the client is associated with an AP on the network. A value of **1** indicates that the client is associated with an AP. A value of **0** indicates that the client is no longer associated with an AP.	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓
ap_name	Name of the AP	✓	✓
radio_mac	MAC address of the radio that hears the client	✓	✓

The Presence API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "presence"
Protobuf schema

message presence {
   optional mac_address sta_eth_mac = 1;
   optional bool associated = 2;
   optional bytes hashed_sta_eth_mac = 3;
   optional string ap_name = 4;
   optional mac_address radio_mac = 5;
}
```

Proximity

The Proximity message indicates which AP is closest to the station, providing a rough location estimation of the client when ALE runs in context mode without maps. This API is available in both controller and IAP deployments.

The output for this message type displays the following information:

Table 41: *Proximity API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
radio_mac	MAC address of the corresponding radio	✓	✓
rssi_val	RSSI value at which the AP hears the station **NOTE:** Attenuation (dBm) = RSSI - 96 - client Tx power. If client power is unknown, a value of 10 is used.	✓	✓
ap_name	Name of the AP	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓

The Proximity API output schema is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “proximity”
Protobuf schema

message proximity {
   optional mac_address sta_eth_mac = 1;
   optional mac_address radio_mac = 2;
   optional uint32 rssi_val = 3;
   optional string ap_name = 4;
   optional bytes hashed_sta_eth_mac = 5;
}
```

Radio

The Radio message sends information about each radio on a newly added AP. This API is available in both controller and IAP deployments.

The output for this message type displays the following information:

Table 42: *Radio API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	✓
radio_bssid	BSSID of the radio	✓	✓
mode	Specifies the radio mode on an AP; each mode provides a different function: ● **0 = RADIO_MODE_AP**: AP ● **1 = RADIO_MODE_MESH_PORTAL**: Mesh portal ● **2 = RADIO_MODE_MESH_POINT**: Mesh point ● **3 = RADIO_MODE_AIR_MONITOR**: Air Monitor ● **4 = RADIO_MODE_SPECTRUM_SENSOR**: Spectrum sensor ● **5 = RADIO_MODE_UNKNOWN**: Unknown	✓	✓
phy	Physical radio type: ● **0 = PHY_TYPE_80211B**: 802.11b radio ● **1 - PHY_TYPE_80211A**: 802.11a radio ● **2 = PHY_TYPE_80211G**: 802.11g radio ● **3 = PHY_TYPE_80211AG**: 802.11ag radio ● **4 = PHY_TYPE_INVALID**: Invalid radio	✓	✓
ht	Type of high-throughput traffic sent by the AP: ● **0 = HTT_NONE**: No high-throughput traffic ● **1 = HTT_20MZ**: High-throughput traffic sent through a 20 MHz channel ● **2 = HTT_40MZ**: High-throughput traffic sent through a 40 MHz channel ● **3 = HTT_VHT_20MZ**: Very high-throughput traffic sent through a 20 MHz channel ● **4 = HTT_VHT_40MZ**: Very high-throughput traffic sent through a 40 MHz channel ● **5 = HTT_VHT_80MZ**: Very high-throughput traffic sent through an 80 MHz channel ● **6 = HTT_VHT_160MZ**: Very high-throughput traffic sent through a 160 MHz channel ● **7 = HTT_VHT_80PLUS80MZ**: Very high-throughput traffic sent through an 80+80 MHz channel ● **8 = HTT_INVALID**: Invalid high-throughput traffic	✓	✓
virtual_access_points	List of virtual APs	✓	✓
radio_num	Number of radios on the AP	✓	x

The Radio API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "radio"
Protobuf schema

message radio {
   optional mac_address ap_eth_mac = 1;
   optional mac_address radio_bssid = 2;
   optional radio_mode mode = 4;
   optional phy_type phy = 5;
   optional ht_type ht = 6;
   repeated virtual_access_point virtual_access_points = 7;
   optional uint32 radio_num = 8;
}
```

Radio Statistics

The Radio Statistics message returns information on network coverage, traffic between APs and devices, client performance, and potential causes of poor performance or connectivity. This API is available in both controller and IAP deployments.

The output for this message type displays the following information, but not all fields are present in both controller and IAP deployments. Refer to Table 43 and the example ouputs to determine which fields are present in each mode.

Table 43: *Radio Statistics API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	x
radio_number	Radio interference number for the given AP	✓	x
channel	Assigned channel for the radio	✓	x
phy_type	Physical radio type: ● **0 = PHY_TYPE_80211B**: 802.11b radio ● **1 - PHY_TYPE_80211A**: 802.11a radio ● **2 = PHY_TYPE_80211G**: 802.11g radio ● **3 = PHY_TYPE_80211AG**: 802.11ag radio ● **4 = PHY_TYPE_INVALID**: Invalid radio	✓	x
radio_mode	Specifies the radio mode on an AP; each mode provides a different function: ● **0 = RADIO_MODE_AP**: AP ● **1 = RADIO_MODE_MESH_PORTAL**: Mesh portal ● **2 = RADIO_MODE_MESH_POINT**: Mesh point ● **3 = RADIO_MODE_AIR_MONITOR**: Air Monitor ● **4 = RADIO_MODE_SPECTRUM_SENSOR**: Spectrum sensor ● **5 = RADIO_MODE_UNKNOWN**: Unknown	✓	x
noise_floor	Noise floor for the given radio	✓	✓
tx_power	Transmitting power for the given radio, in dB	✓	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
channel_utilization	Channel utilization for the given radio	✓	x
rx_channel_utilization	Channel utilization for incoming packets	✓	x
tx_channel_utilization	Channel utilization for outgoing packets	✓	x
tx_received	Total number of 802.11 frames received for transmission	✓	x
tx_transmitted	Total number of transmitted 802.11 frames	✓	x
tx_dropped	Total number of dropped 802.11 frames	✓	✓
tx_data_received	Total number of 802.11 data frames received for transmission	✓	x
tx_data_transmitted	Total number of transmitted 802.11 data frames	✓	x
tx_data_retried	Total number of retried 802.11 data frames	✓	x
rx_frames	Total number of received 802.11 frames	✓	x
rx_retried	Total number of retried 802.11 frames	✓	x
rx_data_frames	Total number of received 802.11 data frames	✓	x
rx_data_retried	Total number of received 802.11 data frames that are retried	✓	x
rx_frame_errors	Frames that the radio cannot decode	✓	✓
traffic_stats	Data traffic type performance statistics	✓	x
prio_stats	Data priority performance statistics	✓	x
rate_stats	Data rate bucket performance statistics	✓	x
actual_eirp	Actual EIRP	✓	x
radio_mac	MAC address of the radio	x	✓
tx_data_bytes	Amount of transmitted data traffic, in bytes	x	✓
rx_data_bytes	Amount of received data traffic, in bytes	x	✓
radio_band	Frequency band for the given radio: • **0 = 2.4 GHz** for **RADIO_BAND_G**: 2.4 GHz frequency band for an 802.11g radio. • **1 = 5 GHz** for **RADIO_BAND_A**: 5 GHz frequency band tor an 802.11a radio. • **2 = RADIO_BAND_AG**: The radio contains two frequency bands (2.4 GHz and 5 GHz), but only one band can be configured and used at a time. Use the **phy_type** field to determine if the radio is 802.11g (2.4 GHz) or 802.11a (5 GHz).	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_number	Number of stations associated with the AP radio	✓	x

Data Rate Stats

The **data_rate_stats** message filter returns information on the data rate statistics for the given station.

Table 44: *Radio Statistics API Message Parameters - data_rate_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
rate	Data rate	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
tx_byte_count	Number of transmitted bytes	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x
rx_byte_count	Number of recieved bytes	✓	x

Data Prio Stats

The **data_prio_stats** message filter returns information on the data priority statistics for the given station.

Table 45: *Radio Statistics API Message Parameters - data_prio_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
prio	Priority for the data type: ● 0 = DATA_PRIO_BK ● 1 = DATA_PRIO_BE ● 2 = DATA_PRIO_VI ● 3 = DATA_PRIO_VO	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x
tx_drop_count	Number of dropped transmission frames	✓	x

Data Traffic Type Stats

The **data_traffic_type_stats** message filter returns information on traffic statistics for the given station, based on the selected traffic type.

Table 46: *Radio Statistics API Message Parameters - data_traffic_type_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
type	Data traffic type: ● **0 = DATA_TRAFFIC_TYPE_BCAST**: Broadcast traffic ● **1 = DATA_TRAFFIC_TYPE_MCAST**: Multicast traffic ● **2 = DATA_TRAFFIC_TYPE_UCAST**: Unicast traffic	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x

The Radio Statistics API output schema is as follows:

```
ØMQ message filter “stats_radio”
Protobuf schema

message stats_radio {
   optional mac_address ap_eth_mac = 1;
   optional uint32 radio_number = 2;
   optional uint32 channel = 3;
   optional phy_type phy = 4;
   optional radio_mode mode = 5;
   optional uint32 noise_floor = 7;
   optional uint32 tx_power = 8;
   optional uint32 channel_utilization = 9;
   optional uint32 rx_channel_utilization = 10;
   optional uint32 tx_channel_utilization = 11;
   optional uint32 tx_received = 12;
   optional uint32 tx_transmitted = 13;
   optional uint32 tx_dropped = 14;
   optional uint32 tx_data_received = 15;
   optional uint32 tx_data_transmitted = 16;
   optional uint32 tx_data_retried = 17;
   optional uint32 rx_frames = 18;
   optional uint32 rx_retried = 19;
   optional uint32 rx_data_frames = 20;
   optional uint32 rx_data_retried = 21;
   optional uint32 rx_frame_errors = 22;
   repeated data_traffic_type_stats traffic_stats = 23;
   repeated data_prio_stats prio_stats = 24;
   repeated data_rate_stats rate_stats = 25;
   optional uint32 actual_eirp = 26;
   optional mac_address radio_mac = 27;
   optional uint64 tx_data_bytes = 28;
   optional uint64 rx_data_bytes = 29;
   optional uint32 radio_band = 30;
   optional uint32 sta_number = 32;
}
message data_rate_stats {
   optional uint32 rate = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 tx_byte_count = 3;
   optional uint32 rx_frame_count = 4;
   optional uint32 rx_byte_count = 5;
}
message data_prio_stats {
   optional data_prio prio = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 rx_frame_count = 3;
   optional uint32 tx_drop_count = 4;
}
message data_traffic_type_stats {
   optional traffic_type type = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 rx_frame_count = 3;
}
```

Radio Utilization/Histogram Statistics

The Radio Utilization/Histogram Statistics message displays information about the radio usage on an AP. This API is only available in controller deployments.

The output for this message type displays the following information:

Utilization Stats Radio

The **utilization_stats_radio** message filter returns information on the utilization statistics available for the AP.

Table 47: *Radio Utilization/Histogram API Message Parameters - utilization_stats_radio*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	x
radio_number	Radio interface number of the AP (0,1)	✓	x
ustats	List of all utilization statistics	✓	x

Util Stats

The **util_stats** message filter returns information on radio usage, based on the selected utilization statistic type.

Table 48: *Radio Utilization/Histogram API Message Parameters - util_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
type	Type of histogram/utilization statistic: ● 0 = UTIL_STAT_TYPE_CHANNEL ● 1 = UTIL_STAT_TYPE_CHANNEL_TX ● 2 = UTIL_STAT_TYPE_CHANNEL_RX ● 3 = UTIL_STAT_TYPE_QUEUE_SWTX ● 4 = UTIL_STAT_TYPE_QUEUE_BE ● 5 = UTIL_STAT_TYPE_QUEUE_BK ● 6 = UTIL_STAT_TYPE_QUEUE_VI ● 7 = UTIL_STAT_TYPE_QUEUE_VO	✓	x

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
	• 8 = UTIL_STAT_TYPE_QUEUE_BCMC • 9 = UTIL_STAT_TYPE_QUEUE_ATIM		
bucket1	Bucket 1, used for categorizing and grouping data	✓	x
bucket2	Bucket 2, used for categorizing and grouping data	✓	x
bucket3	Bucket 3, used for categorizing and grouping data	✓	x
bucket4	Bucket 4, used for categorizing and grouping data	✓	x
bucket5	Bucket 5, used for categorizing and grouping data	✓	x
max	Maximum radio usage on the AP	✓	x
min	Minimum radio usage on the AP	✓	x
curr	Current radio usage on the AP	✓	x
stat	Receives statuses based on the above nine fields	✓	x

The Radio Utilization/Histogram API output schema is as follows:

```
ØMQ message filter "utilization_stats_radio"
Protobuf schema:

message utilization_stats_radio {
   optional mac_address ap_eth_mac = 1;
   optional uint32 radio_number = 2;
   repeated util_stats ustats = 3;
}
message util_stats {
   optional util_stat_type type = 1;
   optional uint32 bucket1 = 2;
   optional uint32 bucket2 = 3;
   optional uint32 bucket3 = 4;
   optional uint32 bucket4 = 5;
   optional uint32 bucket5 = 6;
   optional uint32 max = 7;
   optional uint32 min = 8;
   optional uint32 curr = 9;
   optional uint64 stat = 10;
}
```

Station RSSI

The Station RSSI message returns information on the Received Signal Strength Indication (RSSI) value for a selected client station at a specific point in time. This API is available in both controller and IAP deployments.

NOTE

This API is available as a debug tool and should not be used.

The output for this message type displays the following information:

Table 49: *Station RSSI API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client	✓	✓
radio_mac	MAC address of the AP radio that hears the RSSI value	✓	✓
rssi_val	RSSI value at which the AP hears the station **NOTE:** Attenuation (dBm) = RSSI - 96 - client Tx power. If client power is unknown, a value of 10 is used.	✓	✓
associated	Indicates whether the client is associated with an AP on the network. A value of **1** indicates that the client is associated with an AP. A value of **0** indicates that the client is no longer associated with an AP.	✓	✓
age	Age of the RSSI value (for example, Age=0 indicates the most current RSSI value)	✓	✓
noise_floor	Noise floor for the radio that hears the RSSI value	✓	✓
assoc_bssid	MAC address of the associated client	✓	✓
classification_type	Radio classification type for a station (client) RSSI. ● **0 = RSTAT_VALID**: Valid client ● **1 = RSTAT_INTERFERING**: Interfering client ● **2 = RSTAT_DOS**: Disabled rogue client ● **3 = RSTAT_VALID_EXEMPT**: Valid and exempt client ● **4 = RSTAT_UNKNOWN**: Unknown client	✓	✓

The Station RSSI API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7778"
ØMQ message filter      "sta_rssi"
Protobuf schema

message sta_rssi {
   optional mac_address sta_eth_mac = 1;
   optional mac_address radio_mac = 2;
   optional sint32 rssi_val = 3;
   optional bool associated = 4;
   optional int32 age = 5;
   optional int32 noise_floor = 6;
   optional mac_address assoc_bssid = 7;
   optional uint32 classification_type = 8;
}
```

Security

The Security message provides information related to user authentication, strengthening security and ensuring only legitimate users are accessing the network. This API is only available in controller deployments.

The output for these message types display the following information:

Security Message

The **security_message** message filter returns information on the security method used for client authentication.

Table 50: *Security API Message Parameters - security_message*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
msg_type	Type of security message sent during client authentication: ● **0 - AUTH_SRVR_TIMEOUT_MSG**: Authentication server timeout ● **1 = MACAUTH_MSG**: MAC authentication ● **2 = CAPTIVE_PORTAL_MSG**: Captive Portal ● **3 = WPA_KEY_HANDSHAKE_MSG**: WPA handshake ● **4 = DOT1X_MSG**: Dot1X authentication ● **5 = UNKNOWN_MSG**: Unknown	✓	x
auth_srvr_timeout	Authentication server timeout	✓	x
macauth	MAC authentication	✓	x
captive_portal	Captive Portal authentication	✓	x
wpa_key_handshake	WPA handshake	✓	x
dot1x	Dot1X authentication	✓	x

Dot1x

The **dot1x** message filter returns information on Dot1x authentication.

Table 51: *Security API Message Parameters - dot1x*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
start_timestamp	Time that authentication begins	✓	x
finish_timestamp	Time that authentication ends	✓	x
station_mac	MAC address of the client attempting to authenticate to the network	✓	x
username	A unique identifier for the client	✓	x
bssid	BSSID of the client	✓	x

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
result	Indicates whether authentication is successful	✓	x
reason	Reason for a failed authentication attempt	✓	x
server_retry_cnt	Number of times the server attempts to authenticate a client to the network	✓	x
client_retry_cnt	Number of times the client attempts to authenticate to the network	✓	x
serverip	IP address of the server	✓	x
srvr_elapsed_time	Elapsed time it takes the server to authenticate a client to the network	✓	x
clnt_elapsed_time	Elapsed time is takes the client to authenticate to the network	✓	x

WPA Key Handshake

The **wpa_key_handshake** message filter returns information on authentication through a WPA handshake.

Table 52: *Security API Message Parameters - wpa_key_handshake*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
start_timestamp	Time that authentication begins	✓	x
finish_timestamp	Time that authentication ends	✓	x
station_mac	MAC address of the client attempting to authenticate to the network	✓	x
bssid	BSSID of the client	✓	x
result	Indicates whether authentication is successful	✓	x
trigger_reason	Reason the system triggers another authentication attempt	✓	x
reason	Reason for a failed authentication attempt	✓	x
key1_retry_cnt	The number of times a client attempts to authenticate to the network using WPA key 1	✓	x
key3_retry_cnt	The number of times a client attempts to authenticate to the network using WPA key 3	✓	x
replay_counter_mismatch	Displays mismatches in replay counter errors, indicating the response to an initial packet is not accepted if a later version of the packet is also sent out	✓	x

Captive Portal

The **captive_portal** message filter returns information on Captive Portal authentication.

Table 53: *Security API Message Parameters - captive_portal*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
start_timestamp	Time that authentication begins	✓	x
finish_timestamp	Time that authentication ends	✓	x
station_mac	MAC address of the client attempting to authenticate to the network	✓	x
username	A unique identifier for the client	✓	x
bssid	BSSID of the client	✓	x
result	Indicates whether authentication is successful	✓	x
reason	Reason for a failed authentication attempt	✓	x
server_retry_cnt	Number of times the server attempts to authenticate a client to the network	✓	x
serverip	IP address of the server	✓	x
userip	IP address of the client	✓	x

MAC Authentication

The **macauth** message filter returns information on MAC authentication.

Table 54: *Security API Message Parameters - macauth*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
start_timestamp	Time that authentication begins	✓	x
finish_timestamp	Time that authentication ends	✓	x
station_mac	MAC address of the client attempting to authenticate to the network	✓	x
bssid	BSSID of the client	✓	x
result	Indicates whether authentication is successful	✓	x
reason	Reason for a failed authentication attempt	✓	x
server_retry_cnt	Number of times the server attempts to authenticate a client to the network	✓	x
serverip	IP address of the server	✓	x

Authentication Server Timeout

The **auth_srvr_timeout** message filter returns information on authentication server timeouts.

Table 55: *Security API Message Parameters - macauth*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
timestamp	Time that the authentication server times out	✓	x
station_mac	MAC address of the client attempting to authenticate to the network	✓	x
bssid	BSSID of the client	✓	x
authtype	Method used to authenticate a client into the network: ● MAC authentication ● Captive Portal ● WPA handshake ● Dot1x ● Unknown	✓	x
retry_cnt	Number of times a client attempts to authenticate to the network after a timeout	✓	x
userip	IP address of the client	✓	x
serverip	IP address of the server	✓	x

The Security API output schema is as follows:

```
ØMQ message filter “security_message”
Protobuf schema:

message security_message {
   optional security_msg_type msg_type = 1
   optional auth_srvr_timeout auth_srvr_timeout = 2;
   optional macauth macauth = 3;
   optional captive_portal captive_portal = 4;
   optional wpa_key_handshake wpa_key_handshake = 5;
   optional dot1x dot1x = 6;
}
message dot1x {
   optional uint64 start_timestamp = 1;
   optional uint64 finish_timestamp = 2;
   optional mac_address station_mac = 3;
   optional string username = 4;
   optional mac_address bssid = 5;
   optional uint32 result = 6;
   optional uint32 reason = 7;
   optional uint32 server_retry_cnt = 8;
   optional uint32 client_retry_cnt = 9;
   optional ip_address serverip = 10;
   optional uint32 srvr_elapsed_time = 11;
   optional uint32 clnt_elapsed_time = 12;
}

message wpa_key_handshake {
   optional uint64 start_timestamp = 1;
   optional uint64 finish_timestamp = 2;
   optional mac_address station_mac = 3;
   optional mac_address bssid = 4;
   optional uint32 result = 5;
   optional uint32 trigger_reason = 6;
   optional uint32 reason = 7;
   optional uint32 key1_retry_cnt = 8;
   optional uint32 key3_retry_cnt = 9;
   optional uint32 replay_counter_mismatch = 10;
}
message captive_portal {
   optional uint64 start_timestamp = 1;
   optional uint64 finish_timestamp = 2;
   optional mac_address station_mac = 3;
   optional string username = 4;
   optional mac_address bssid = 5;
   optional uint32 result = 6;
   optional uint32 reason = 7;
   optional uint32 server_retry_cnt = 8;
   optional ip_address serverip = 9;
   optional ip_address userip = 10;
}

message macauth {
   optional uint64 start_timestamp = 1;
   optional uint64 finish_timestamp = 2;
   optional mac_address station_mac = 3;
   optional mac_address bssid = 4;
   optional uint32 result = 5;
   optional uint32 reason = 6;
   optional uint32 server_retry_cnt = 7;
   optional ip_address serverip = 8;
}

message auth_srvr_timeout {
   optional uint64 timestamp = 1;
   optional mac_address station_mac = 2;
   optional mac_address bssid = 3;
   optional uint32 authtype = 4;
   optional uint32 retry_cnt = 5;
   optional ip_address userip = 6;
   optional ip_address serverip = 7;
}
```

Station

The Station message is sent when a user associates to the WLAN. This API is available in both controller and IAP deployments.

NOTE

The station is removed from the ALE table if the controllers also remove the station from its user-table. The user idle-timeout for removal is 5 minutes by default and is configured using "aaa timers idle-timeout x", where x is the number of minutes until the user is idled out of the system.

The output for this message type displays the following information:

Table 56: *Station API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client station	✓	✓
username	Corresponding username from the user table on the WLAN controller or IAP	✓	✓
role	Name of the user role currently assigned to the client. This is only applicable to authenticated users	✓	✓
bssid	BSSID of the client	✓	✓
device_type	Type of device used by the client ● Windows 7 ● iOS devices	✓	✓
sta_ip_address	IP address of the client station	✓	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	✓	✓
hashed_sta_ip_address	Anonymized value of the client IP address	✓	✓
ht	Type of high-throughput traffic sent by the AP: ● **0 = HTT_NONE**: No high-throughput traffic ● **1 = HTT_20MZ**: High-throughput traffic sent through a 20 MHz channel ● **2 = HTT_40MZ**: High-throughput traffic sent through a 40 MHz channel ● **3 = HTT_VHT_20MZ**: Very high-throughput traffic sent through a 20 MHz channel ● **4 = HTT_VHT_40MZ**: Very high-throughput traffic sent through a 40 MHz channel	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
	• **5 = HTT_VHT_80MZ**: Very high-throughput traffic sent through an 80 MHz channel • **6 = HTT_VHT_160MZ**: Very high-throughput traffic sent through a 160 MHz channel • **7 = HTT_VHT_80PLUS80MZ**: Very high-throughput traffic sent through an 80+80 MHz channel • **8 = HTT_INVALID**: Invalid high-throughput traffic		
ap_name	Name of the AP	✓	✓

The Station API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "station"
Protobuf schema

message station {
   optional mac_address sta_eth_mac = 1;
   optional string username = 2;
   optional string role = 3;
   optional mac_address bssid = 4;
   optional string device_type = 5;
   optional ip_address sta_ip_address = 6;
   optional bytes hashed_sta_eth_mac = 7;
   optional bytes hashed_sta_ip_address = 8;
   optional ht_type ht = 9;
   optional string ap_name = 10;
}
```

Station Statistics

The Station Statistics message returns performance information for a given station after a client associates to the station and successfully authenticates to the network. This API is available in both controller and IAP deployments, but not all fields are present in both deployments. Refer to the schema below to determine which fields are present in each mode.

The output for these message types display the following information:

Stats Station

The **stats_station** message filter returns performance information for the given station.

Table 57: *Station Statistics API Message Parameters - stats_station*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client station	✓	✓
ap_eth_mac	MAC address of the AP associated with the station	✓	x

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
bssid	BSSID of the client station	✓	x
snr	Signal-to-noise ratio used to measure the station signal strength	✓	✓
tx_received	Total number of 802.11 frames received for transmission	✓	x
tx_transmitted	Total number of transmitted 802.11 frames	✓	x
tx_dropped	Total number of dropped 802.11 frames	✓	x
tx_data_received	Total number of 802.11 data frames received for transmission	✓	x
tx_data_transmitted	Total number of transmitted 802.11 data frames	✓	x
tx_data_retried	Total number of retried 802.11 data frames	✓	x
rx_data_received	Total number of received 802.11 data frames	✓	x
rx_data_retried	Total number of received 802.11 data frames that are retried	✓	x
prio_stats	Data priority performance statistics	✓	x
rate_stats	Data rate bucket performance statistics	✓	x
speed	Application usage on the station	x	✓
rx_rate	Connected data rate for received data	x	✓
tx_rate	Connected data rate for transmitted data	x	✓
rx_data_bytes	Amount of received data traffic, in bytes	x	✓
tx_data_bytes	Amount of transmitted data traffic, in bytes	x	✓
ssid_up	Number of SSIDs or VAPs	x	✓
rogue_ap	Number of rogue APs detected	x	✓
hashed_sta_eth_mac	Anonymized value of the station MAC address	✓	✓
max_tx_rate	Maximum transmission rate	x	✓
tx_data_bytes_transmitted	Amount of transmitted tx data traffic, in bytes	x	✓
tx_time_data	Data transmission time	x	✓
rx_time_data	Data receival time	x	✓
sta_client_health	Station health	x	✓
rx_retries	Retry count for frames received by a station	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_retries	Retry count for frames transmitted by a station	x	✓

Data Rate Stats

The **data_rate_stats** message filter returns information on the data rate statistics for the given station.

Table 58: *Station Statistics API Message Parameters - data_rate_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
rate	Data rate	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
tx_byte_count	Number of transmitted bytes	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x
rx_byte_count	Number of recieved bytes	✓	x

Data Prio Stats

The **data_prio_stats** message filter returns information on the data priority statistics for the given station.

Table 59: *Station Statistics API Message Parameters - data_prio_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
prio	Priority for the data type: ● 0 = DATA_PRIO_BK ● 1 = DATA_PRIO_BE ● 2 = DATA_PRIO_VI ● 3 = DATA_PRIO_VO	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x
tx_drop_count	Number of dropped transmission frames	✓	x

Data Traffic Type Stats

The **data_traffic_type_stats** message filter returns information on traffic statistics for the given station, based on the selected traffic type.

Table 60: *Station Statistics API Message Parameters - data_traffic_type_stats*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
type	Data traffic type: ● **0 = DATA_TRAFFIC_TYPE_BCAST**: Broadcast traffic ● **1 = DATA_TRAFFIC_TYPE_MCAST**: Multicast traffic ● **2 = DATA_TRAFFIC_TYPE_UCAST**: Unicast traffic	✓	x
tx_frame_count	Number of transmitted 802.11 frames	✓	x
rx_frame_count	Number of received 802.11 frames	✓	x

The Station Statistics API output schema is as follows for a controller deployment:

```
ØMQ message filter "stats_station"
Protobuf schema:

message stats_station {
   optional mac_address sta_eth_mac = 1;
   optional mac_address ap_eth_mac = 2;
   optional mac_address bssid = 3;
   optional uint32 snr = 4;
   optional uint32 tx_received = 5;
   optional uint32 tx_transmitted = 6;
   optional uint32 tx_dropped = 7;
   optional uint32 tx_data_received = 8;
   optional uint32 tx_data_transmitted = 9;
   optional uint32 tx_data_retried = 10;
   optional uint32 rx_data_received = 11;
   optional uint32 rx_data_retried = 12;
   repeated data_prio_stats prio_stats = 13;
   repeated data_rate_stats rate_stats = 14;
}
message data_rate_stats {
   optional uint32 rate = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 tx_byte_count = 3;
   optional uint32 rx_frame_count = 4;
   optional uint32 rx_byte_count = 5;
}
message data_prio_stats {
   optional data_prio prio = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 rx_frame_count = 3;
   optional uint32 tx_drop_count = 4;
}
message data_traffic_type_stats {
   optional traffic_type type = 1;
   optional uint32 tx_frame_count = 2;
   optional uint32 rx_frame_count = 3;
```

The Station Statistics API output schema is as follows for an IAP deployment:

```
ØMQ message filter "stats_station"
Protobuf schema:

message stats_station {
   optional mac_address sta_eth_mac = 1;
   optional uint32 snr = 2;
   optional uint32 speed = 3;
   optional uint64 rx_rate = 4;
   optional uint64 tx_rate = 5;
   optional uint64 rx_data_bytes = 6;
   optional uint64 tx_data_bytes = 7;
   optional uint32 ssid_up = 8;
   optional uint32 rogue_ap = 9;
   optional bytes hashed_sta_eth_mac = 10
   optional uint64 max_tx_rate = 11;
   optional uint64 tx_data_bytes_transmitted = 12;
   optional uint32 tx_time_data = 13;
   optional uint32 rx_time_data = 14;
   optional uint32 sta_client_health = 15;
   optional uint32 rx_retries = 16;
   optional uint32 tx_retries = 17;
}
```

State Station

The State Station message provides information about the state of a station after a client associates to the station and successfully authenticates to the network. This API is only available in IAP deployments.

The output for this message type displays the following information:

Table 61: *State Station API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
sta_eth_mac	MAC address of the client station	x	✓
ap_eth_mac	BSSID of the client station	x	✓
snr	Signal strength of the station	x	✓
rx_tries	Retry count for frames received by a station	x	✓
tx_tries	Retry count for frames transmitted by a station	x	✓
phy_type	Physical radio type: ● **0 = PHY_TYPE_80211B**: 802.11b radio ● **1 - PHY_TYPE_80211A**: 802.11a radio ● **2 = PHY_TYPE_80211G**: 802.11g radio ● **3 = PHY_TYPE_80211AG**: 802.11ag radio ● **4 = PHY_TYPE_INVALID**: Invalid radio	x	✓
security_type	Station security type: ● **0 = OPENSYSTEM**: Open sytem security ● **1 = STATIC_WEP**: Static WEP key ● **2 = DYNAMIC_WEP**: Dynamic WEP key ● **3 = WPA_TKIP**: WPA with TKIP encryption ● **4 = WPA_AES**: WPA with AES encryption ● **5 = WPA_PSK_TKIP**: WPA-PSK with TKIP encryption ● **6 = WPA_PSK_AES**: WPA-PSK with AES encryption ● **7 = WPA2_AES**: WPA2 with AES encryption ● **8 = WPA2_TKIP**: WPA2 with TKIP encryption ● **9 = WPA2_PSK_AES**: WPA2-PSK with AES encryption	x	✓
hashed_sta_eth_mac	Anonymized value of the client MAC address	x	✓

The State Station output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "state_station"
Protobuf schema

message state_station {
   optional mac_address sta_eth_mac = 1;
   optional mac_address ap_eth_mac = 2;
   optional uint32 snr = 3;
   optional uint32 rx_tries = 4;
   optional uint32 tx_tries = 5;
   optional phy_type phy_type = 6;
   optional uint32 security_type = 7;
   optional bytes hashed_sta_eth_mac = 8;
}
```

Virtual Access Point (VAP)

The Virtual Access Point message is sent for each virtual AP (VAP) associated with a newly added AP. This API is available in both controller and IAP deployments unless an exception has specifically been called out.

The output for this message type displays the following information:

Table 62: *VAP API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
bssid	BSSID of the virtual AP	✓	✓
ssid	SSID of the virtual AP	✓	✓
radio_bssid	BSSID of a radio on the virtual AP	✓	✓

The VAP API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "virtual_access_point"
Protobuf schema

message virtual_access_point {
   optional mac_address bssid = 1;
   optional string ssid = 2;
   optional mac_address radio_bssid = 3;
```

Virtual Access Point (VAP) Statistics

The Virtual Access Point (VAP) Statistics message provides performance information for virtual APs. The VAP Statistics API is only available in controller deployments.

The output for this message type displays the following information:

Table 63: *VAP Statistics API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_eth_mac	MAC address of the AP	✓	x
bssid	BSSID of the virtual AP	✓	x
tx_received	Total number of 802.11 frames received for transmission	✓	x
tx transmitted	Total number of transmitted 802.11 frames	✓	x
tx_dropped	Total number of dropped 802.11 frames	✓	x
tx_data_received	Total number of 802.11 data frames received for transmission	✓	x
tx_data_transmitted	Total number of transmitted 802.11 data frames	✓	x
tx_data_retried	Total number of retried 802.11 data frames	✓	x
rx_frames	Number of received 802.11 frames	✓	x
rx_retried	Total number of received 802.11 frames that are retried	✓	x
rx_data_frames	Total number of received 802.11 data frames	✓	x
rx_data_retried	Total number of received 802.11 data frames that are retried	✓	x
traffic_stats	Data traffic type statistics	✓	x
prio_stats	Priority statistics for the given station	✓	x
rate_stats	Data rate bucket statistics for the given station	✓	x
sta_number	Number of stations associated with the virtual AP	✓	x

The VAP Statistics API output schema is as follows:

```
ØMQ message filter "stats_vap"
Protobuf schema:

message stats_vap {
   optional mac_address ap_eth_mac = 1;
   optional mac_address bssid = 2;
   optional uint32 tx_received = 5;
   optional uint32 tx_transmitted = 6;
   optional uint32 tx_dropped = 7;
   optional uint32 tx_data_received = 8;
   optional uint32 tx_data_transmitted = 9;
   optional uint32 tx_data_retried = 10;
   optional uint32 rx_frames = 18;
   optional uint32 rx_retried = 19;
   optional uint32 rx_data_frames = 11;
   optional uint32 rx_data_retried = 12;
   repeated data_traffic_type_stats traffic_stats = 23;
   repeated data_prio_stats prio_stats = 13;
   repeated data_rate_stats rate_stats = 14;
   optional uint32 sta_number = 15;
}
```

WebCC

The Web Content Classification (WebCC) message provides information about the types of websites clients visit when they are connected to the network. This also provides safety/security information about each website (for example, whether the website contains malicious malware, spyware, or adware).

The message filters for the WebCC API include **webcc_category** and **webcc_info**. The **webcc_category** filter is available in both controller and IAP deployments, while **webcc_info** is only available in controller deployments. In IAP deployments, information similar to the **webcc_info** output is available through the Visibility Record API.

NOTE

The WebCC API is only used by HTTP/HTTPS.

The output for these message types display the following information:

WebCC Category

The **webcc_category** message filter returns information on the WebCC category of the website.

Table 64: *WebCC API Message Parameters - webcc_category*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
cat_id	ID number specifying the website category	✓	✓
category	The website category	✓	✓

WebCC Info

The **webcc_info** message filter returns WebCC information for the given website, including the category and security risks.

Table 65: *WebCC API Message Parameters - webcc_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
webcc_md5	The 16 byte md5 checksum of the `webcc_url_prefix`. This field maps to the table build from the `webcc_info` feed.	✓	x
webcc_cat_id	ID number specifying the WebCC category	✓	x
webcc_rep_score	Reputation score for websites based on security risks (for example, malware and phishing)	✓	x
webcc_url_prefix	Maps the local WebCC file with a URL	✓	x
webcc_not_supported	Indicates if WebCC is not supported on the given controller. This field is only available in controller versions 6.4.3.0 and earlier.	✓	x

The following WebCC message filter builds the `webcc_category` table. In controller deployments, this message is only sent when the website category changes. In IAP deployments, this message is sent when ALE starts up.

```
ØMQ message filter "webcc_category"
Protobuf schema:

message webcc_category {
   optional uint32 cat_id = 1;
   optional string category = 2;
}
```

The following WebCC message filter displays WebCC information relevant to controllers. In IAP deployments, WebCC information is embedded as fields in the Visibility Records API.

```
ØMQ message filter "webcc_info"
Protobuf schema:
   message webcc_info {
      optional bytes webcc_md5 = 1;
      optional uint32 webcc_cat_id = 2;
      optional uint32 webcc_rep_score = 3;
      optional string webcc_url_prefix = 4;
      optional bool webcc_not_supported = 5;
   }
}
```

NOTE

After md5 maps to the webcc_info table build, the tables built from the webcc_category or REST API can generate the string category fields.

Visibility Record

The Visibility Record message represents a unique station session for each associated client on a given application or destination. This API is available in both controller and IAP deployments.

The output for this message type displays the following information:

Table 66: *Visibility Record API Message Parameters*

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
client_ip	IP address of the client	✓	✓	✓
dest_ip	IP address of the client receiving traffic through the network	✓	✓	✓
ip_proto	Protocol defining packet formatting and the addressing scheme: ● **6 = IP_PROTOCOL_VAL_6**: Transmission Control Protocol (TCP) ● **17 = IP_PROTOCOL_VAL_17**: User Datagram Protocol (UDP)	✓	x	✓
app_id	ID number identifying the application	✓	✓	✓
session_flags	Session flag indicating the application enforcement status: ● **ENF_PERMIT**: Permit ● **ENF_DENY**: Deny	x	✓	x

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
tx_pkts	Number of packets transmitted by the client	✓	x	✓
tx_bytes	Number of bytes transmitted by the client	✓	✓	✓
rx_pkts	Number of packets received by the client	✓	x	✓
rx_bytes	Number of bytes received by the client	✓	✓	✓
hashed_client_ip	Anonymized value of the client IP address	✓	✓	✓
ap_mac	MAC address of the access point **NOTE:** For controller and BoC deployments, a MAC address of **00:00:00:00:00:00** indicates that it is a wired interface, as no BSSIDs are available.	✓	✓	✓
device_mac	MAC address of the client device	x	✓	x
hashed_device_mac	Anonymized value of the client MAC address	x	✓	x
cc_cat_id	ID number specifying the WebCC category	x	✓	x
cc_rep_score	Reputation score for websites based on security risks (for example, malware and phishing)	x	✓	x
cc_url_prefix	Specifies the URL prefix of the visited site	✓	x	✓
cc_md5	Webroot md5 checksum of the `cc_url_prefix`. This field can be used to look up the table build from the `webcc_info` feed.	✓	x	✓

The Visibility Record API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "visibility_rec"
Protobuf schema:

message visibility_rec {
   optional ip_address client_ip = 1;
   optional ip_address dest_ip = 2;
   optional ip_protocol ip_proto = 3;
   optional uint32 app_id = 4;
   optional IapAppEnforcementStatus session_flags = 5;
   optional uint64 tx_pkts = 6;
   optional uint64 tx_bytes = 7;
   optional uint64 rx_pkts = 8;
   optional uint64 rx_bytes = 9;
   optional bytes hashed_client_ip = 10;
   optional mac_address ap_mac = 11;
   optional mac_address device_mac = 12;
   optional bytes hashed_device_mac = 13;
   optional uint32 cc_cat_id = 15;
   optional uint32 cc_rep_score = 16;
   optional string cc_url_prefix = 17;
   optional bytes cc_md5 = 18;
}
```

The cc_cat_id and cc_rep_score parameters are not available for applications or sessions without a valid/extractable HTTP/HTTPS URL or URI. These sessions only contain an app-id.

Controller Info

The Controller Info message returns the list of controllers within a network.

The output for this message type displays the following information:

Table 67: *Controller Info API Message Parameter*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
controller_ip_address	IP address of the controller	✓	x

The Controller API output schema is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “controller”
Protobuf schema

message controller {
   required ip_address controller_ip_address = 1;
}
```

OP_UPDATE and OP_DELETE are never sent for the Controller API.

Cluster Info

The Cluster Info message returns information about clusters in an IAP deployment.

The output for this message type displays the following information:

Table 68: *Cluster Info API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
cluster_key	Unique key identifying the cluster deployment	x	✓
cluster_name	Name of the cluster	x	✓
organization	Name of the organization	x	✓
cluster_ip	IP address of the cluster	x	✓

The Cluster Info API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "cluster"
Protobuf schema

message cluster {
   optional string cluster_key = 1;
   optional string cluster_name = 2;
   optional string organization = 3;
   optional ip_address cluster_ip = 4;
}
```

NOTE

OP_UPDATE and OP_DELETE are never sent for the Cluster Info API.

Uplink Bandwidth

The Uplink Bandwidth message returns information about the connection capacity, or bandwidth, between users and the Iperf server. This provides information on network throughput and indicates the quality of applications, traffic, and ports. The Uplink Bandwidth API is only available in IAP deployments.

The output for this message type displays the following information:

Table 69: *Uplink Bandwidth API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
server_ip	IP address of the Iperf server	x	✓
server_port	Iperf server port to which clients connect	x	✓
local_ip	IP address of the IAP Iperf client	x	✓
local_port	IAP Iperf client port connected to the server	x	✓
ap_mac	MAC address of the IAP	x	✓
ap_name	Name of the AP	x	✓
timestamp	Iperf test execution time stamp	x	✓
direction	Direction of test traffic (upstream or downstream)	x	✓
protocol	Protocol used for data communication during the test: ● TCP ● UDP	x	✓
interval	Measurement interval for the bandwidth test	x	✓
upstream_bytes	Number of bytes transferred during the upstream speed test	x	✓
upstream_bandwidth	Upstream bandwidth, in bits per second (bps)	x	✓
upstream_retries	Number of retries for the upstream TCP test	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
downstream_bytes	Number of bytes received for the downstream speed test	x	✓
downstream_bandwidth	Downstream bandwidth, in bits per second (bps)	x	✓
downstream_retries	Number of retries for the downstream TCP test	x	✓
upstream_datagrams	Number of datagrams transmitted through UDP	x	✓
downstream_datagrams	Number of datagrams received through UDP	x	✓
upstream_jitter	Upstream UDP traffic jitter, in milliseconds (ms)	x	✓
downstream_jitter	Downstream UDP traffic jitter, in milliseconds (ms)	x	✓
upstream_lost_packets	Percentage of lost upstream UDP packets	x	✓
downstream_lost_packets	Percentage of lost downstream UDP packets	x	✓
hashed_ap_eth_mac	Anonymized value of the client MAC address	x	✓

The Uplink Bandwidth API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "uplink_bandwidth"
Protobuf schema

message uplink_bandwidth {
   optional ip_address server_ip = 1;
   optional uint32 server_port =2;
   optional ip_address local_ip =3;
   optional uint32 local_port =4;
   optional mac_address ap_mac =5;
   optional string ap_name =6;
   optional uint64 timestamp=7;
   optional string protocol = 8;
   optional uint32 interval = 9;
   optional uint32 upstream_bytes = 10;
   optional uint32 upstream_bandwidth = 11;
   optional uint32 upstream_retries = 12;
   optional uint32 downstream_bytes = 13;
   optional uint32 downstream_bandwidth = 14;
   optional uint32 downstream_retries = 15;
   optional uint32 upstream_datagrams = 16;
   optional uint32 downstream_datagrams = 17;
   optional string upstream_jitter = 18;
   optional string downstream_jitter =19;
   optional uint32 upstream_lost_packets =20;
   optional uint32 downstream_lost_packets =21;
   optional bytes hashed_ap_eth_mac =22
}
```

Geofence Notify

The Geofence Notify message returns information when a device enters or leaves a GeoFence region. The Geofence Notify API is only available under context modes with device location (estimation or calibration) in both controller and IAP deployments.

The output for this message type displays the following information:

Table 70: *Geofence Notify API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
geofence_event	Notification triggered when a device enters or leaves a GeoFence region ● **0 = ZONE_IN**: The device is inside a GeoFence region ● **1 = ZONE_OUT**: The device is outside a GeoFence region	✓	✓
geofence_id	ID number identifying a specific GeoFence region	✓	✓
geofence_name	Name of the GeoFence region	✓	✓
sta_mac	MAC address of the client station	✓	✓
associated	Indicates whether the client is associated with an AP on the network. A value of **1** indicates that the client is associated with an AP. A value of **0** indicates that the client is no longer associated with an AP.	✓	✓
dwell_time	Amount of time the device must be inside or outside a GeoFence region to trigger a notification	✓	✓
access_point_info	Information about the AP to which the client is associated	✓	✓
ap_mac	MAC address of the AP	✓	✓
ap_name	Name of the AP	✓	✓
radio_bssid	BSSID of the radio on the AP	✓	✓
rssi_val	RSSI value at which the AP hears the station **NOTE:** Attenuation (dBm) = RSSI - 96 - client Tx power. If client power is unknown, a value of 10 is used.	✓	✓
hashed_sta_mac	Anonymized value of the client MAC address	✓	✓

The Geofence Notify API output schema is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “geofence_notify”
Protobuf schema

message geofence_notify {
   optional zone_event geofence_event = 1;
   optional bytes geofence_id = 2;
   optional string geofence_name = 3;
   optional mac_address sta_mac = 4;
   optional bool associated = 5;
   optional uint32 dwell_time = 6 [default=0];
```

```
    repeated group Access_point_info = 7;
    optional mac_address ap_mac = 8;
    optional string ap_name = 9;
    optional mac_address radio_bssid = 10;
    optional uint32 rssi_val = 11;
    optional bytes hashed_sta_mac = 30;
}
```

Client URL

The Client URL message returns information on URL visibility support. The Client URL API is only available in IAP deployments.

The output for these message types display the following information:

IAP Client URL

The **iap_client_url** message filter returns general information on URL visibility support.

Table 71: *Client URL API Message Parameters - iap_client_url*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
info	Common information across all message types (for example, all ALE feeds from the IAP)	x	✓
timestamp	Timestamp of the message origination, not specific to any independent URL record	x	✓
ap_mac	MAC address of the IAP from which the message is sent	x	✓
url_record	Repeated record of URL record data. Refer to the output below to view the URL record fields	x	✓
ap_host_name	Hostname of the IAP from which the message is sent. The hostname can have a maximum of 64 characters.	x	✓

URL Detail Record

The **url_detail_record** message filter returns information on the URL that is being browsed by a client.

Table 72: *Client URL API Message Parameters - url_detail_record*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
client_ip	Client/source IP address where the particular URL is browsed or extracted	x	✓
dest_ip	Destination IP address of the specified URL resolved on the client	x	✓
url	Actual URL data in the byte stream	x	✓
hit_count	Number of times similar URL data is browsed by the same client with the same destination	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
client_mac	MAC address of the client	x	✓
http_method	HTTP method used to perform a specific action: • **1 = NON_HTTP**: Non-HTTP traffic (HTTPS or QUIC) • **2 = HTTP_GET**: Requests data from the specified source • **3 = HTTP_PUT**: Replaces existing data at the specified source • **4 = HTTP_POST**: Sends new data to the specified source • **5 = HTTP_HEAD**: Requests data from the specified source without a message body (only message headers)	x	✓
last_hit_timestamp	Last hit timestamp for a specific URL in the client session. When the same URL is accessed by the same client multiple times, this field only returns the timestamp for the last instance in which the URL is accessed.	x	✓

IAP Message Info

The **iap_message_info** message filter returns information on messages sent by the IAP.

Table 73: *Client URL API Message Parameters - iap_message_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
version	Version of the IAP request message	x	✓
sequence	Sequence number of the IAP message, which can be used to reorder the IAP message feed	x	✓
guid	Globally unique identifier (GUID) of the IAP cluster feed, presented as a VC key	x	✓
oem_tag	Tag identifying the original equipment manufacturer (OEM), if applicable	x	✓

The Client URL API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "client_url"
Protobuf schema

message iap_client_url {
   optional iap_message_info info = 1;
   optional uint64 timestamp = 2;
   optional mac_address ap_mac = 3;
   optional url_detail_record url_record = 4;
   optional string ap_host_name = 5;
}
```

```
message iap_message_info {
   optional uint32 version = 1;
   optional uint64 sequence = 2;
   optional bytes guid = 3;
   optional string oem_tag = 4;
}
message url_detail_record {
   enum UrlHttpMethod {
      NON_HTTP = 1;
      HTTP_GET = 2;
      HTTP_PUT = 3;
      HTTP_POST = 4;
      HTTP_HEAD = 5;
   }
   optional ip_address client_ip = 1;
   optional ip_address dest_ip = 2;
   optional bytes url = 3;
   optional uint32 hit_count = 4;
   optional mac_address client_mac = 5;
   optional UrlHttpMethod http_method = 6;
   optional uint64 last_hit_timestamp = 7;
}
```

Rogue Info

The Rogue Info message returns information on unauthorized rogue devices that can potentially disrupt network operations. This API is only available in IAP deployments.

The output for this message type displays the following information:

Table 74: *Rogue Info API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
monitor_mac	MAC address of the Air Monitor that hears the rogue device	x	✓
heard_mac	MAC address of the rogue device	x	✓
monitor_channel	Channel on which the air monitor hears the rogue device	x	✓
is_ap	Indicates if the rogue device is an AP. A value of **1** indicates that the rogue device is an AP. A value of **0** indicates that the rogue device is not an AP.	x	✓

The Rogue Info API output schema is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "rogue_info"
Protobuf schema

message rogue_info {
   optional mac_address monitor_mac = 1;
   optional mac_address heard_mac = 2;
   optional uint32 monitor_channel = 3;
   optional uint32 is_ap = 4;
}
```

Air Monitor Info

The Air Monitor Info message returns information on devices that are discovered by Aruba Air Monitor, which analyzes channels to detect potential wireless attacks from neighboring APs and clients. Air Monitor classifies APs and clients as valid, interfering, or rogue. This API is only available in IAP deployments.

The output for this message type displays the following information:

Table 75: *Air Monitor Info API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
monitor_mac	MAC address of the AP or client detected by Air Monitor	x	✓
essid	ESSID of the AP or the AP to which the client is associated	x	✓
monitor_channel	Channel on which the air monitor hears the AP or client	x	✓
is_ap	Indicates if the device is an AP. A value of **1** indicates that the device is an AP. A value of **0** indicates that the device is not an AP.	x	✓
classification_type	Classification type for an AP or station (client) RSSI AP RSSI: ● **0 = RAPT_VALID**: Valid AP ● **1 = RAPT_INTERFERING**: Interfering AP ● **2 = RAPT_UNSECURE**: Rogue AP ● **3 = RAPT_DOS**: Disabled rogue AP ● **4 = RAPT_UNKNOWN**: Unknown AP ● **5 = RAPT_KNOWN_INTERFERING**: Known interfering AP ● **6 = RAPT_SUSPECT_UNSECURE**: Suspected rogue AP Station RSSI: ● **0 = RSTAT_VALID**: Valid client ● **1 = RSTAT_INTERFERING**: Interfering client ● **2 = RSTAT_DOS**: Disabled rogue client	x	✓
phy_type	Physical radio type: ● **0 = WIFI_80211B**: 802.11b Wi-Fi radio ● **1 = WIFI_80211A**: 802.11a Wi-Fi radio ● **2 = WIFI_80211G**: 802.11g Wi-Fi radio	x	✓
ht_type	Type of high-throughput traffic sent by the AP: ● **0 = HTT_NONE**: No high-throughput traffic ● **1 = HTT_20MZ**: High-throughput traffic sent through a 20 MHz channel ● **2 = HTT_40MZ**: High-throughput traffic sent through a 40 MHz channel ● **3 = HTT_VHT_20MZ**: Very high-throughput traffic sent through a 20 MHz channel ● **4 = HTT_VHT_40MZ**: Very high-throughput traffic sent through a 40 MHz channel ● **5 = HTT_VHT_80MZ**: Very high-throughput traffic sent through an 80 MHz channel ● **6 = HTT_VHT_160MZ**: Very high-throughput traffic sent through a 160 MHz channel ● **7 = HTT_VHT_80PLUS80MZ**: Very high-throughput	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
	traffic sent through an 80+80 MHz channel ● **8 = HTT_INVALID**: Invalid high-throughput traffic		
heard_mac	Wireless MAC address of the scanner AP (bss)	x	✓
bssid	BSSID of the AP or the AP to which the client is associated	x	✓
encr_type	Encryption protocol type, which is only available for APs: ● **0 = WMS_SNMP_WPA_ENCR_OPEN**: Open source encryption ● **1 = WMS_SNMP_WPA_ENCR_WEP**: WEP encryption ● **2 = WMS_SNP_WPA_ENCR_WPA**: WPA encryption ● **3 = WMS_SNMP_WPA_ENCR_WPA2**: WPA2 encryption	x	✓
encr_cipher	Encryption cipher mode, which is only available for APs: ● **0 = WMS_SNMP_WPA_CIPHER_NONE**: None ● **1 = WMS_SNMP_WPA_CIPHER_WEP40**: WEP 40 cipher mode ● **2 = WMS_SNMP_WPA_CIPHER_WEP104**: WEP 104 cipher mode ● **3 = WMS_SNMP_WPA_CIPHER_TKIP**: TKIP cipher mode ● **4 = WMS_SNMP_WPA_CIPHER_AESCCMP**: AES-CCMP cipher mode ● **5 = WMS_SNMP_WPA_CIPHER_OTHER**: Other cipher mode	x	✓
auth_alg	Authorization algorithm class, which is only available for APs: ● **0 = WMS_SNMP_WPA_AUTH_NONE**: None ● **1 = WMS_SNMP_WPA_AUTH_PSK**: PSK authorization algorithm ● **2 = WMS_SNMP_WPA_AUTH_8021X**: 802.1X authorization algorithm ● **3 = WMS_SNMP_WPA_AUTH_OTHER**: Other authorization algorithm	x	✓

The Air Monitor Info API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "air_monitor_info"
Protobuf schema

message air_monitor_info {
   optional mac_address monitor_mac = 1;
   optional bytes essid = 2;
   optional uint32 monitor_channel = 3;
   optional uint32 is_ap = 4;
   optional uint32 classification_type = 5;
   optional uint32 phy_type = 6;
   optional uint32 ht_type = 7;
   optional mac_address heard_mac = 8;
   optional mac_address bssid = 9;
   optional uint32 encr_type = 10;
   optional uint32 encr_cipher = 11;
   optional uint32 auth_alg = 12;
```

Spectrum Info

The Spectrum Info message provides visilibility into network RF interference, which can cause connectivity and performance issues in a deployment. APs scan each channel periodically to monitor and detect RF interference within a wireless environment.

The output for this message type displays the following information:

Table 76: *Spectrum Info API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_mac	MAC address of the wireless AP	x	✓
radio_number	Number of radios supported on the AP (for example, AP-205 supports two radios)	x	✓
dev_id	Flagged ID of a video device, according to the spectrum channel frequency	x	✓
dev_type	Device type: ● **0 = SPECTRUM_DEVICE_TYPE_UKNOWN**: Uknown device ● **1 = SPECTRUM_DEVICE_TYPE_WIFI**: Wi-Fi device ● **2 = SPECTRUM_DEVICE_TYPE_MICROWAVE**: Microwave device ● **3 = SPECTRUM_DEVICE_TYPE_BLUETOOTH**: Bluetooth device ● **4 = SPECTRUM_DEVICE_TYPE_GENERIC_FIXED_FREQ**: Generic fixed frequency device ● **5 = SPECTRUM_DEVICE_TYPE_CORDLESS_PHONE_FIXED_FREQ**: Cordless phone with fixed frequency ● **6 = SPECTRUM_DEVICE_TYPE_VIDEO_DEVICE_FIXED_FREQ**: Video device with fixed frequency ● **7 = SPECTRUM_DEVICE_TYPE_AUDIO_DEVICE_FIXED_FREQ**: Audio device with fixed frequency ● **8 = SPECTRUM_DEVICE_TYPE_GENERIC_FREQ_HOPPER**: Generic frequency hopper device ● **9 = SPECTRUM_DEVICE_TYPE_CORDLESS_PHONE_FREQ_HOPPER**: Cordless phone with frequency hopping ● **10 = SPECTRUM_DEVICE_TYPE_XBOX_FREQ_HOPPER**: Microsoft Xbox with frequency hopping ● **11 = SPECTRUM_DEVICE_TYPE_MICROWAVE_INVERTER**: Microwave device with inverter technology ● **12 = SPECTRUM_DEVICE_TYPE_CORDLESS_BASE_FREQ_HOPPER**:Cordless phone base unit with frequency hopping	x	✓
low_ch	Lowest channel that is affected by the device spectrum	x	✓
high_ch	Highest channel that is affected by the device spectrum	x	✓
signal	Signal strength of the device	x	✓
duty_cycle	Percent usage of each channel in the spectrum monitor radio's frequency band	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
center_freq	Center frequency of the device	x	✓
timestamp	Time that the device is discovered	x	✓
active_time	Activity duration for the device	x	✓

The Spectrum Info API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "spectrum_info"
Protobuf schema

message spectrum_info {
   optional mac_address ap_mac= 1;
   optional uint32 radio_number = 2;
   optional uint32 dev_id = 3;
   optional uint32 dev_type = 4;
   optional uint32 low_ch = 5;
   optional uint32 high_ch = 6;
   optional uint32 signal = 7;
   optional uint32 duty_cycle = 8;
   optional uint32 center_freq = 9;
   optional uint64 timestamp = 10;
   optional uint64 active_time = 11;
}
```

Access Point State

The Access Point State message returns (among other information) details on modem status when a modem connects to or disconnects from an IAP. This API is only available in IAP deployments.

The output for this message type displays the following information:

Table 77: *Access Point State API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_mac	MAC address of the AP	x	✓
serial_number	Serial number of the AP	x	✓
ap_name	Name of the AP	x	✓
model	AP model type	x	✓
mode	The AP's deployment mode string: ● Access ● Monitor ● Dedicated spectrum ● Hybrid spectrum	x	✓

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
ap_ip	IP address of the AP	x	✓
cpu_usage	Percentage of CPU usage by the AP	x	✓
memory_total	Total AP memory, in bytes	x	✓
memory_free	Free AP memory, in bytes	x	✓
is_master	Indicates if the IAP is an AP master. A value of **1** indicates that the IAP is an AP master. A value of **0** indicates that the IAP is not an AP master.	x	✓
uptime	Running time of the AP since the last bootup, in seconds	x	✓
mesh_mode	Mesh mode: ● 0 = No mesh mode ● 1 **=** Mesh Point ● 2 **=** Mesh Portal	x	✓
led_status	Status of LEDs on the AP: ● 0 = Each LED displays a different status ● 1 = All LEDs flash/blink	x	✓
ext_ssid_state	Enables or disables the extended SSID feature. If enabled, the mesh service is disabled and users can configure additional SSIDS. If disabled, the mesh service is enab led. ● 0 = disable ● 1 = enable	x	✓
service_tag	Unique service tag to identify the AP **NOTE:** For OEM Dell products	x	✓
modem_status	Modem status: ● **0 = UNPLUGIN**: Modem is unplugged or not supported ● **1 = PLUGIN**: Modem is plugged in	x	✓
current_uplink_inuse	Indicates which uplink type is currently being used: ● **1 = UPLINK_TYPE_ETH**: Ethernet ● **2 = UPLINK_TYPE_MESH**: WiFi mesh ● **3 = UPLINK_TYPE_STA**: WiFi station ● **4 = UPLINK_TYPE_3G**: 3G/4G modem	x	✓

The Access Point State API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "state_access_point"
Protobuf schema

message state_access_point {
   optional mac_address ap_mac = 1;
   optional string serial_number = 2;
   optional string ap_name = 3;
   optional string model = 4;
   optional string mode = 5;
   optional ip_address ap_ip = 6;
   optional uint32 cpu_usage = 7;
   optional uint32 memory_total 8;
   optional uint32 memory_free = 9;
   optional uint32 is_master = 10;
   optional uint32 uptime = 11;
   optional uint32 mesh_mode = 12;
   optional uint32 led_status = 13;
   optional uint32 ext_ssid_state = 16;
   optional string service_tag = 17;
   optional uint32 modem_status = 18;
   optional uint32 current_uplink_inuse = 19;
}
```

Modem Statistics

The Modem Statistics message returns modem statistics data from a Virtual Controller (VC) if a modem is plugged in and the AP uplink type is set to **3G/4G modem**. Refer to the Access Point State to view the modem status (**modem_status**) and current uplink type (**current_uplink_inuse**). This API is only availble in IAP deployments.

The output for this message type displays the following information:

Table 78: *Modem Statistics API Message Parameters*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the AP	x	✓
tx_data_bytes	Amount of transmitted data traffic, in bytes	x	✓
rx_data_bytes	Amount of received data traffic, in bytes	x	✓

The Modem Statistics API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "stats_modem"
Protobuf schema

message stats_modem {
   optional mac_address iap_mac = 1;
   optional uint64 tx_data_bytes = 2;
   optional uint64 rx_data_bytes = 3;
}
```

Uplink Info

The Uplink Info message returns general information about uplink ports and tunnels configured on Aruba Branch Office Controllers or IAPs. This API is available in BOC (Branch Office Controller) and IAP deployments.

The output for this message type displays the following information, but not all fields are present in both BOC and IAP deployments. Refer to Table 79 and the example outputs to determine which fields are present in each mode.

Table 79: *Uplink Info API Message Parameters*

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
device_mac	MAC address of the Branch Office Controller or IAP, which uniquely identifies the device address	x	✓	✓
link_id	Unique identifier of the physical uplink port or tunnel: ● Uplink port IDs begin sequentially from 101. ● Tunnel IDs begin sequentially from 201. Each tunnel profile contains a primary and backup tunnel: ● 0 = Primary tunnel ● 1 = Backup tunnel **NOTE:** Primary and backup tunnels are only available in IAP deployments but are not populated in 7005 controllers. In IAP deployments, the **link_id** and **tunnel_name** identify the VPN tunnel.	x	✓	✓
desc	Peer name (VPN server domain or IP) or VLAN description	x	✓	✓
link_status	Uplink status: ● 0 = VPN tunnel is down ● 1 = VPN tunnel is up	x	✓	✓
vlan_id	VLAN ID of the uplink port	x	x	✓
port_tunnel_desc	Description of the port or tunnel	x	x	✓
link_prio	Uplink priority	x	x	✓
wan_status	Port status	x	x	✓
crypto_type	Crypto type: ● 0 = Certificate ● 1 = Pre-shared key (PSK)	x	✓	x
tunnel_active_status	Indicates if the tunnel is active: ● 0 = Current tunnel is not active ● 1 = Current tunnel is active **NOTE:** The primary and backup tunnel can be UP without being active.	x	✓	x
tunnel_uptime	Tunnel uptime	x	✓	x
peer_tunnel_ip	Internal IP of the VPN server	x	✓	x
tunnel_ip	Internal IP of the IAP	x	✓	x
tunnel_name	Name of the VPN tunnel profile: ● **"default"**: Aruba VPN type ● **User-Defined Name**: Virtual Interface Gateway (VIG) special VPN type	x	✓	x

The Uplink Info API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "uplink_info"
```

```
Protobuf schema

message uplink_info {
   ooptional mac_address device_mac = 1;
   optional uint64 link_id = 2;
   optional string desc = 3;
   optional uint32 link_status = 4;
   optional uint32 vlan_id = 5;
   optional string port_tunnel_desc = 6;
   optional uint32 link_prio = 7;
   optional uint32 wan_status = 8;
   optional uint32 crypto_type = 9;
   optional uint32 tunnel_active_status = 10;
   optional uint32 tunnel_uptime = 11;
   optional ip_address peer_tunnel_ip = 12;
   optional ip_address tunnel_ip = 13;
   optional string tunnel_name = 14;
}
```

Uplink Statistics

The Uplink Statistics message returns performance information for uplink ports and tunnels configured on Aruba Branch Office Controllers or IAPs. This API is available in BOC (Branch Office Controller) and IAP deployments.

The output for this message type displays the following information, but not all fields are present in both BOC and IAP deployments. Refer to Table 80 and the example outputs to determine which fields are present in each mode.

Table 80: *Uplink Statistics API Message Parameters*

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
device_mac	MAC address of the Branch Office Controller or IAP, which uniquely identifies the device address	x	✓	✓
link_id	Unique identifier of the physical uplink port or tunnel: ● Uplink port IDs begin sequentially from 101. ● Tunnel IDs begin sequentially from 201. Each tunnel profile contains a primary and backup tunnel: ● 0 = Primary tunnel ● 1 = Backup tunnel **NOTE:** Primary and backup tunnels are only available in IAP deployments but are not populated in 7005 controllers. In IAP deployments, the **link_id** and **tunnel_name** identify the VPN tunnel.	x	✓	✓
rx_pkts	Number of packets received through an uplink	x	✓	✓
tx_pkts	Number of packets transmitted through an uplink	x	✓	✓
rx_bytes	Number of bytes received through an uplink	x	✓	✓
tx_bytes	Number of bytes transmitted through an uplink	x	✓	✓

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
vlan_id	VLAN ID of the uplink port	x	x	✓
tunnel_name	Name of the VPN tunnel profile: ● **"default"**: Aruba VPN type ● **User-Defined Name**: Virtual Interface Gateway (VIG) special VPN type	x	✓	x

The Uplink Statistics API output is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “stats_uplink”
Protobuf schema

message stats_uplink {
   optional mac_address device_mac = 1;
   optional uint64 link_id = 2;
   optional uint64 rx_pkts = 3;
   optional uint64 tx_pkts = 4;
   optional uint64 rx_bytes = 5;
   optional uint64 tx_bytes = 6;
   optional uint32 vlan_id = 7;
   optional string tunnel_name = 8;
}
```

Uplink WAN Compression

The Uplink WAN Compression message returns information on uplink data compression, which compresses packets to increase data-tranfser efficiency. This API is only available in BOC (Branch Office Controller) deployments.

The output for this message type displays the following information:

Table 81: *Uplink WAN Compression API Message Parameters*

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
boc_mac	MAC address of the Branch Office Controller, which uniquely identifies the branch device address	x	x	✓
link_id	Unique identifier of the physical uplink port or tunnel: ● Uplink port IDs begin sequentially from 101. ● Tunnel IDs begin sequentially from 201. Each tunnel profile contains a primary and backup tunnel: ● 0 = Primary tunnel ● 1 = Backup tunnel **NOTE:** Primary and backup tunnels are only available in IAP deployments but are not populated in 7005 controllers. In IAP deployments, the **link_id** and **tunnel_ name** identify the VPN tunnel.	x	x	✓

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
tot_comp_bytes	Total uplink bytes after compression	x	x	✓
tot_uncomp_bytes	Total uplink bytes before compression	x	x	✓
tot_comp_savings	Total savings from compression	x	x	✓

The Uplink WAN Compression API output is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “wan_comp_uplink”
Protobuf schema

message wan_comp_uplink {
   optional mac_address boc_mac = 1;
   optional uint64 link_id = 2;
   optional uint64 tot_comp_bytes = 3;
   optional uint64 tot_uncomp_bytes = 4;
   optional uint64 tot_comp_savings = 5;
}
```

Uplink IP Probe Statistics

The Uplink IP Probe Statistics message returns monitoring information for traffic sent through an uplink port or tunnel. In order to retrieve the uplink IP probe statistics, you must configure the probe IP address (probe-ip x.x.x.x) for the Virtual Interface Gateway (VIG) tunnel profile. This API is available in BOC (Branch Office Controller) and IAP deployments.

NOTE

The Uplink IP Probe Statistics message is sent every 300 seconds for IAP deployments and every 60 seconds for BOC deployments.

The output for this message type displays the following information:

Table 82: *Uplink IP Probe Statistics API Message Parameters*

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
device_mac	MAC address of the Branch Office Controller or IAP, which uniquely identifies the device address	x	✓	✓
link_id	Unique identifier of the physical uplink port or tunnel: ● Uplink port IDs begin sequentially from 101. ● Tunnel IDs begin sequentially from 201. Each tunnel profile contains a primary and backup tunnel: ● 0 = Primary tunnel ● 1 = Backup tunnel **NOTE:** Primary and backup tunnels are only available in IAP deployments but are not populated in 7005 controllers. In IAP deployments, the **link_id** and **tunnel_name** identify the VPN tunnel.	x	✓	✓

Output Parameter	Definition	Deployment Availability		
		Controller	IAP	BoC
probe_status	Probe status: ● 1 = Success (≥0% packet loss) ● 2 = Failure (100% packet loss)	x	✓	✓
ip_probe_pkt_loss_pct	Packet loss percentage, based on the number of failures out of the total number of attempts	x	✓	✓
probe_ip_addr	IP address to which the probe is sent	x	✓	✓
vlan_id	VLAN ID of the uplink port or tunnel used by the probe	x	x	✓
avg_rtt	Average round-trip delay time	x	x	✓
max_rtt	Maximum round-trip delay time	x	x	✓
min_rtt	Minimum round-trip delay time	x	x	✓
avg_jitter	Average traffic jitter	x	x	✓
max_jitter	Maximum traffic jitter	x	x	✓
min_jitter	Minimum traffic jitter	x	x	✓
Mos_quality	Mean Opinion Score quality metric	x	x	✓
tunnel_name	Name of the VPN tunnel profile: ● **"default"**: Aruba VPN type ● **User-Defined Name**: Virtual Interface Gateway (VIG) special VPN type **NOTE:** IAP deployments only support VIG tunnel profiles.	x	✓	x

The Uplink IP Probe Statistics API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "stats_ip_probe_uplink"
Protobuf schema

message stats_ip_probe_uplink {
   optional mac_address device_mac = 1;
   optional uint64 link_id = 2;
   optional ip_address probe_ip_addr = 3;
   optional uint32 vlan_id = 4;
   optional uint32 avg_rtt = 5;
   optional uint32 max_rtt = 6;
   optional uint32 min_rtt = 7;
   optional uint32 avg_jitter = 8;
   optional uint32 max_jitter = 9;
   optional uint32 min_jitter = 10;
   optional uint32 mos_quality = 11;
   optional uint32 probe_status = 12;
   optional uint32 ip_probe_pkt_loss_pct = 13;
   optional string tunnel_name = 14;
}
```

IAP Web Category Summary

The IAP Web Category Summary message returns IAP performance information based on the type of websites clients visit when they are connected to the network. This API is only available in IAP deployments.

The output for this message type displays the following information:

Web Category Summary

The **summary_webcat_iap** message filter returns performance information based on the website category.

Table 83: *IAP Web Category Summary API Message Parameters - summary_webcat_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
webcat_id	ID number specifying the website category	x	✓
iap_bytes_info	Bytes received and transmitted, with the enforcement status	x	✓

IAP Bytes Info

The **iap_bytes_info** message filter returns information on transmitted and received bytes, including the enforcement status.

Table 84: *IAP Web Category Summary API Message Parameters - iap_bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
session_flags	Session flag indicating the IAP enforcement status: ● **ENF_PERMIT**: Permit ● **ENF_DENY**: Deny	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 85: *IAP Web Category Summary API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP Web Category Summary API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "summary_webcat_iap"
Protobuf schema

message summary_webcat_iap {
   optional mac_address iap_mac = 1;
   optional uint32 webcat_id = 2;
   optional iap_bytes_info bytes_info = 3;
      optional iap_enforcement_status session_flags = 1;
      optional bytes_info bytes_info = 2;
         optional uint64 tx_bytes = 1;
         optional uint64 rx_bytes = 2;
}
```

IAP Application ID Summary

The IAP Application ID Summary message returns IAP performance information based on the type of applications being used by clients. This API is only available in IAP deployments.

The output for this message type displays the following information:

Application ID Summary

The **summary_appid_iap** message filter returns performance information based on the application type.

Table 86: *IAP Application ID Summary API Message Parameters - summary_appid_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
app_id	ID number identifying the application	x	✓
iap_bytes_info	Bytes received and transmitted, with the enforcement status	x	✓

IAP Bytes Info

The **iap_bytes_info** message filter returns information on transmitted and received bytes, including the enforcement status.

Table 87: *IAP Application ID Summary API Message Parameters - iap_bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
session_flags	Session flag indicating the application enforcement status: • **ENF_PERMIT**: Permit • **ENF_DENY**: Deny	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 88: *IAP Application ID Summary API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP Application ID Summary API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "summary_appid_iap"
Protobuf schema

message summary_appid_iap {
   optional mac_address iap_mac = 1;
   optional uint32 app_id = 2;
   optional iap_bytes_info bytes_info 3;
      optional iap_enforcement_status session_flags = 1;
      optional bytes_info bytes_info = 2;
         optional uint64 tx_bytes = 1;
         optional uint64 rx_bytes = 2;
}
```

IAP Web Reputation Summary

The IAP Web Reputation Summary message returns IAP performance information based on the reputation of websites clients visit when they are connected to the network. This API is only available in IAP deployments.

The output for this message type displays the following information:

Web Reputation Summary

The **summary_webrep_iap** message filter returns performance information based on the reputation of a website.

Table 89: *IAP Web Reputation Summary API Message Parameters - summary_webrep_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
webrep_id	Reputation of the website based on security risks (for example, malware and phishing)	x	✓
iap_bytes_info	Bytes received and transmitted, with the enforcement status	x	✓

IAP Bytes Info

The **iap_bytes_info** message filter returns information on transmitted and received bytes, including the enforcement status.

Table 90: *IAP Web Reputation Summary API Message Parameters - iap_bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
session_flags	Session flag indicating the IAP enforcement status: ● **ENF_PERMIT**: Permit ● **ENF_DENY**: Deny	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 91: *IAP Web Reputation Summary API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP Web Reputation Summary API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "summary_webrep_iap"
Protobuf schema

message summary_webrep_iap {
   optional mac_address iap_mac = 1;
   optional uint32 webrep_id = 2;
   optional iap_bytes_info bytes_info 3;
      optional iap_enforcement_status session_flags = 1;
      optional bytes_info bytes_info = 2;
         optional uint64 tx_bytes = 1;
         optional uint64 rx_bytes = 2;
}
```

IAP Role Statistics

The IAP Role Statistics message returns IAP performance information based on the user role of a client associated with the IAP. This API is only available in IAP deployments.

The output for this message type displays the following information:

Role Statistics

The **stats_role_iap** message filter returns performance information based on the client user role.

Table 92: *IAP Role Statistics API Message Parameters - stats_role_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
user_role	User role currently assigned to the client	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 93: *IAP Role Statistics API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP Role Statistics API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "stats_role_iap"
Protobuf schema

message stats_role_iap {
   optional mac_address iap_mac = 1;
   optional bytes user_role = 2;
   optional bytes_info bytes_info = 3;
      optional uint64 tx_bytes = 1;
      optional uint64 rx_bytes = 2;
}
```

IAP VLAN Statistics

The IAP VLAN Statistics message returns IAP performance information based on the VLAN to which the IAP is assigned. This API is only available in IAP deployments.

The output for this message type displays the following information:

VLAN Statistics

The **stats_vlan_iap** message filter returns performance information based on the assigned VLAN.

Table 94: *IAP VLAN Statistics API Message Parameters - stats_vlan_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
vlan	VLAN ID	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 95: *IAP VLAN Statistics API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP VLAN Statistics API output is as follows:

```
ØMQ endpoint    “tcp://localhost:7779”
ØMQ message filter      “stats_vlan_iap”
Protobuf schema

message stats_vlan_iap {
   optional mac_address iap_mac = 1;
   optional uint32 vlan = 2;
   optional bytes_info bytes_info = 3;
      optional uint64 tx_bytes = 1;
      optional uint64 rx_bytes = 2;
}
```

IAP SSID Statistics

The IAP SSID Statistics message returns IAP performance information based on the SSID of the IAP. This API is only available in IAP deployments.

The output for this message type displays the following information:

SSID Statistics

The **stats_ssid_iap** message filter returns performance information based on the SSID of the IAP.

Table 96: *IAP SSID Statistics API Message Parameters - stats_ssid_iap*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
iap_mac	MAC address of the IAP	x	✓
essid	ESSID of the IAP	x	✓
bytes_info	Bytes received and transmitted	x	✓

Bytes Info

The **bytes_info** message filter returns information on transmited and received bytes.

Table 97: *IAP SSID Statistics API Message Parameters - bytes_info*

Output Parameter	Definition	Deployment Availability	
		Controller	IAP
tx_bytes	Number of transmitted bytes	x	✓
rx_bytes	Number of received bytes	x	✓

The IAP SSID Statistics API output is as follows:

```
ØMQ endpoint    "tcp://localhost:7779"
ØMQ message filter      "stats_ssid_iap"
Protobuf schema

message stats_ssid_iap {
   optional mac_address iap_mac = 1;
   optional bytes essid = 2;
   optional bytes_info bytes_info = 3;
      optional uint64 tx_bytes = 1;
      optional uint64 rx_bytes = 2;
}
```