

Analytics and Location Engine 2.0

Author:
Syed Ahmed

Contributors:
Anupam Wadhawan
Manjunath Mahishi



Copyright Information

© Copyright 2016 Hewlett Packard Enterprise Development LP

Open Source Code

This product includes code licensed under the GNU General Public License, the GNU Lesser General Public License, and/or certain other open source licenses. A complete machine-readable copy of the source code corresponding to such code is available upon request. This offer is valid to anyone in receipt of this information and shall expire three years following the date of the final distribution of this product version by Hewlett-Packard Company. To obtain such source code, send a check or money order in the amount of US \$10.00 to:

Hewlett-Packard Company

Attn: General Counsel

3000 Hanover Street

Palo Alto, CA 94304

USA

Please specify the product and version for which you are requesting source code. You may also request a copy of this source code free of charge at dl-gplquery@arubanetworks.com.

Contents	3
Figures	6
Acronyms	8
Introduction	10
ALE Details	12
Analytics and Location Engine	12
Key Value Proposition of this Solution	13
What ALE Provides	13
ALE Advantages over Traditional RTLS Engines	15
ALE and Privacy	15
ALE and Analytics Partners	16
ALE Historical Data	17
ALE Scale Limits and Hardware Requirements	17
ALE Licensing	17
ALE Internal Workflow	18
ALE 2.0 New Features	19
ALE in Aruba Controller-based WLAN	20
Communication Topology	20
Communication Workflow	20
AMON or RTLS?	21
Ports Used for ALE and Controller Communication	22
ALE in Aruba Instant-based WLAN	23
Communication Topology	23
Communication Workflow	24
Ports Used for ALE and IAP Communication	24
Bandwidth Considerations	24
AirWave and ALE	25
Ports Used for ALE and AirWave Communication	25
ALE Design Options for Controllers	26
AP Placement and Design	30
AP Count	30

Air Monitors	30
AP Placement	31
Ideal AP Placement	31
Perimeter APs	31
Hexagonal AP Placement	31
AP Mounting Height	32
AP Placement Recommendations	32
AP Separation	33
Minimum RSSI	33
Designing for a Location + Voice Ready WLAN	33
Example of a Non-voice-ready WLAN vs. Voice-ready WLAN	33
Installing and Configuring ALE	36
Installing ALE	36
Configuring the Deployment Modes on ALE	36
Context Mode	36
Context with Device Location (Estimated) Mode	36
Context with Device Location (Calibration) Mode	37
Choosing the Right ALE Mode	37
Best Practice Guidelines When Deploying ALE in Estimation Mode	37
Best Practice Guidelines When Deploying ALE in Calibration Mode	38
Troubleshooting	39
General ALE Issues	39
ALE UI Error Message: "No location data found for floor"	39
ALE UI Error: "Database regenerating failed"	42
No User-specific Information Visible in Presence/Location Feed	43
No Location Events Generated on ALE	46
Troubleshooting Location Issues For a Specific Client	51
Very Few Location Events on ALE Compared to Device Traffic on the WLAN	53
No Presence/Location Events for Unassociated devices on the WLAN	54
ALE-AirWave Issues	58
ALE UI Error: "Error Fetching Sites from AirWave"	58
Client Devices Being Incorrectly Located on the ALE Map	61
Troubleshooting AP Placement Inconsistencies Resulting in Inadequate/Incorrect Location Events on ALE	68
WebSocket Connection Issues	69
ALE Cannot Initiate a Connection to the WebSocket Server	69

ALE Troubleshooting Logs	73
ALE Logs	73
ALE Logging Levels	74
WebSocket Logs	75
On the WebSocket Client (ALE)	75
On the WebSocket Server	75
Tech Support Logs	76
Appendix	77
ALE Feed Reader Reference Code (Java)	77
ALE Feed Reader Reference Code (C++)	77
IAP-ALE Bandwidth Calculator	77
Configuring and Testing a WebSocket Tunnel	77
Ports Used for ALE and WebSocket Server Communication	78
Configuration Requirements	78
Configuration Steps	78
WebSocket Server	79
Install Java	79
Install the WebSocket Software Package	79
Create a Server Certificate	80
Set the WebSocket Server's Properties	82
Make Firewall Exceptions	84
Start the WebSocket Service	85
WebSocket Client (ALE)	86
Verify WebSocket Server Reachability	86
Trust the WebSocket Server's Identity	87
Configure the WebSocket Entry On ALE	88
Verify the Connection	88
Test the Polling and Publish/Subscribe APIs	89
Polling APIs	90
Building API	90
Floor API	90
Station API	91
Publish/Subscribe APIs	92
Using the ALE Demonstrator Application	95

Figure 1 Disable Anonymization	16
Figure 2 ALE Internal Architecture	18
Figure 3 Data Flow Between ALE and Aruba Controller-based WLAN	20
Figure 4 Data Flow between ALE and an Aruba Instant-based WLAN	23
Figure 5 Design 1: All-Master Campus Network	27
Figure 6 Design 2: Master and Local network in which the Master Controller does not Terminate APs	28
Figure 7 Design 3: Controller-based Distributed Network with Secure Connection between the Remote Sites and the Datacenter	29
Figure 8 Ideal AP Placement = Center + Perimeter + Corner APs	31
Figure 9 Hexagonal AP Layout	32
Figure 10 Non-voice-ready WLAN vs. Voice-ready WLAN	34
Figure 11 One Frequency at a Time 2.4 GHz	35
Figure 12 One Frequency at a Time 5 GHz	35
Figure 13 Error Message: No location data found for floor	39
Figure 14 Generated PDB	40
Figure 15 Maintenance PDB	40
Figure 16 Database Successfully Generated	41
Figure 17 ALE Map Dashboard	41
Figure 18 Database regenerating failed	42
Figure 19 Anonymization Enabled	45
Figure 20 Analytics and Location Engines Configuration	48
Figure 21 Analytics & Location Engine and IP Address	49
Figure 22 Trace Level	52
Figure 23 No Data Observed for Unassociated Devices	54
Figure 24 No Unassociated Devices	55
Figure 25 includeUnassocSta Knob	55
Figure 26 Enable includeUnassocSta Knob	56
Figure 27 Unassociated Devices Present	56
Figure 28 ALE Maps Dashboard	57
Figure 29 Enabling AMON	58
Figure 30 Adding AirWave	59
Figure 31 Error Fetching Sites from AirWave	59
Figure 32 Enabling VisualRF Engine	60
Figure 33 Selecting AirWave Floors on ALE	61
Figure 34 ALE to AirWave Connection Successful	61
Figure 35 Dots Incorrectly Laid Out	62

<i>Figure 36 ALE Distance Metric</i>	63
<i>Figure 37 Use Metric Units</i>	63
<i>Figure 38 Set Enable VisualRF Engine to No</i>	64
<i>Figure 39 Set Enable VisualRF Engine to Yes</i>	64
<i>Figure 40 Delete AirWave Entry on ALE</i>	65
<i>Figure 41 Adding an AirWave Entry on ALE.</i>	65
<i>Figure 42 Specifying AirWave Credentials on ALE</i>	66
<i>Figure 43 Selecting Floors from AirWave</i>	66
<i>Figure 44 ALE to AirWave Connection Successful</i>	67
<i>Figure 45 Regenerating PDB</i>	67
<i>Figure 46 Database Successfully Generated</i>	68
<i>Figure 47 ALE Map</i>	68
<i>Figure 48 WebSocket Connection Failed</i>	69
<i>Figure 49 Remote Endpoint as FQDN on ALE</i>	72
<i>Figure 50 Remote Endpoint</i>	73
<i>Figure 51 Enabling TRACE Level on ALE Logs</i>	74
<i>Figure 52 ALE Tech Support Logs</i>	76
<i>Figure 53 WebSocket Connections</i>	78
<i>Figure 54 WebSocket Workflow</i>	78
<i>Figure 55 Download WebSocket Software</i>	79
<i>Figure 56 Remote End-Point</i>	88
<i>Figure 57 WebSocket Authentication</i>	91
<i>Figure 58 WebSocket Floor API</i>	92
<i>Figure 59 ALE Demonstrator</i>	95
<i>Figure 60 ALE Demonstrator - Current Settings</i>	96
<i>Figure 61 ALE Demonstrator - Map Information</i>	97
<i>Figure 62 ALE Demonstrator - Pick a Floor</i>	98
<i>Figure 63 ALE Demonstrator - Pick a target to track</i>	99

[Table 1](#) defines the acronyms used in this guide.

Table 1: Acronyms

Acronym	Definition
ALE	Analytics and Location Engine
AMON	Advanced Monitoring
AP	Access Point
API	Application Programming Interface
ARM	Adaptive Radio Management
CAP	Campus Access Point
FTP	File Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IAP	Instant Access Point
IP	Internet Protocol
MAC	Media Access Control
NAT	Network Address Translation
PAPI	Proprietary Access Protocol Interface
PDB	Positioning Database
PII	Personally Identifiable Information
RAP	Remote Access Point
REST	REpresentational State Transfer
RF	Radio Frequency
RSSI	Received Signal Strength Indication
RTLS	Real-Time Locating Systems
RTT	Round-Trip Time
SCP	Secure Copy
SNR	Signal-to-Noise Ratio
TDOA	Time Difference of Arrival

Table 1: Acronyms

Acronym	Definition
VAP	Virtual Access Point
VBR	Virtual Beacon Report
VC	Virtual Controller
WAN	Wide Area Network
WLAN	Wireless Local Area Network

Use this guide in conjunction with the ALE API Guide and ALE User Guide. If you are looking for detailed information on the different modes of configuration or the different APIs or topics available with ALE, please refer to the ALE API Guide and ALE User Guide.

The proliferation of e-commerce and mobile commerce has changed the shopping behavior of consumers. Smart phones and mobile apps allow customers in brick-and-mortar stores to perform a price check with online retailers and other competitors before making purchase decisions. So changes in how and where we buy, coupled with substantial investments in existing brick-and-mortar stores has raised the urgency of understanding new cross-channel buying behavior, brand loyalty, preferences, and emerging trends.

The importance of understanding customer behavior and customer engagement has brought about a new wave of services such as way finding, Loyalty Analytics, and geo-fencing. These applications and services that allow retailers to understand and engage with customers depend heavily upon the information and data available to data networks to which the mobile devices connect. As the primary conduits between users, applications, and the Internet, data networks are windows into the attributes and behavior of anyone coming into contact with them. Mining these networks for business intelligence about social, mobile, cloud, behavioral, and situational data about people, locations, activities, preferences, and associations should be a low hanging fruit.

But it is not. Why? Because these rich sources of data are typically inaccessible to analytics engines that could derive meaningful business intelligence from seemingly random network activity. These data sets, which are exceptionally valuable to business intelligence applications, are worthless to the networking infrastructure, which consequently discards them.

A case in point. When a Wi-Fi enabled device comes within range of a wireless network it sends a message, called a “probe request,” asking for services available on the network. Probe requests are discarded by most Wi-Fi networks if the client device does not connect (associate) to the network. And yet, these simple transactions from unassociated client devices can yield important, business-relevant information. Broadly classified as “presence” information, probe requests can show how many people are near your network, how long they dwell there, and, if they leave and return, the recentness, frequency, and timing of their visits.

Instead of discarding probe request and other context- and location-related information, it is essential to make them available to applications that can extract meaning and value. As a rule, the value of business intelligence is directly related to the infrastructure required to extract it. The higher the value, the more sophisticated the infrastructure that’s required to mine it.

The business intelligence derived from the Wi-Fi networks allows customers in verticals such as Retail, Hospitality, Transportation, etc. to monetize their network using analytics and location tracking applications.

The following are some common use cases of business intelligence using Wi-Fi analytics:

- Use dwell times and customer traffic insights (associated and un-associated clients) to feed into business intelligence and marketing decisions. For example, determine the "busy time" of the day for the store or a particular section and use this to make staffing decisions.
- Use traffic patterns to determine the impact of a particular promotional campaign.
- Workspace optimization: Enterprise customers who lease expensive real estate for their offices can optimize their workspace by understanding how different areas in the office are being utilized, based on data obtained from the network and other sources such as employee card readers, etc.
- Smart Energy Management: Venues can keep a check on energy consumption by monitoring Wi-Fi traffic patterns (in conjunction with other sources of data) on the floor and accordingly control HVAC systems.

- Public venues such as airports, shopping centers, and shopping malls can monetize the network by providing analytics services to their tenants.

This chapter includes the following topics:

- [Analytics and Location Engine on page 12](#)
- [Key Value Proposition of this Solution on page 13](#)
- [What ALE Provides on page 13](#)
- [ALE Advantages over Traditional RTLS Engines on page 15](#)
- [ALE and Privacy on page 15](#)
- [ALE and Analytics Partners on page 16](#)
- [ALE Historical Data on page 17](#)
- [ALE Scale Limits and Hardware Requirements on page 17](#)
- [ALE Licensing on page 17](#)
- [ALE Internal Workflow on page 18](#)
- [ALE 2.0 New Features on page 19](#)

Analytics and Location Engine

The wireless network has a wealth of information about unassociated and associated devices. This rich information should be correlated and presented to applications and services that require it to construct meaningful business intelligence data. To achieve this, Aruba introduced the Analytics and Location Engine (ALE). ALE is designed to gather information from the network, process it and share it through simple and standard APIs. ALE computes locations for every client on the network in near real time and gives visibility into everything the wireless network knows. In general, Wi-Fi based locationing from ALE lends itself to use cases where traffic trends and patterns can be analyzed over a period of time. ALE provides industry standard APIs that include data on associated as well as unassociated clients, such as:

- Media Access Control (MAC) address
- Current location
- Historical data* for locations, applications, destinations, etc.
- Client User name and role
- Internet Protocol (IP) address
- Device type
- Geofence entry and exit events
- Application firewall data, showing the destinations and applications used by associated devices

This information allows customers and partner services and applications to mine the wealth of information about the people on their premises and extract useful business intelligence.

Providing visibility into the network helps in extracting valuable business intelligence but at the same time it introduces the controversial realm of privacy. To address the concerns around privacy, ALE supports data Anonymization. For more information on data anonymization, see [ALE and Privacy on page 15](#).

* See [ALE Historical Data on page 17](#) to understand the scope of historical data retention on ALE.

Key Value Proposition of this Solution

The following are the key value propositions of the ALE solution:

- **Ease of integration** - ALE provides a single point of integration for 3rd party analytics services for both Aruba Controller and Aruba Instant based platforms.
- **ALE addresses customer privacy concerns with data anonymization** - With this, the same useful analytics data is available without associating a mobile device to a user.
- **ALE provides more context if required** - ALE not only sends device MAC and RSSI details, but it can also provide information like user role, device type, web sites visited, etc. when on the network.
- **Open analytics ecosystem** - ALE Northbound Publish/Subscribe API is based on industry standard 0MQ for reliable transport and uses binary Google Protobuf encoding and this lets the customer choose from any of the available analytics services or integrate with internally developed apps.
- **Tight integration with Aruba Wireless Local Area Network (WLAN)** - A tight integration with Aruba specific features like Virtual Beacon Reports (VBR) of ClientMatch (ARM 3.0) provide better location accuracy than other solutions.
- **Location accuracy and scale** - Based on the configuration mode, ALE provides better location accuracy and scale than other Wi-Fi based solutions.

What ALE Provides

Every associated and unassociated client information that an Aruba WLAN knows about is received by ALE. With this information ALE performs two primary functions:

1. Calculates locations of associated and unassociated clients.
2. Stores user specific information such as location, user, and application context in its database and present it to third party applications and services that require it.

This information is available on ALE through two north-bound Application Programming Interfaces (APIs):

1. **Polling API** (based on REpresentational State Transfer (REST)): This API provides the ability to query for specific information such as stations, location, destinations, and applications. This API is mostly recommended for bootstrapping an application when it first starts or for information that does not change very often and where the latency with knowing about changes is not as important.
2. **Publish/Subscribe API** (based on Google Protobuf and ZeroMQ): This API allows the ability to subscribe to ALE for specific information. Once subscribed, ALE will start publishing messages on the subscribed topics, and sends them to the analytics application. This API is recommended for any application that wants near real-time updates for all the devices on the network.

Once the ALE in a data center is configured to talk to a partner application, ALE will start publishing messages on the subscribed topic to appropriate subscribers. The partner application can simultaneously integrate with ALE using the Polling API. However, note that the Polling API is used to query for historical information or some specific information. For example, assume that the link between ALE and the cloud application is down for a few minutes. When the link is up again, the Publish/Subscribe API will only send recent information as ALE learns about it but the application can use the Polling API to query for data that ALE collected when the link was down.

In general, applications and services use the Polling API to get the current snapshot of the network, then use the Publish/Subscribe API to learn about changes as they happen.

The Polling APIs supported by ALE are:

- Access Point API
- Virtual Access Point API
- Stations API

- Presence API
- Proximity API
- Campus API
- Building API
- Floor API
- Location API
- Application API
- Destination API
- Geo_Fence API
- System Information API
- WebCC Category API
- Topology API
- Controller API
- Cluster Info API

The Event messages published by the Publish/Subscribe API include:

- Access Point
- Application
- Campus
- Building
- Floor
- Destination
- Location
- Presence
- Proximity
- Radio
- Radio Statistics
- Radio Utilization/Histogram Statistics
- Station RSSI
- Security
- Station
- Station Statistics
- State Station
- Virtual Access Point (VAP)
- Virtual Access Point (VAP) Statistics
- WebCC
- Visibility Record
- Controller
- Cluster Info
- Uplink Bandwidth
- Geofence Notify

For more information on each of the above APIs and events, please see the ALE API Guide.



ALE hosts a native debugging tool called "feed-reader" that can subscribe to the ZMQ feed on ALE, either entirely or for select topics. Customers and partners can use the feed-reader code as a reference when building their own analytics applications. See the [Appendix on page 77](#).

To test the feed-reader tool, you can issue the following command on ALE to subscribe to the entire feed:

```
[root@ale ~]# /opt/ale/bin/feed-reader
```

Or you can issue the following command to subscribe to select topics:

```
[root@ale ~]# /opt/ale/bin/feed-reader -f location
```



There is an app called "ALE Demonstrator" (for Android) available on the Google Play Store that can be used as a reference in building your application. The ALE Demonstrator app communicates with an ALE server and displays the location of client devices on a floor plan. The source code for this Android app is available on GitHub. It shows how the different REST and Publish/Subscribe (ZMQ) APIs are used to pull both static data like AP placements and streaming data like location feeds. The source code for this app can be found here: <https://github.com/pthornycroft/ALE-Demonstrator>. See [Using the ALE Demonstrator Application on page 95](#) in the Appendix to configure the app.

ALE Advantages over Traditional RTLS Engines

Real-time locating systems (RTLS) are traditionally used for tracking physical assets. With RTLS, an Aruba WLAN sends information about the clients and devices it hears to an RTLS engine. The RTLS engine then calculates the client location. The Aruba WLAN also has the capability of sending unassociated client information to RTLS systems. This capability of the Aruba WLAN can be leveraged by partners to build location tracking applications.

However, this is not the recommended approach because the Received Signal Strength Indicator (RSSI) exported as RTLS has always been vendor dependent. Some vendors send signal strength, some send Signal-to-Noise Ratio (SNR) while others send average values. Moreover, the technology and metrics used for location calculation (RSSI, Time Difference of Arrival (TDOA), Round-trip time (RTT), etc.) is ever changing and is tightly coupled with AP capabilities. So to expect third party applications and services to keep up with this changing landscape is not desirable. Aruba solves this problem with ALE.

ALE is purpose built to be tightly integrated with the Aruba WLAN and provide location and other metrics required by third party applications and services using standards based APIs.

Some of the key benefits of ALE over other RTLS engines are:

- Secure communication between ALE and analytics partners
- Simplified deployments (works through Network Address Translation (NAT) when using a WebSocket tunnel between ALE and the analytics application)
- Only one or two connections to manage rather than a connection per AP
- Includes information beyond RSSI (location, user name, role, device type, etc.)
- Easier to integrate with than RTLS. ALE simplifies integration with partner applications and services by leveraging open source best practices and technologies like ZeroMQ and Google Protobuf.
- Supports multiple destinations
- Supports Remote Access Points (RAPs)

ALE and Privacy

A possible concern for customers who adopt analytics and other such applications is user privacy. In this age of data analytics, user privacy is one of the most widely debated topics and privacy restrictions vary from country to country. So the ability for a system to support data anonymization is becoming more critical to avoid any possible legal issues.

ALE supports data anonymization by performing a one-way hash of user MAC address and IP address. Other Personally Identifiable Information (PII) such as user name are nulled. Data anonymization is enabled by default on ALE, and the salting schedule can be set to be changed daily, weekly, or monthly.

However, there are often applications and services that might not work with data anonymization, and as such, anonymization might have to be disabled for such applications and services. Applications and services that require non-anonymous data from ALE can provide anonymization after processing the non-anonymized data from ALE. In this case, the data anonymization is provided by the third party application and service rather than ALE.

Data anonymization on ALE is enabled by default and can be disabled using the following configuration change on the ALE UI. As shown in [Figure 1](#), clear the Enable Anonymization checkbox and click **Apply**.

Figure 1 *Disable Anonymization*

The screenshot shows the Aruba ALE 2.0.0.1 (52070) configuration interface. The left sidebar has tabs for Monitoring, Configuration, Mode, Source, Options (which is selected), Admin, and Maintenance. The main panel has a 'General' section with several checkboxes: 'Enable Single Ap Location Calculation' (unchecked), 'Enable Geo Fences' (checked), and 'Enable Anonymization' (unchecked). Below these are dropdowns for 'Hash Generation Schedule' (set to 'NEVER') and 'Distance metric' (set to 'FEET'). There are also collapsed sections for 'NTP Server', 'WebSocket Tunnel', and 'Logging & Notifications'. At the bottom right are 'Cancel' and 'Apply' buttons.

ALE and Analytics Partners

ALE has a wealth of information about the users and activity on the network. However, ALE 2.0 does not provide any analytics-based reporting or visualization. For all analytics-based reporting, trend analysis and visualization, a third party analytics application that consumes ALE data is required.

Analytics providers distinguish themselves based on the value added services they provide on top of basic analytics. So based on the customer requirements on what is required and what is valuable, an appropriate analytics partner may be chosen by the customer. Analytics partners will integrate with ALE using a WebSocket connection.

For a complete list of analytics partners, check out the official [Aruba ecosystem partners](#) page.



When choosing an Analytics partner, the customer should carefully evaluate whether the analytics service provider meets the privacy definitions and requirements of the organization.

ALE Historical Data

ALE is not designed to store historical data. The main purpose of the Redis database is to help analytics applications learn about the wireless network during the bootstrap process via the Polling APIs. It is up to the analytics application to subscribe to the ALE feed and store relevant data accordingly.

Redis grows to a limited percent of free memory and then starts dropping. It is only fit for bootstrapping the analytics apps. Older data is deleted as new data arrives. ALE does not store records for any set time period. ALE does not store historical locations for a specific MAC client; only the last known location is retrieved by using the Location API. If historical locations are important to your use-case, they can be stored in an external database or file system by listening to and saving the publish/subscribe Location API.

ALE Scale Limits and Hardware Requirements

The current release of ALE (2.0) can support up to 100 controllers on a single ALE instance and up to 8 AirWave servers, with up to 1,000 floors per AirWave server and 2,000 floors total across all AirWave servers. When dealing with such a large scale, it is recommended to read directly from the VisualRF backup file instead of importing it over the network from AirWave.

[Table 2](#) provides recommended hardware requirements based on the size of the deployment.

Table 2: Recommended Hardware Requirements

Number of APs/Clients	CPU	RAM (default)	Hard Disk or Secondary Storage	Comments
Up to 1K/16K	8	16GB	120GB	Medium
2K/32K	16	32GB	160GB	Large

It is also important to consider the following factors when designing ALE for scale:

- The client count in the above table includes both associated and unassociated clients and this can vary based on the network environment.
- Above are suggested numbers, depending on the specific scenarios. For example, if there are a lot more clients resulting in more location calculations, then more CPU and Memory might be needed.
- When RTLS is configured as the source of data, the scale numbers decrease due to the higher rate of packets that need to be processed on ALE (since each AP is now sending data to ALE, instead of data being aggregated and sent from the controller).
- With "Context (Proximity)" mode, a higher scale can be achieved because there are no location computations that need to be performed on ALE.
- Depending on how much history needs to be stored, more memory may need to be added.

ALE 2.0 does not currently support clustering or redundancy. A future ALE release may be designed to accommodate clustering and redundancy.



ALE Licensing

ALE 2.0 is priced on a one-time, per-AP license. Three licensing schemes are available:

1. **Unlicensed:** Without licenses, ALE does not publish any ZMQ messages (needed for the Publish/Subscribe API).

2. **Evaluation license:** An evaluation license contains an AP-count and is valid for 90 days. After the evaluation license expires, ALE will not publish any ZMQ messages.
3. **Permanent license:** A permanent license contains an AP-count and does not expire.

 Please refer to the ALE User Guide for details on generating ALE licenses.

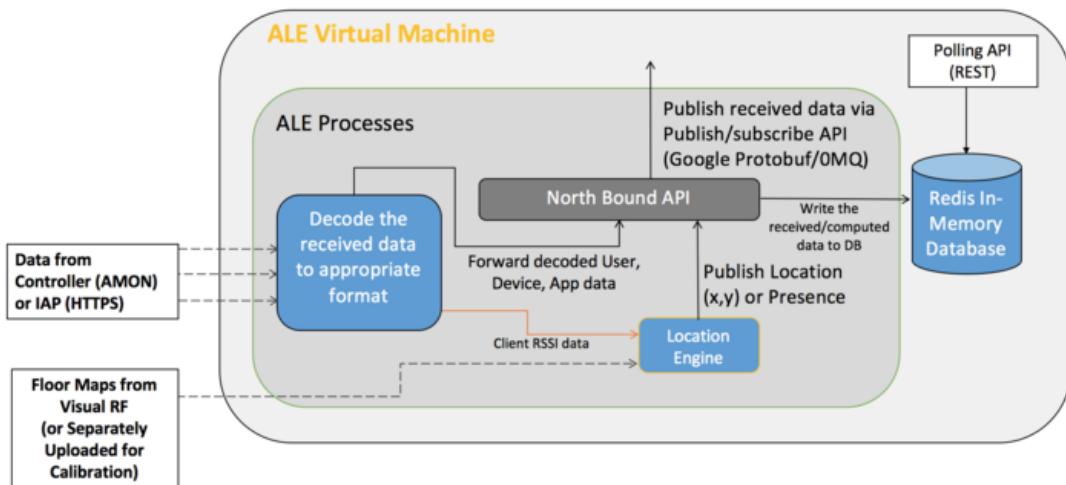
ALE Internal Workflow

To perform location calculation, ALE requires the followings two sets of data (assuming the Estimated mode of operation is selected on ALE. Please refer to the ALE User Guide for details on ALE modes):

- Associated and unassociated client information from Controller or Instant.
- AP location on a floor plan from Visual RF backup, either fetched from AirWave or uploaded on ALE.

[Figure 2](#) below shows the internal architecture of ALE.

Figure 2 ALE Internal Architecture



ALE Internal Workflow:

1. Receives and decodes feed from the controller (AMON/RTLS) or Instant AP (HTTPS).
2. The decoded information is sent to the North Bound API. The North Bound API publishes appropriate data to the subscribers as event messages.
3. Decoded client RSSI data is also sent to the location engine. The location engine has the floor maps (with appropriate AP info) from Visual RF (AirWave)
4. The location engine calculates the appropriate location information for each client and sends it to the North Bound API.
5. The North Bound API publishes the location information to the subscribers.
6. The North Bound API forwards all the data it receives (in steps 2 and 5) to the Redis database.
7. Application and services can query the Redis database using the Polling API.

 The data received by the North Bound API is published to the appropriate subscribers and forwarded to the Redis database as it is received, rather than at periodic intervals.

 Only one data source (controller or IAP) can be configured on ALE at a time.

ALE 2.0 New Features

If you are running ALE 1.3 in your network, below are some reasons why you should consider moving to ALE 2.0.

For those who are new to ALE and want to gain a better understanding of how ALE works, please refer to this ALE 2.0 Deployment and Troubleshooting Guide, the ALE 2.0 API Guide, and the ALE 2.0 User Guide.

- ALE 2.0 comes with a bunch of scale and security improvements over older ALE versions.

Scale:

- Up to 100 controllers can terminate on a single ALE instance
- Up to 8 AirWave servers supported per ALE
 - Up to 1,000 floors per AirWave server
 - Up to 2,000 floors across all AirWave servers

Security:

- All REST APIs now require authentication
- Support for mutual authentication for WebSocket connections
- A completely new look and feel that makes configuration and troubleshooting much easier.
- A host of new APIs introduced in ALE 2.0, allowing the customer to correlate WLAN data more easily. The complete list of APIs is available in the ALE 2.0 API Guide.
- Three modes available for configuration, depending on use case and desired level of location accuracy.
- ALE 2.0 optionally provides the capability to upload VisualRF back up files on ALE, which is helpful in very large deployments or in cases where ALE - AirWave communication is not feasible.
- A UI wizard now makes it easier to set up ALE after it is online. This is in contrast to 1.3 where most of the initial configuration was CLI-based.
- Stability

In an Aruba controller-based WLAN, both the Controller and the ALE must be configured to communicate with each other. For configuration details, refer to the ALE 2.0 User Guide.

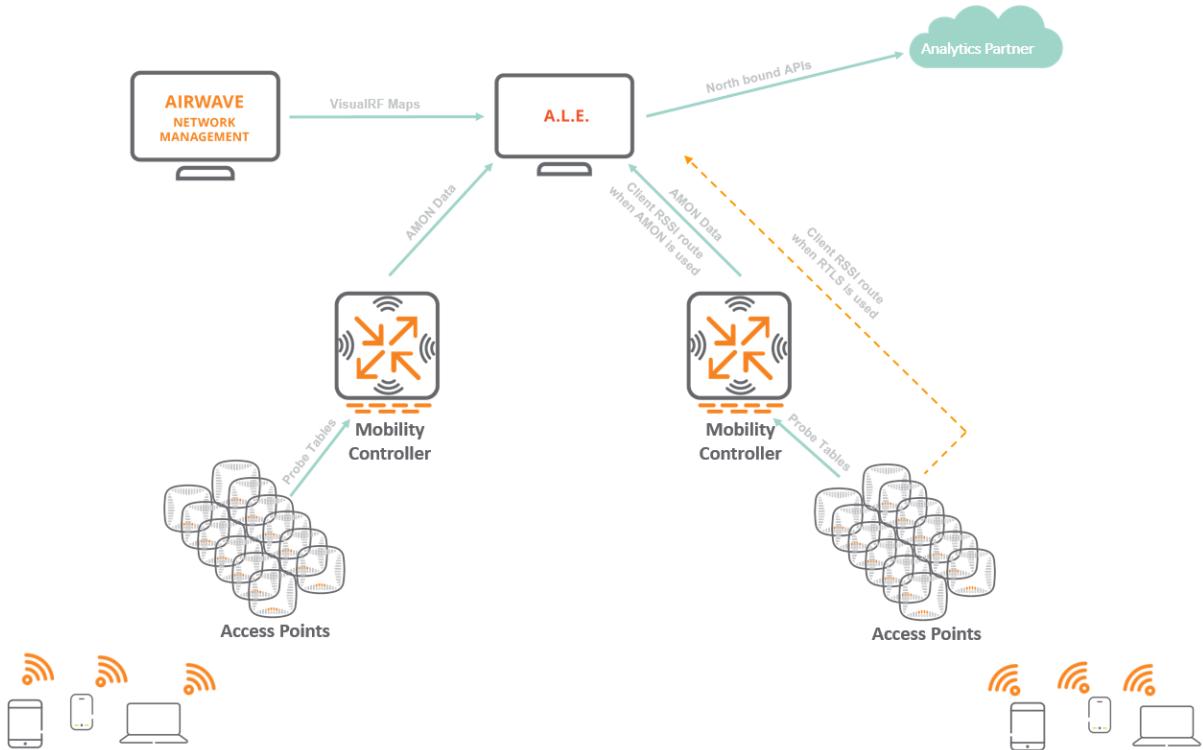
This section includes the following topics:

- [Communication Topology on page 20](#)
- [Communication Workflow on page 20](#)
- [AMON or RTLS? on page 21](#)
- [Ports Used for ALE and Controller Communication on page 22](#)

Communication Topology

The diagram below shows the data flow between ALE and an Aruba controller-based WLAN.

Figure 3 Data Flow Between ALE and Aruba Controller-based WLAN



Communication Workflow

In order to forward client RSSI to ALE, there are two types of feeds that you can configure on the controller:

1. **RTLS:** If RTLS is configured on the controller (under AP System Profile), client RSSI will be forwarded to ALE directly from each AP terminating on the controller. All other data (for example, users, applications, etc.) is still sent from the controller via Advanced Monitoring (AMON) (provided an ALE entry is configured on the controller).
2. **AMON:** If RTLS is not configured on the controller, not only user/application data but also client RSSI will be forwarded from the controller via AMON.

The communication workflow between ALE and the Aruba controller is as follows:

- After ALE and the controllers are configured to communicate with each other, ALE will perform an HTTP GET to learn about AMON data on the controller. This HTTP GET is a bootstrap operation that is performed when the ALE initially communicates to a controller.
- After the initial HTTP GET, all further user and application data, client RSSI data (if AMON is configured as the source of data on ALE), and AP-AP RSSI data received by ALE is an AMON push from the controller. This AMON push from the controller to ALE is a time based periodic push (currently, this is once around every 30 seconds).
- AMON data received by ALE includes the clients heard and the APs that hear them.
- ALE builds a Positioning DataBase (PDB) based on one of the following:
 1. AP placement and AP-AP RSSI information received from the WLAN, when ALE is deployed in Estimation mode.
 2. Fingerprinting data, when ALE is deployed in Calibration mode.
- With this information, ALE calculates the (X,Y) coordinates of clients on the floor, based on client RSSI and the PDB.
- The controller sends all other network changes such as addition and deletion of stations and APs, user roles, client session information, etc. as AMON data to ALE as soon as they occur.
- ALE publishes the data as soon as it is calculated, to the North bound API. The North bound API also forwards the data to the Redis DB.



Both Campus Access Points (CAPs) and RAPs are supported.



Remember that the communication between ALE and Controller is currently not secure. So when deploying ALE in an Aruba Controller-based WLAN, the network between controller and ALE must be trusted. The network between ALE and Instant Access Point (IAP) clusters, however, can be untrusted since IAPs use Hypertext Transfer Protocol Secure (HTTPS) to communicate with ALE.

AMON or RTLS?

In past AOS releases, the AMON feed contained RSSI only from probe request frames and not a lot of clients frequently send out probes. This is especially true for associated clients. This meant that the frequency at which ALE received client RSSI updates via AMON was considerably less, which introduced latency in location computations on ALE. Since an RTLS feed, in addition to probe RSSI, also takes into account RSSI from data frames, it was the recommended source of data for ALE.

In recent AOS releases (6.4.3 and above), however, the quality of RSSI in AMON has improved significantly. For example, RSSI from data frames is also included. RTLS has no known advantage over AMON now.

Ports Used for ALE and Controller Communication

[Table 3](#) lists the ports used for ALE and controller communication.

Table 3: Ports for ALE and Controller Communication

Communication	Port	Protocol	Notes
ALE > Controller	4343	HTTPS	During ALE bootstrap
Controller > ALE	8211	Proprietary Access Protocol Interface (PAPI)	Sending AMON feed to ALE
AP > ALE	Administrator-defined	RTLS	Only when the source of data feed on ALE is RTLS

ALE is supported with Instant 6.3.1.1-4.0.

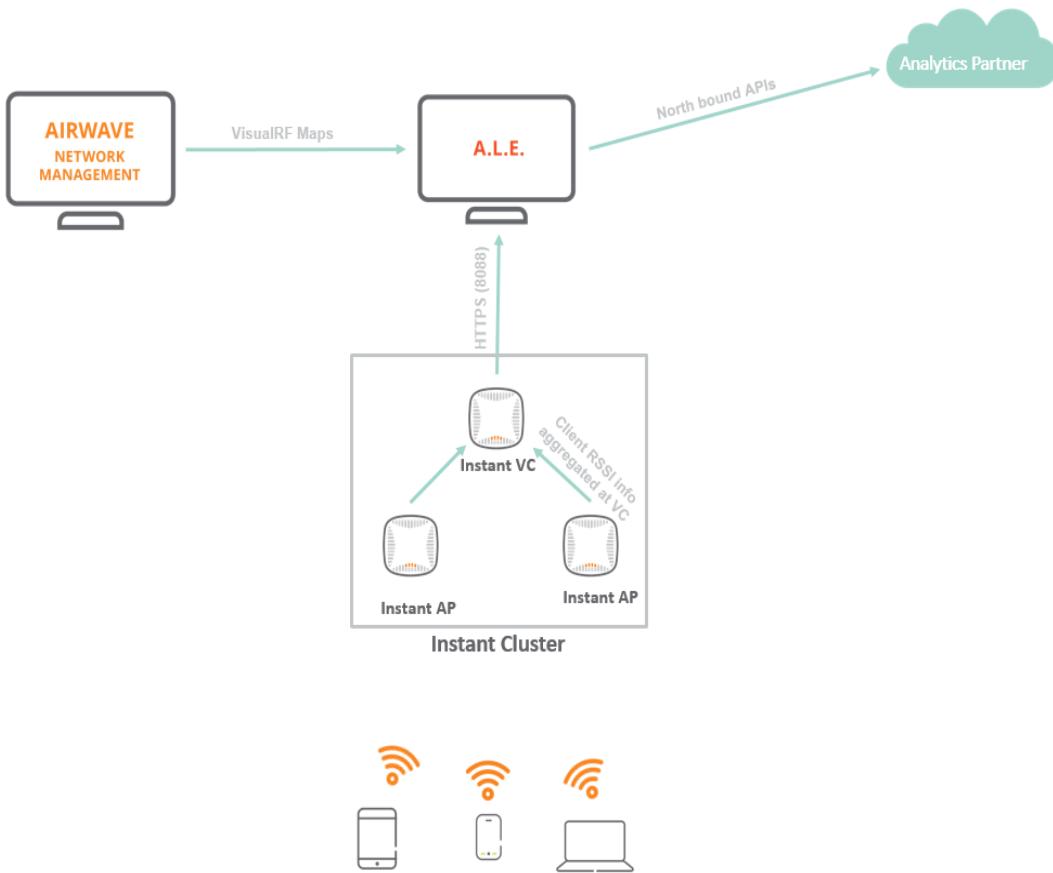
This section includes the following topics:

- [Communication Topology on page 23](#)
- [Communication Workflow on page 24](#)
- [Ports Used for ALE and IAP Communication on page 24](#)
- [Bandwidth Considerations on page 24](#)

Communication Topology

[Figure 4](#) below shows the data workflow between ALE and an Aruba Instant-based WLAN.

Figure 4 Data Flow between ALE and an Aruba Instant-based WLAN



Communication Workflow

The communication and workflow between ALE and the Aruba Instant cluster is as follows:

- When the Instant Virtual Controller (VC) and ALE are configured to communicate with each other, the VC sends a SYNC message to ALE with details about the Instant cluster (IAPs, radios, associated and unassociated clients, etc.).
- Every IAP in a cluster creates a probe table for clients in its vicinity and shares this information with the Instant VC.
- The Instant VC consolidates RSSI data from all such Instant APs in the cluster and shares it with ALE as part of the SYNC message.
- Any further changes on the WLAN (for example, addition and removal of APs and clients) are generally conveyed to ALE in the form of incremental updates.
- By default, updates are shared by the VC every 30 seconds. This interval is configurable.
- ALE compares incoming client RSSI against a PDB to compute client locations (x,y) on the floor.



RSSI data is aggregated on the VC and sent out to ALE. If AppRF is enabled, AppRF data is sent out to ALE in a distributed fashion (that is, from each individual IAP in the cluster).



The default frequency at which IAP sends updates to ALE is 30 seconds. Depending on the scale of the IAP deployment, it is recommended to set the interval at a higher frequency in order to achieve better location results. For very large deployments though, the default or a lower frequency interval is recommended.

Ports Used for ALE and IAP Communication

[Table 4](#) lists the port for Instant VC and ALE communication.

Table 4: Port for Instant VC and ALE Communication

Communication	Port	Protocol	Notes
Instant VC > ALE	8088	HTTPS	This port is configurable

Bandwidth Considerations

When deploying ALE in large distributed networks, it is critical to understand the amount of ALE-bound traffic being generated by an IAP cluster so that the Wide Area Network (WAN) pipe can be designed accordingly. The number of APs, associated and unassociated client devices, active client sessions, and update intervals - all these factors need to be considered when designing the WAN for capacity.

The IAP-ALE bandwidth calculator is a helpful tool that can be used to understand these bandwidth requirements. Contact Support or your local Aruba account team for access to this content.

In the 'Context + location (Estimated)' mode, AirWave is essential for ALE to obtain AP location and thus compute client locations. ALE contacts AirWave on initial bootstrap or when ALE services are restarted, and fetches floor plans from AirWave. The client and RSSI data received by ALE has a radio variable. So for location accuracy, it is important that the AP location on the Visual RF floor plan matches the physical AP location. Remember that data received from APs not present on the floor plan imported by ALE is not used for location calculation. It is important to update the maps if the physical location of APs are changed or if additional APs are added.



AirWave does not leverage the location calculation of ALE for client location on the Visual RF floor map.

For details on configuring AirWave and ALE, please refer to the ALE 2.0 User & API Guide.

With ALE 2.0, in scenarios where ALE-to-AirWave communication is not feasible or if the VisualRF files are very large, floor plans can be imported on ALE by downloading the VisualRF backup file from AirWave and permanently uploading it on ALE.

Ports Used for ALE and AirWave Communication

[Table 5](#) lists the port for ALE and AirWave communication.

Table 5: Port for ALE and AirWave Communication

Communication	Port	Protocol	Notes
ALE > AirWave	443	HTTPS	For fetching floor plan data from AirWave

The following are important to remember when designing an ALE network:

- Clustering or redundancy is not currently supported.
- Data segregation on ALE is currently not supported. ALE does not segregate data coming from multiple Aruba controllers as data coming from different customers or login accounts. The output generated by ALE is a single feed containing data from all controllers and IAP clusters. If data segregation is required, it is up to the analytics partner to parse and segregate data accordingly.
- The communication between ALE and the controller is not secure, so it is recommended that ALE and the controller not be connected through an untrusted WAN network. The network between ALE and IAP clusters, however, can be untrusted since IAPs use HTTPS to communicate with ALE.
- Multiple controllers can terminate on a single ALE instance (up to 100 controllers, with ALE 2.0).
- When running ALE in 'Context mode with location - Estimation' mode, accurate placement of APs on the Visual RF floor map is critical for location accuracy.
- Analytics partners who use ALE location data must scale their maps to the scale of the Visual RF maps to ensure location accuracy.
- If analytics partners need non-anonymized data, Anonymization must be disabled on ALE. Currently, Anonymization is a global setting and cannot be disabled for individual APIs, applications, or services.

Figure 5 Design 1: All-Master Campus Network

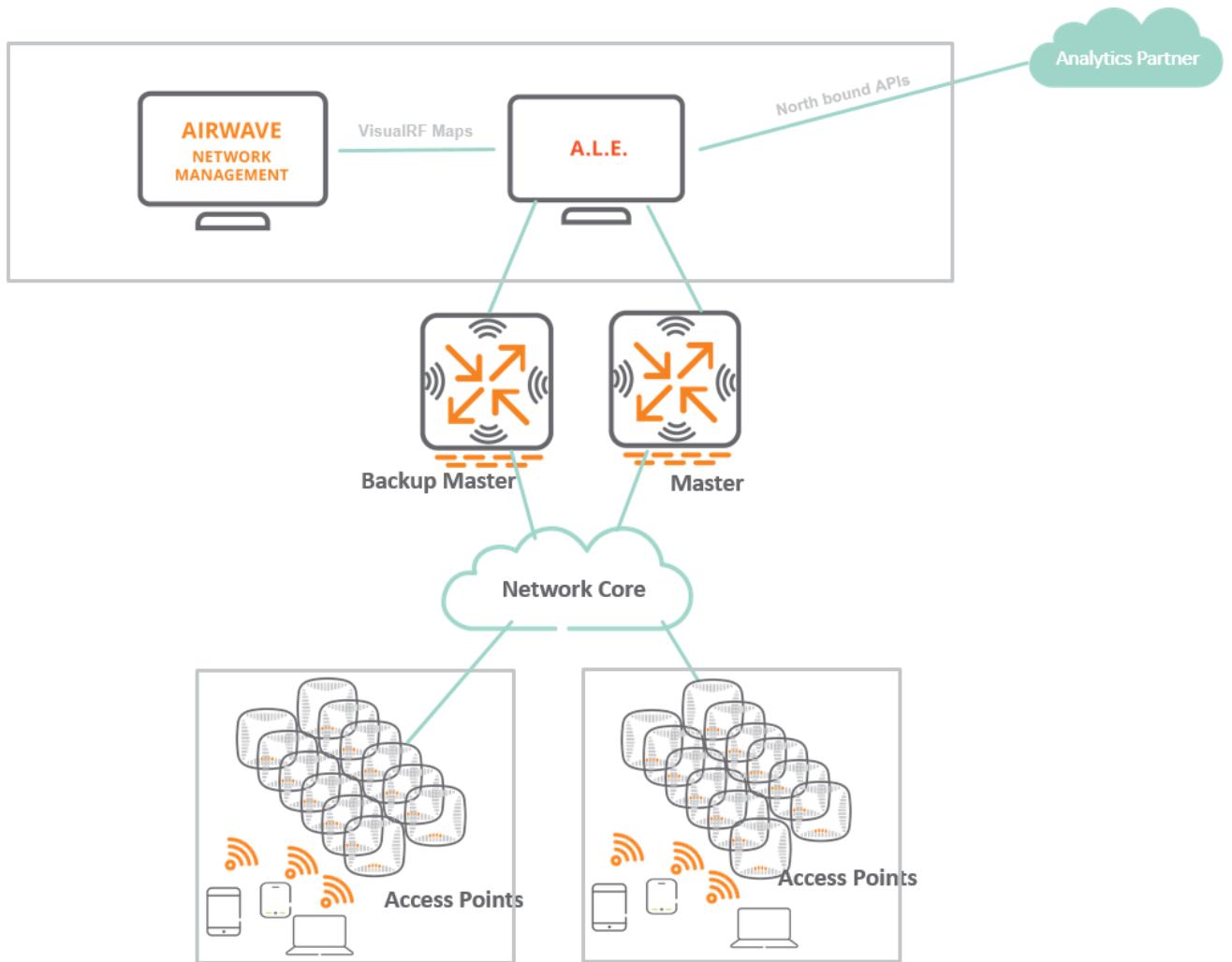


Figure 6 Design 2: Master and Local network in which the Master Controller does not Terminate APs

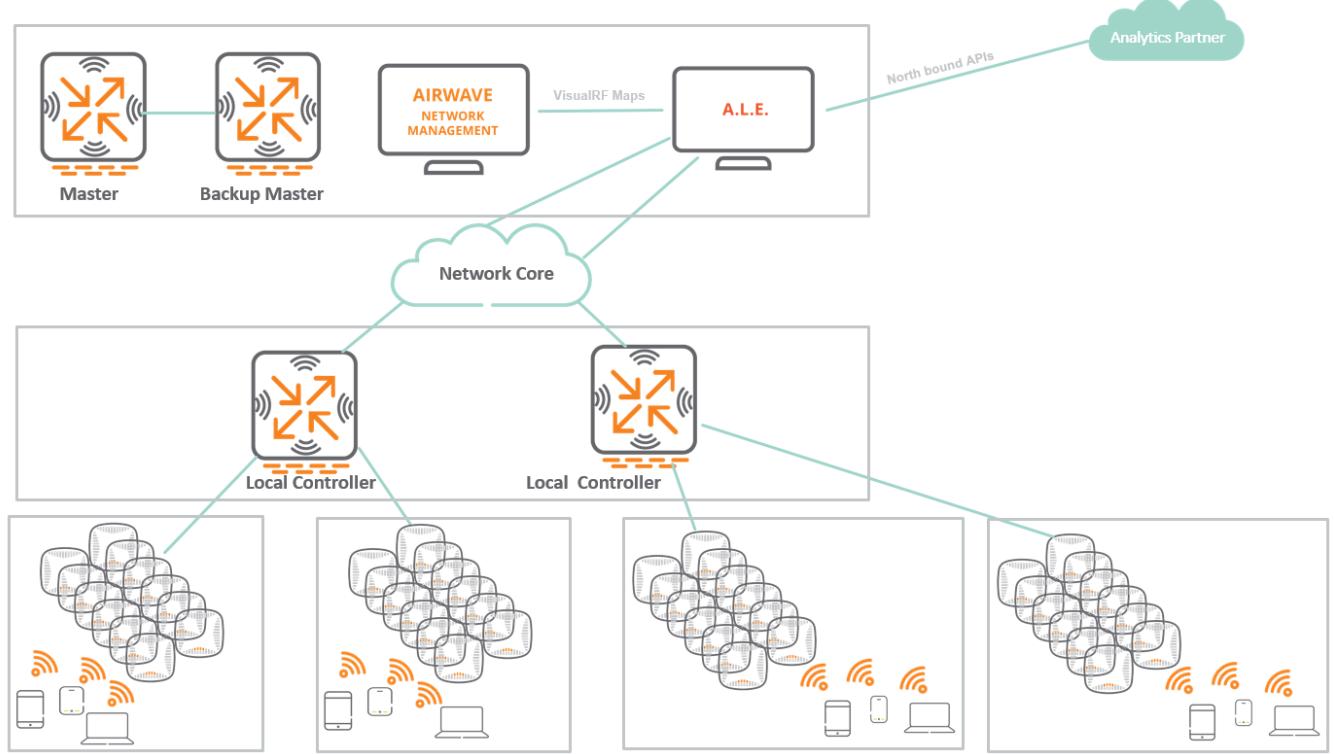
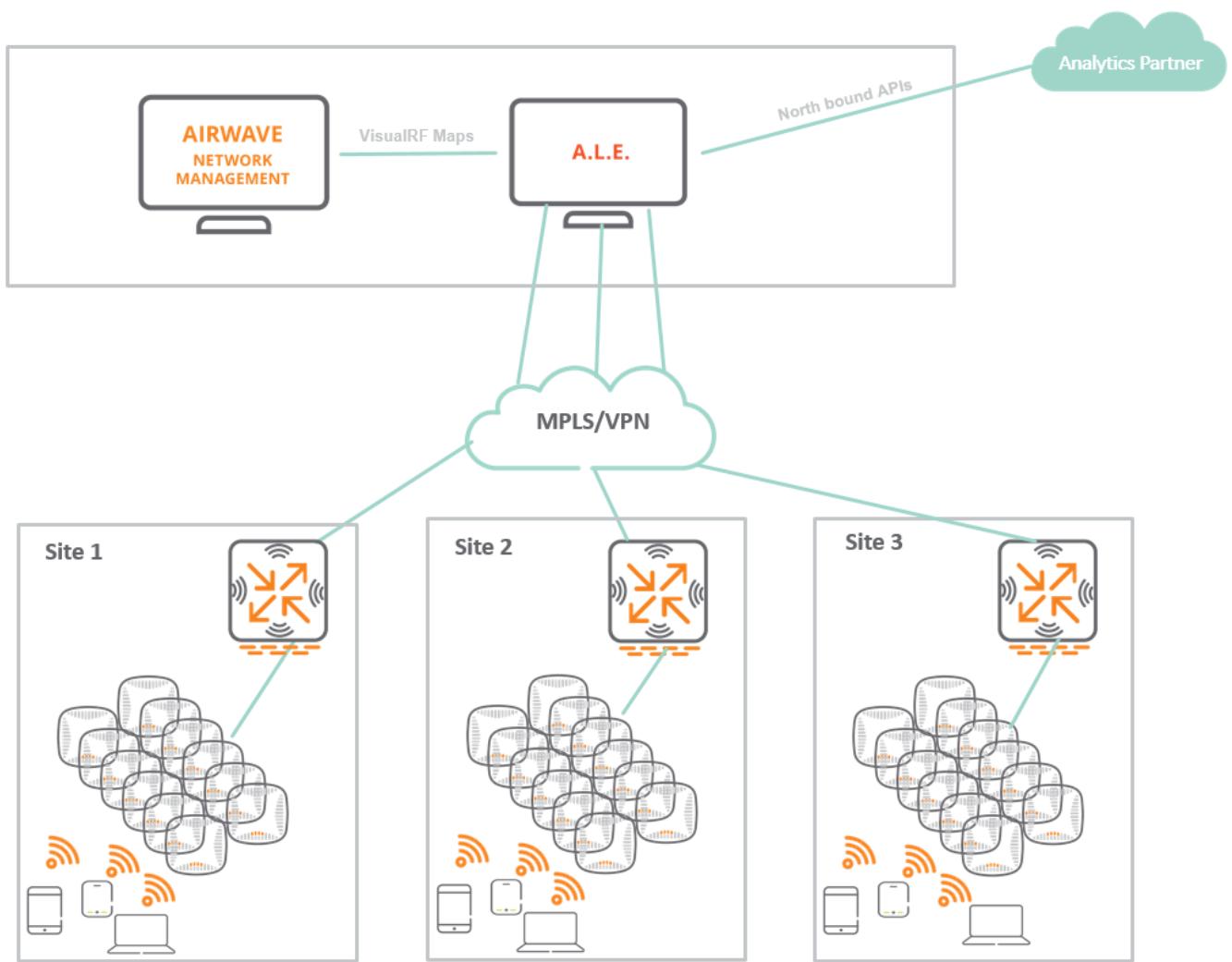


Figure 7 Design 3: Controller-based Distributed Network with Secure Connection between the Remote Sites and the Datacenter



Location accuracy is heavily dependent on AP placement and Radio Frequency (RF) design. This section describes the general guidelines to consider when designing location-ready wireless networks, either for greenfield deployments or existing networks.

This chapter includes the following topics:

- [AP Count on page 30](#)
- [Air Monitors on page 30](#)
- [AP Placement on page 31](#)
- [AP Separation on page 33](#)
- [Minimum RSSI on page 33](#)
- [Designing for a Location + Voice Ready WLAN on page 33](#)

AP Count

The bare minimum requirement for approximating the X,Y coordinates of any wireless device is for it to be detected by at least three APs and located inside the perimeter boundaries of these APs.

Generally, the more APs that are able to detect a device, the better the chances are of accurately locating it. If location accuracy is a requirement, it is recommended to design your WLAN for capacity vs. coverage.

When testing ALE-based locationing, especially for associated client devices, it is important to understand that APs are not designed to scan channels continuously, so the use of only 3 APs almost certainly means that the algorithm does not get the minimum 3 APs it needs to triangulate or fix locations and it will fall back to the single AP method. The right way to test is to have 7-8 APs at least, 1-2 of them being AM.

Air Monitors

For analytic use cases that rely on location data, it is important to understand how client devices behave on the WLAN. Different client devices have different probing behaviors. Associated clients in particular, do not probe as frequently as unassociated devices, especially if not constantly moving. As such, simply relying on RSSI received from client probe requests may affect the latency involved with location computations. For this reason, ALE also takes into account RSSI received from data frames on the WLAN. A higher volume and frequency of client data means ALE can compute locations with improved latency.

In the case of an AP serving clients on the WLAN, the AP may pick up data frame RSSI from clients that are associated with the AP because both AP and clients are exchanging data on the same channel. But for the same AP to pick up data frame RSSI from clients associated to other APs, it needs to hear on the same channel that the clients are exchanging data frames on. For example, if AP1 is serving client 1 on channel 1 and AP2 is serving client 2 on channel 11, the likelihood that AP1 will go off-channel and listen on channel 11 at the same time client 2 is transmitting data is probabilistically low. Since an AM scans channels more aggressively than an AP, it has a higher chance of picking up data frame RSSI from both channels 1 and 11, and therefore contribute towards more data for location computation.

If your use case requires analytics around associated devices, it is recommended to have a mix of APs and AMs (for example, 1 AM for every 3 APs) in your WLAN deployment. Associated devices can probe very infrequently especially if not constantly moving.

AP Placement

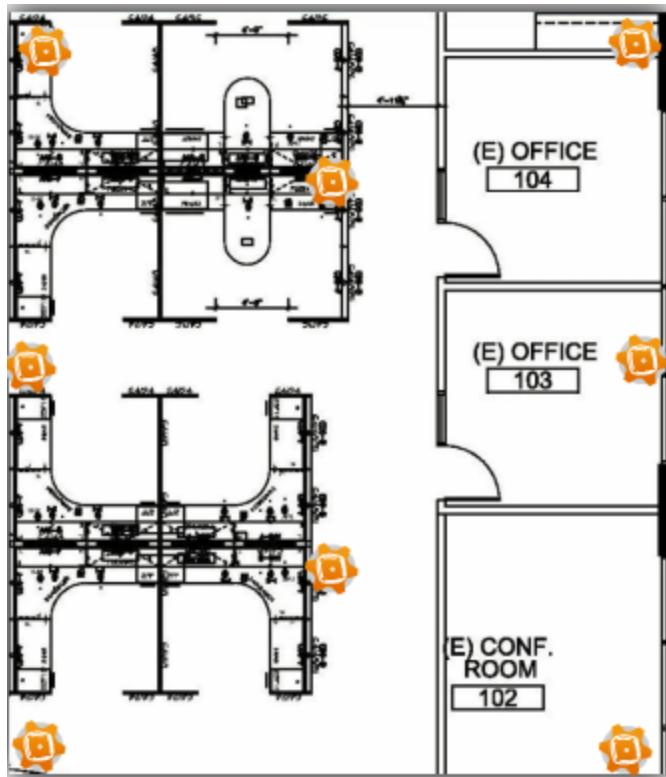
Ideal AP Placement

Perimeter APs

Most non-location-ready deployments are designed such that APs are located in the inner spaces of the floor plan. APs are generally not placed on the perimeters or corners since their coverage cells would bleed outside the floor, which may not be necessary as the WLAN, in most cases, is not required to serve clients outside the floor boundaries.

In a location ready network, APs should not only be placed in the center spaces but also spread across the perimeters and corners of the floor. This helps in improving location accuracy as the likelihood that a device is triangulated by more than 3 APs and at better signal strengths is higher.

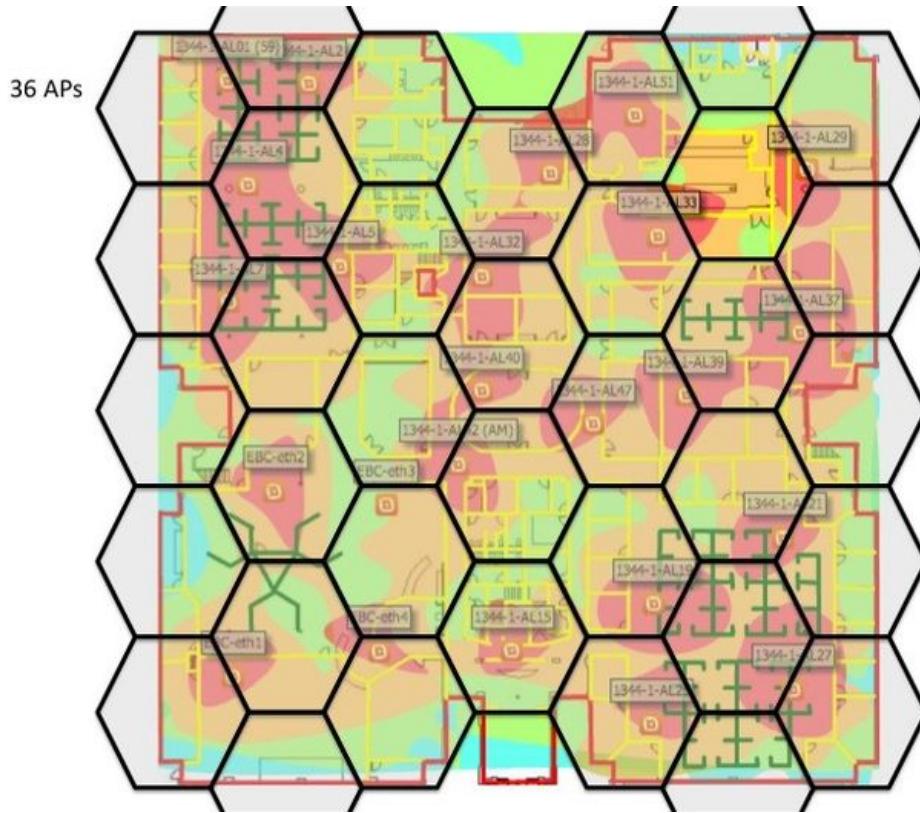
Figure 8 Ideal AP Placement = Center + Perimeter + Corner APs



Hexagonal AP Placement

Use a hexagonal pattern for AP layout. This ensures that distance is normalized along all directions. It is a very popular pattern to use (circles will be normalized, but you will find it hard to create tiles that cover your area. Hexagon is the smallest polygon with this property).

Figure 9 Hexagonal AP Layout



AP Mounting Height

If APs are mounted too high from the ground, as the distance between the AP and the client device increases, the RSSI differentiation at the AP starts reducing beyond a certain distance, which may affect location accuracy (see section below). Generally, for APs mounted 15-20 feet high from the ground, this may not be a problem as long as the client does not move too far away from the AP and is audible to at least three surrounding APs on the floor.

AP Placement Recommendations

Achieving all the above mentioned recommendations might not be possible in certain deployments. [Table 6](#) below tries to categorize the recommendation based on priority.

Table 6: AP Placement Recommendations

Recommendation	Priority	Comments
Voice Overlay	1	This is a must in all deployments to achieve triangulation which is the core requirement for location calculation.
AP every 2500 sq. feet or 50 feet apart and cover the edges	1	This is to help achieve a good coverage pattern and triangulation and is a must for most deployments.
Hexagonal pattern for AP layout	2	This is recommended but might be hard to achieve in certain scenarios due to the physical layout.
-65 dbm coverage	2	This is strongly recommended but might be hard to achieve in certain parts of the building. In those cases, ensure that there is at least a -75 dbm coverage in those areas.

AP Separation

The distance between APs can impact the location accuracy for a wireless client. As the client moves away from an AP, the signal level at which the AP hears the client decreases and the RSSI differentiation per distance is significant until a certain point (typically -65dBm) after which it becomes smaller and smaller as the client moves further away from the AP. This makes it difficult to accurately locate client devices at longer distances.

It is recommended that APs are separated by about 40-50 feet where possible, but not more than 60 feet.

Minimum RSSI

As mentioned above, there is lesser RSSI differentiation once the signal level that the AP hears from a client crosses the -65dBm mark. Since the APs depend on the RSSI in the probe/data packets coming from the client in order to calculate location, it is important to receive good RSSI.

Hence, it is recommended that APs be placed such that at any given point on the floor, at least three APs can hear the client at -65dBm or better.

Designing for a Location + Voice Ready WLAN

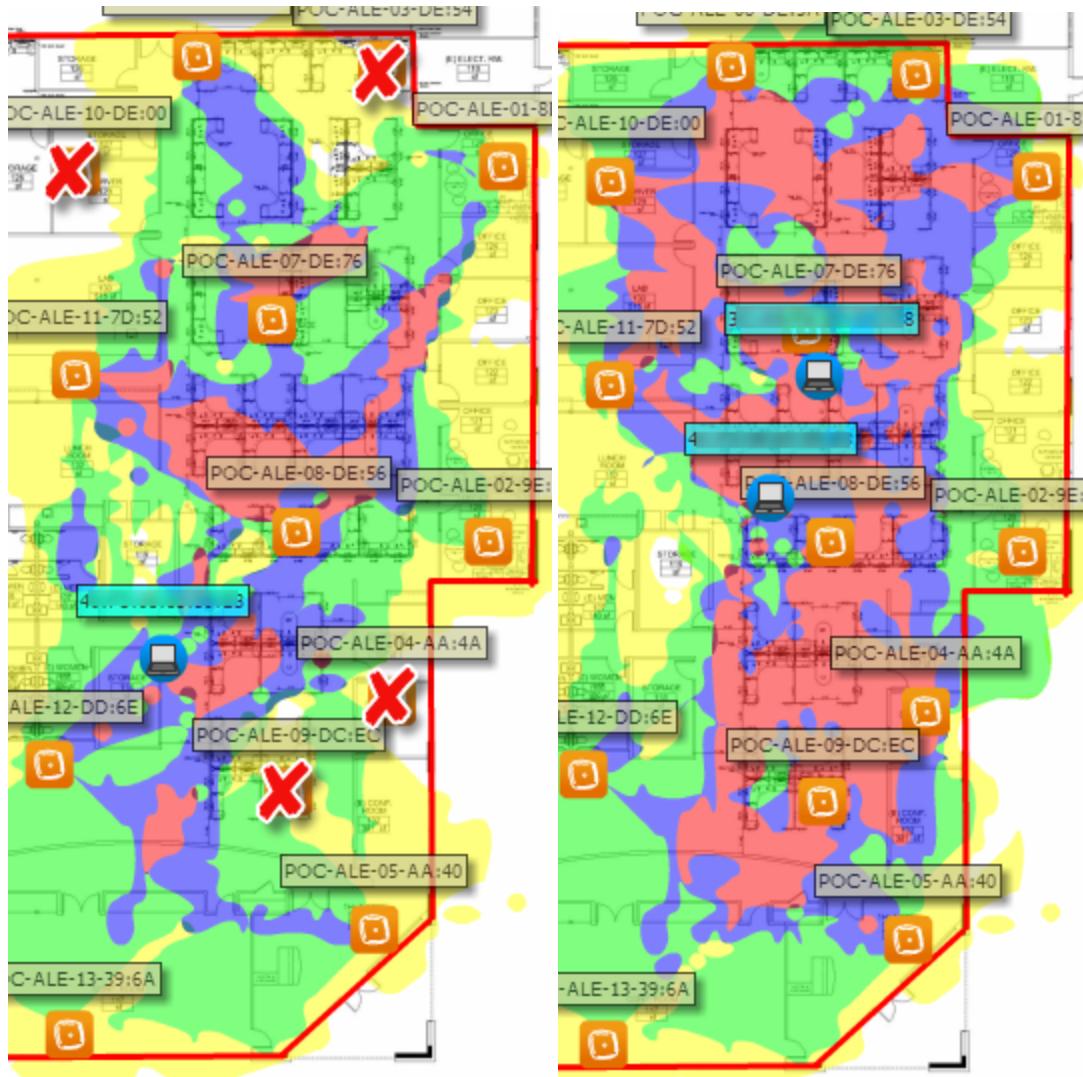
Typically, not many WLANs are designed with location tracking as the only use case. Rather, they are designed keeping voice, data, and location services in mind. So it is quite likely that designing a location-ready network will also meet the major requirements of voice WLANs such as -65dBm signal level, SNR, data rates, etc.

Example of a Non-voice-ready WLAN vs. Voice-ready WLAN

As you can see in the first image, there are quite a few points on the floor where the client most likely sees only two APs. Compare this to the second image which shows a more 'location-ready' WLAN with extra corner and perimeter APs. At most places on the floor map, the client can hear 3 APs or more, which means the voice clients will be able to make faster roaming decisions while still maintaining the quality of voice/video applications.

At the same time, the APs have a better chance of hearing the client at -65dBm or better at different points on the floor. Better RSSI differentiation means better location tracking accuracy.

Figure 10 Non-voice-ready WLAN vs. Voice-ready WLAN



At -65dBm cell coverage

- *client can see 2+ APs*
- *client can see 3+ APs*
- *client can see 4+ APs*
- X → *No APs*

 While using AirWave for validating voice overlay, it is very important to select only one frequency at a time as shown in the screenshots below. If you enable both frequencies, the AP count will be doubled by AirWave and this does not provide a true representation of voice overlay in your deployment.

Figure 11 One Frequency at a Time 2.4 GHz

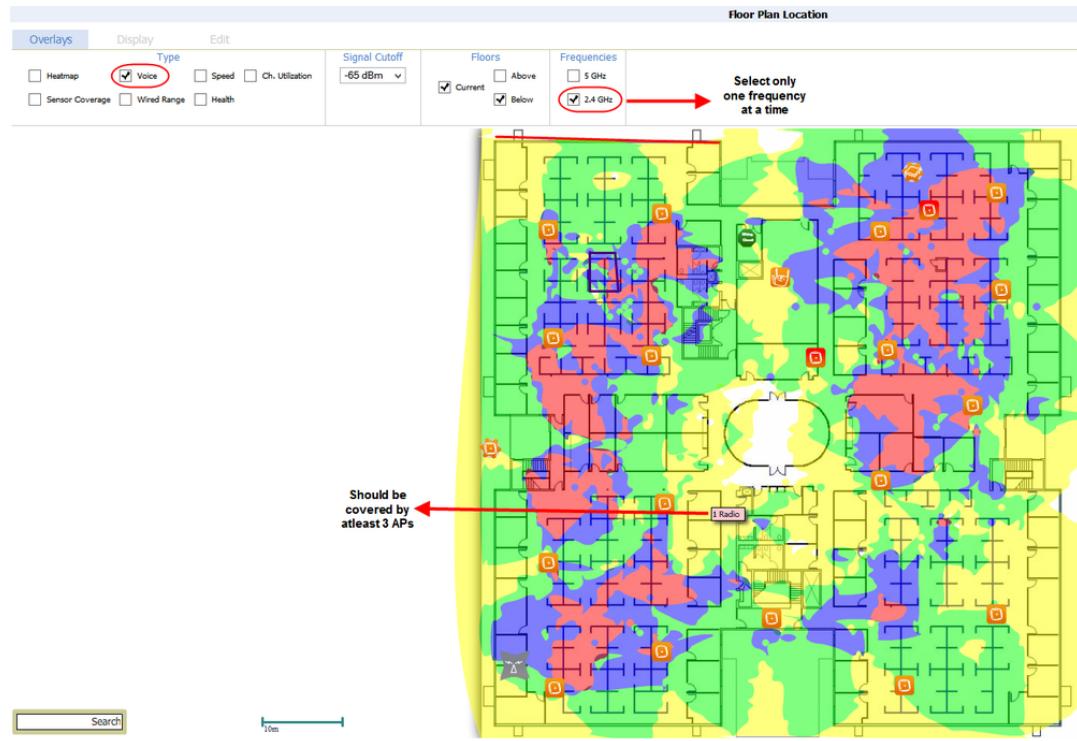
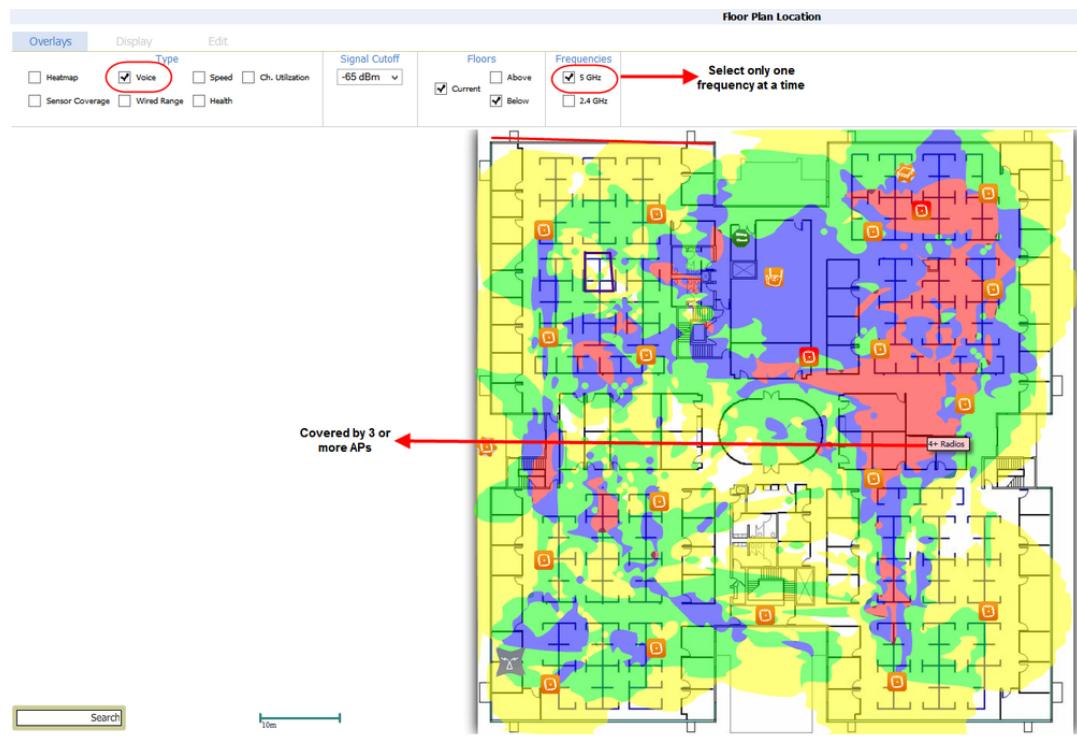


Figure 12 One Frequency at a Time 5 GHz



Refer to the ALE 2.0 User Guide for installation and configuration details.

This chapter includes the following topics:

- [Installing ALE on page 36](#)
- [Configuring the Deployment Modes on ALE on page 36](#)
- [Choosing the Right ALE Mode on page 37](#)
- [Best Practice Guidelines When Deploying ALE in Estimation Mode on page 37](#)
- [Best Practice Guidelines When Deploying ALE in Calibration Mode on page 38](#)

Installing ALE

Please refer to the chapter “Installing and Configuring ALE” in the ALE User Guide for more details.

Configuring the Deployment Modes on ALE

This guide provides a general overview and best practices around the three deployment modes on ALE. For additional details on each of the modes including how to configure them, please refer to the ALE User Guide.

Depending on the desired use case and level of location accuracy required, you can configure one of the three deployment modes on ALE.

This section includes the following deployment modes:

- [Context Mode on page 36](#)
- [Context with Device Location \(Estimated\) Mode on page 36](#)
- [Context with Device Location \(Calibration\) Mode on page 37](#)

Context Mode

This mode is also referred to as “Context-only” mode.

The “Context” mode is typically used in smaller, distributed deployments (for example, coffee shops or retail stores) where there are 1-3 APs per location. The Context mode provides a topic called “Proximity” as part of the North-Bound API that tells you which access point is closest to the wireless client, providing a rough location estimation. Each proximity message is associated with a client’s MAC address, the AP that hears it the strongest and the signal strength at which the AP hears the client.

This mode is especially useful for venues that want to provide analytics around passers-by vs. engaged visitors, average dwell time, new vs. repeat visitors, but cannot utilize location coordinates (x,y) due to the sparse deployment of APs.

This mode is suited when a floor plan is not available and does not require ALE to communicate with AirWave.

Context with Device Location Modes:

In networks where a good density of APs are deployed according to best practice guidelines for location-ready networks, you would use one of the “Context with Device Location” modes.

Context with Device Location (Estimated) Mode

This mode is also referred to as “Estimation” mode.

In the Estimation mode, ALE learns about floor plans and AP placement information from AirWave VisualRF. It couples this information with AP-AP RSSI information learned from the WLAN to create a Positioning Database (PDB). ALE compares all incoming client RSSI with this PDB to compute approximate x,y locations for the client devices.

Context with Device Location (Calibration) Mode

This mode is also referred to as “Calibration” mode.

Calibration mode is used when a higher degree of location accuracy is required. This mode requires pre-deployment floor calibration (sometimes referred to as fingerprinting) which is done by drawing ‘paths’ on the floor map in ALE and walking a reference client device (with the Aruba Nao Logger app installed on the device) along these paths. Once all floors are calibrated, a PDB is generated, based on which ALE computes locations for actual floor traffic, post-deployment. The calibration can be performed by a single person or multiple persons.

Calibration mode does not use AirWave to download the floor maps. Instead, all floor maps, calibration paths, and geofences are defined on ALE (NaoCampus).

Choosing the Right ALE Mode

As mentioned above, Calibration mode delivers a higher degree of location accuracy when compared to Estimation mode. The improvement in accuracy in Calibration mode comes at the cost of having the manpower to calibrate each floor before use. Since Estimation mode is easier to set up, it is sometimes preferred by customers. But if greater location accuracy is desired, it is strongly recommended to use Calibration mode in your deployment.

In large deployments where the network keeps changing every now and then (addition of new buildings and floors, requirement to define new calibration paths on the floor, etc.) post-calibration, Estimation mode may be a more viable option.

Best Practice Guidelines When Deploying ALE in Estimation Mode

Follow the guidelines mentioned below for optimal results in Estimation mode:

- For the best location results, ensure that these guidelines are followed. See [AP Placement and Design on page 30](#). In general, it is recommended to have a good density of APs, a good ratio of APs and AMs (highly recommended for analytics around associated devices), and APs placed at the edges of the floor.
- When testing ALE-based locationing for associated devices, it is important to understand that APs are not designed to scan channels as aggressively as AMs, so the use of only 3 APs almost certainly means that the algorithm does not get the minimum 3 APs it needs to triangulate or fix locations and it will fall back to the Single AP method. The right way to test is to have at least 7-8 APs, 1-2 of them being AM.
- It is recommended to keep the Single AP location feature enabled on ALE. If Single AP is disabled, it is likely that some clients that are not heard by at least 3 or more APs might be discarded. By keeping the Single AP feature enabled, we are ensuring that locations for clients, even though they cannot be properly triangulated, are approximated and are thus accounted for.
- For very large deployments (with a lot of floors), it is recommended to export the VisualRF backup file from AirWave and upload it directly on ALE. By doing this, ALE does not need to communicate with AirWave during bootstrap.
- In Estimation mode, the fractional deletion of a few APs will not affect location accuracy to a great extent. When new APs are added in an active deployment, they will not be considered during location calculations. It is recommended to repeat sitefetch (by re-adding the AirWave server) and then regenerate the PDB on ALE when APs are added to or removed from the WLAN.

Best Practice Guidelines When Deploying ALE in Calibration Mode

Follow the guidelines mentioned below for optimal results in Calibration mode:

- ALE uses the NaoCampus tool to define floors, calibration paths, and geofences. Once your deployment is calibrated and the PDB is published, new changes to NaoCampus such as addition of new floors for calibration or drawing additional calibration paths cannot be made.
- Calibration mode is recommended for WLANs that do not change significantly post-deployment. For example, all floors and areas are calibrated in one go; there is not a significant change in the number of APs that are added or deleted post-calibration, etc.
- In Calibration mode, the fractional deletion of a few APs will not affect location accuracy to a great extent. When new APs are added in an active deployment, they will not be considered during location calculations. Re-calibration is required to take into account data from new APs.
- If additional floors are required to be calibrated, the PDB needs to be deleted and calibration needs to be performed again for all floors (including the ones that were previously calibrated). As such, Calibration mode may or may not be suitable for very large campuses where the deployment keeps changing post-calibration (for example, addition of new buildings and floors, requirement to draw additional calibration paths on the floor). In these scenarios, Estimation mode may be a more viable option.
- Minor changes can be made post-calibration. For example, calibrated paths can be re-calibrated using the NaoLogger app.
- Calibration requires an Internet connection to download the Open Street maps in order to enable latitude and longitude values in location messages.
- For the best location results, ensure that these guidelines are followed. See [AP Placement and Design on page 30](#). In general, it is recommended to have a good density of APs, a mix of APs and AMs (highly recommended for analytics around associated devices), and APs placed at the edges of the floor.
- AMs do not broadcast or respond to client probe requests. So when deploying AMs in Calibration mode, the AMs need to be converted to APs during the calibration process (so that they are able to respond to probes from the calibrating device) and later converted back to AMs in the actual deployment (after PDB is published).
- Although not a strict requirement, it is recommended to disable ARM during calibration. Note that only calibration may be affected by ARM. Location computation is never affected by ARM settings.
- It is recommended to have an Internet connection in order for NaoCloud to download OpenStreetMaps when configuring ALE in Calibration mode. Please note that an Internet connection is a must if ALE is behind a firewall and requires a WebSocket connection to cloud-based analytics applications.
- When drawing paths on ALE for the purposes of calibration, it is recommended to draw them in a grid pattern, with each parallel grid about 15-25 feet apart (10-15 feet apart for tighter location accuracy).
- Before using NaoLogger please ensure TCP connections to ALE on port 443 are allowed by your network firewall.
- When using Aruba NaoLogger for calibration, it is recommended to click (or lay your marker) at every intersection point on the floor, although the algorithm is robust enough to handle a few missed intersections.
- When walking a path (say from point A to point B) during the calibration process, it is recommended to also walk back (for example, from point B to point A) to enhance your calibration data.
- When calibration is performed by multiple people at a time, it is recommended that the same type of device be used by everyone during the calibration process.

This troubleshooting chapter includes the following topics:

- [General ALE Issues on page 39](#)
- [ALE-AirWave Issues on page 58](#)
- [WebSocket Connection Issues on page 69](#)
- [ALE Troubleshooting Logs on page 73](#)

General ALE Issues

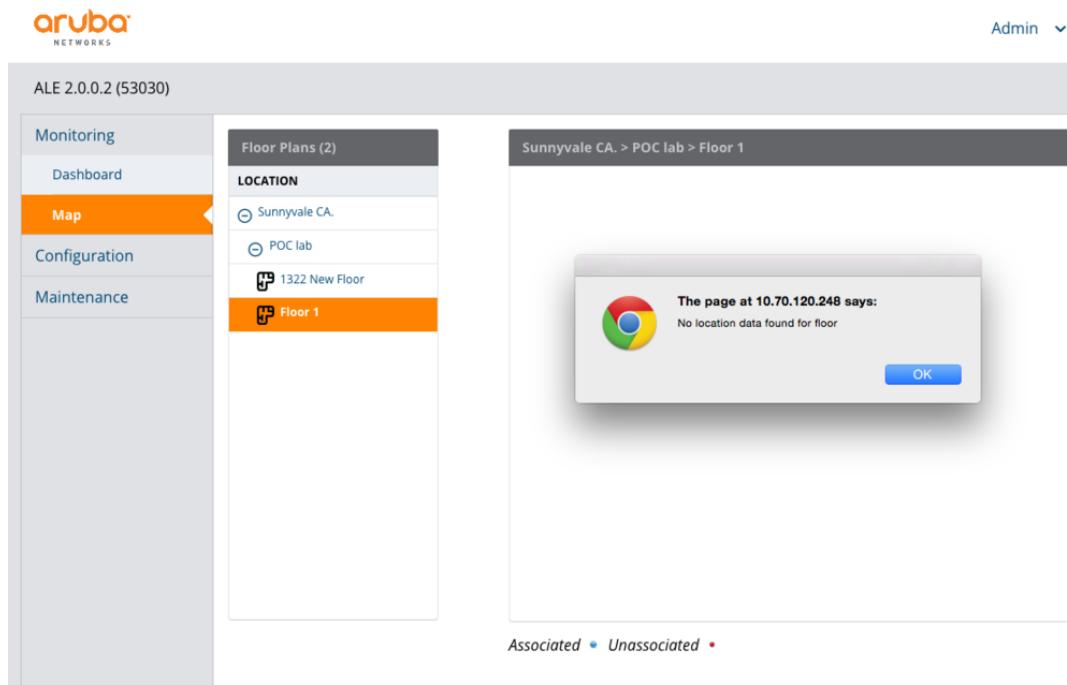
This section includes the following topics:

- [ALE UI Error Message: "No location data found for floor" on page 39](#)
- [ALE UI Error: "Database regenerating failed" on page 42](#)
- [No User-specific Information Visible in Presence/Location Feed on page 43](#)
- [No Location Events Generated on ALE on page 46](#)
- [Troubleshooting Location Issues For a Specific Client on page 51](#)
- [Very Few Location Events on ALE Compared to Device Traffic on the WLAN on page 53](#)
- [No Presence/Location Events for Unassociated devices on the WLAN on page 54](#)

ALE UI Error Message: "No location data found for floor"

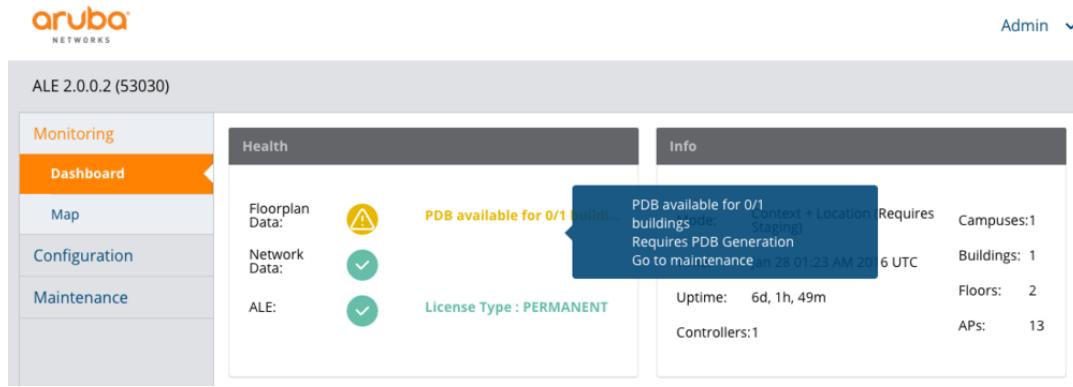
Problem: Selecting a floor on the ALE Map dashboard will throw out an error "No location data found on floor".

Figure 13 *Error Message: No location data found for floor*



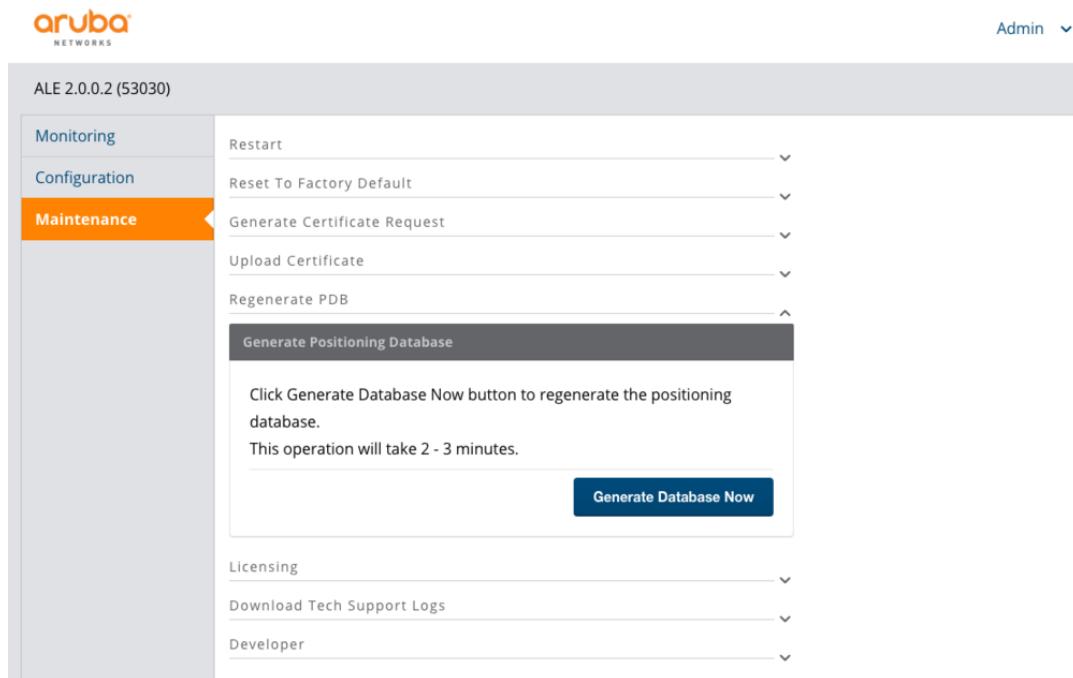
Check 1: Is this a newly configured ALE instance? Has the ALE mode been changed recently? It is likely that the PDB has not been generated yet.

Figure 14 Generated PDB



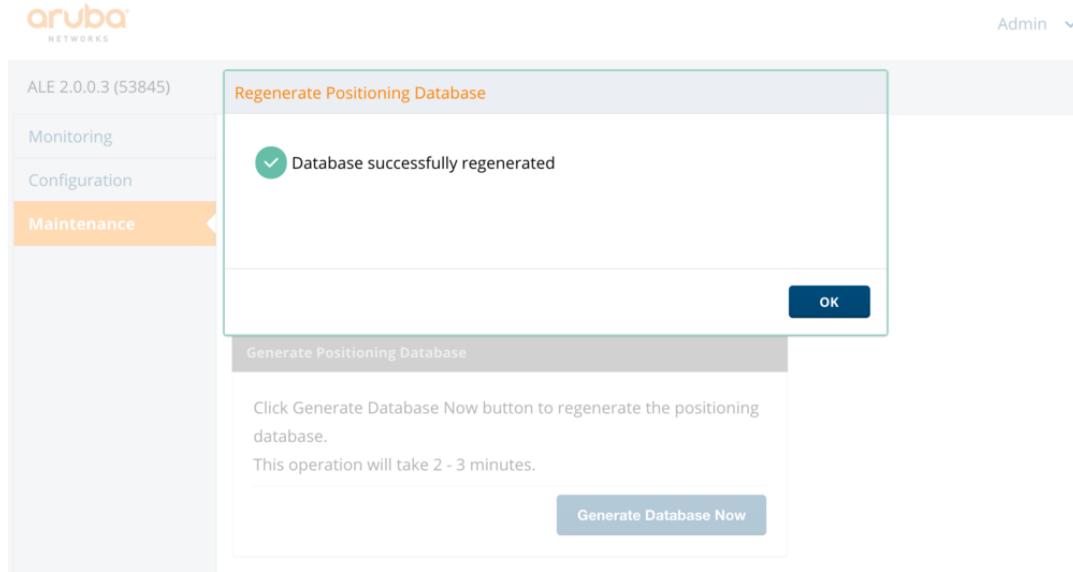
Under Maintenance > Regenerate PDB > click on **Generate Database Now**:

Figure 15 Maintenance PDB



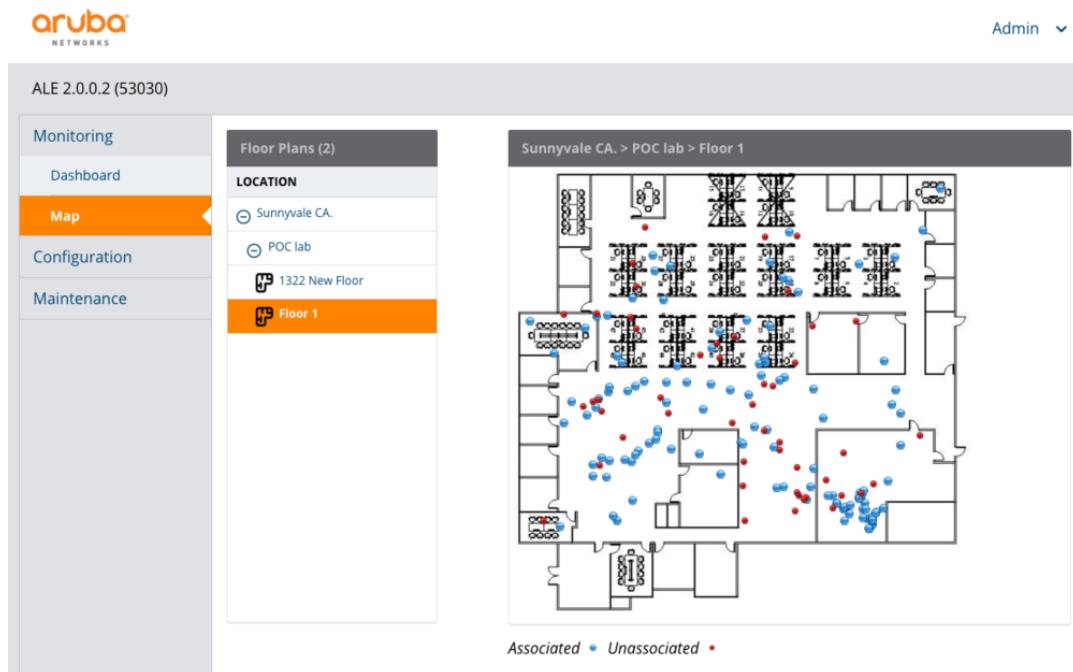
This might take a few minutes, depending on the size of your WLAN deployment.

Figure 16 Database Successfully Generated



Now you should see blue and red dots (associated and unassociated clients, respectively) on the ALE Map.

Figure 17 ALE Map Dashboard



Check 2: Does the feed-reader output display any location events?

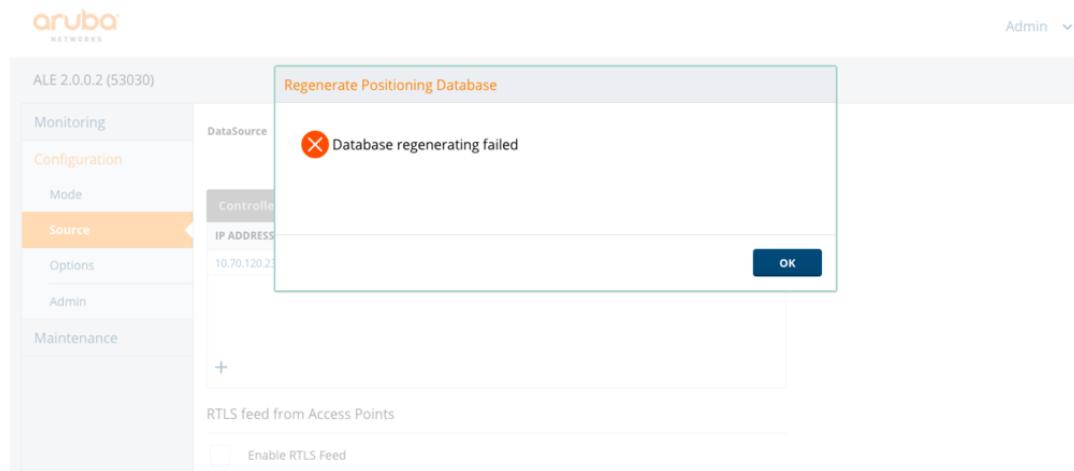
```
[root@ale ~]# /opt/ale/bin/feed-reader -f location
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "location"
```

If no location events are generated, follow the troubleshooting steps mentioned in the section titled [No Location Events Generated on ALE on page 46](#), in order to narrow down the issue.

ALE UI Error: "Database regenerating failed"

When you add a controller entry on ALE and are prompted to regenerate the PDB, doing so results in a "Database regenerating failed" error message.

Figure 18 Database regenerating failed



Check 1: Are there any communication issues between ALE and the controller?

Try performing a Telnet connection to the controller on port 4343 (the port that ALE uses to communicate with the controller).

If you see this:

```
[root@ale ~]# telnet 10.70.120.237 4343
Trying 10.70.120.237...
telnet: connect to address 10.70.120.237: No route to host
[root@ale ~]#
```

Or this:

```
[root@ale ~]# telnet 10.70.120.237 4343
Trying 10.70.120.237...
telnet: connect to address 10.70.120.237: Connection refused
[root@ale ~]#
```

The above outputs indicate that ALE cannot communicate with the controller either due to a routing issue, or there is a firewall on the network that is blocking connections to the controller. Check for end-to-end network connectivity and also the firewall settings in your network to make sure the controller is reachable.

You can also use a port scanning utility such as Nmap to further investigate firewall issues. Nmap is not installed on ALE by default.

Here's an example of a successful port scan:

```
[root@ale ~]# nmap -p 4343 10.70.120.237
Starting Nmap 5.51 ( http://nmap.org ) at 2016-01-28 21:47 UTC
Nmap scan report for 10.70.120.237
Host is up (0.00039s latency).
PORT      STATE SERVICE
4343/tcp  open  unicall

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
[root@ale ~]#
```

Check 2: You may also encounter this error when you have APs placed on VisualRF in AirWave as "planned" APs and not "deployed" APs.

The following error will be observed in the ale-jwebapp log (/opt/ale/ale-jwebapp/logs/ale-jwebapp.log) that indicates the placement of planned APs in AirWave:

```
ERROR com.aruba.ale.derby.db.Airwave - Error inserting accesspoint in DB. Message:A truncation error was encountered trying to shrink VARCHAR 'AP12_THE_RESERVE_RECEPCION_VILLAS_52-53_RACK52' to length 36
```

To resolve this issue, ensure that all APs are placed on VisualRF as deployed APs.

Check 3: You can also look at the ale-jwebapp logs to investigate any other issues that caused the PDB generation to fail.

No User-specific Information Visible in Presence/Location Feed

Problem: The ALE APIs do not display any client-specific data such as MAC addresses, IP addresses, and user names.

```

[root@ale ~]# /opt/ale/bin/feed-reader -f presence
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "presence"
[1] Recv event with topic "presence"
seq: 55492
timestamp: 1453937605
op: OP_DELETE
topic_seq: 3522
source_id: 005056852F31
presence {
    associated: true
    hashed_sta_eth_mac: E62E7834EF4419D0BBF9DFC16B26BDD22E4D7B47
    ap_name: POC-ALE-11-a7:78
    radio_mac {
        addr: 18:64:72:fa:77:90
    }
}
[root@ale ~]# /opt/ale/bin/feed-reader -f location
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "location"
[1] Recv event with topic "location"
seq: 56222
timestamp: 1453937752
op: OP_UPDATE
topic_seq: 36568
source_id: 005056852F31
location {
    sta_location_x: 15.9103
    sta_location_y: 92.6017
    error_level: 48
    associated: false
    campus_id: FCCD5881918E3C8D8628130773403AA6
    building_id: 659685E876A6325EB3F974FBBBA530F8
    floor_id: 8B5207D8FDB1319AB35A72A32C7E5937
    hashed_sta_eth_mac: E2F01DE79509D810D9C816C307C0024BD8540B9C
    geofence_ids: EAD42871E6A53A1193CC243700FD4497
    loc_algorithm: ALGORITHM_ESTIMATION
    unit: FEET
}

```

Check: Is anonymization enabled on ALE?

Under **Configuration > Options:**

Figure 19 Anonymization Enabled

The screenshot shows the Aruba ALE 2.0.0.2 (53030) configuration interface. The left sidebar has tabs: Monitoring, Configuration, Mode, Source, Options (which is selected and highlighted in orange), Admin, and Maintenance. The main panel is titled 'General'. It contains several configuration items:

- Enable Single Ap Location Calculation
- Enable Geo Fences
- Enable Anonymization (this checkbox is highlighted with a red box)

Below these are dropdown menus for 'Hash Generation Schedule' set to 'NEVER' and 'Distance metric' set to 'FEET'. There are also collapsed sections for 'NTP Server', 'WebSocket Tunnel', and 'Logging & Notifications'.

Disable anonymization on ALE by unchecking the Enable Anonymization knob and click on **Apply**. Verify that client MAC addresses are now visible in the ALE output feed.

```

[root@ale ~]# /opt/ale/bin/feed-reader -f presence
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "presence"
[1] Recv event with topic "presence"
seq: 58026
timestamp: 1453938166
op: OP_DELETE
topic_seq: 3698
source_id: 005056852F31
presence {
    sta_eth_mac {
        addr: 92:68:c3:e4:23:18
    }
    addresses are now visible
}
associated: false
hashed_sto_eth_mac: 224738D5D14658BA35B3CAED07A4E959A2A07624
ap_name: POC-ALE-09-35:96
radio_mac {
    addr: 18:64:72:e3:59:70
}
}

[root@ale ~]# /opt/ale/bin/feed-reader -f location
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "location"
[1] Recv event with topic "location/50:87:89:bf:74:0c"
seq: 57993
timestamp: 1453938141
op: OP_UPDATE
topic_seq: 37935
source_id: 005056852F31
location {
    sta_eth_mac {
        addr: 50:87:89:bf:74:0c
    }
    addresses are now visible
}
sta_location_x: 20.9299
sta_location_y: 84.8438
error_level: 35
associated: true
campus_id: FCCD5881918E3C8D8628130773403AA6
building_id: 659685E876A6325EB3F974FBBBA530F8
floor_id: 8B5207D8FDB1319AB35A72A32C7E5937
hashed_sto_eth_mac: 513941ED69088F1E7E6CC18D5CC63756CDCEEBAA2
geofence_ids: EAD42871E6A53A1193CC243700FD4497
loc_algorithm: ALGORITHM_ESTIMATION
unit: FEET
}

```

No Location Events Generated on ALE

Problem: The ALE feed-reader output does not display any location events.

```

[root@ale ~]# /opt/ale/bin/feed-reader -f location
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "location"

```

Check 1: Does ALE generate any events at all?

```
[root@ale ~]# /opt/ale/bin/feed-reader
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: ""
```

Check 2: Is ALE receiving client RSSI feed from the controller/IAP?

If the source for client RSSI is AMON:

```
[root@ale ~]# /opt/ale/bin/feed-reader -e tcp://localhost:7778
Attempting to 'connect' to endpoint: tcp://localhost:7778
Connected to endpoint: tcp://localhost:7778
Subscribed to topic: ""
```

If the source for client RSSI is RTLS:

```
[root@ale ~]# /opt/ale/bin/feed-reader -e tcp://localhost:7777
Attempting to 'connect' to endpoint: tcp://localhost:7777
Connected to endpoint: tcp://localhost:7777
Subscribed to topic: ""
```

Check 3: Is the controller's **physical** IP correctly configured on ALE?

Only the physical/switch IP of the controller is recommended to be configured on ALE.

Check 4: Is the ALE IP address correctly configured on the controller/VC?

On the controller: Under **Configuration > Management** > Scroll down to the **Analytics and Location Engines** section > Verify whether an ALE entry is configured.

Figure 20 Analytics and Location Engines Configuration

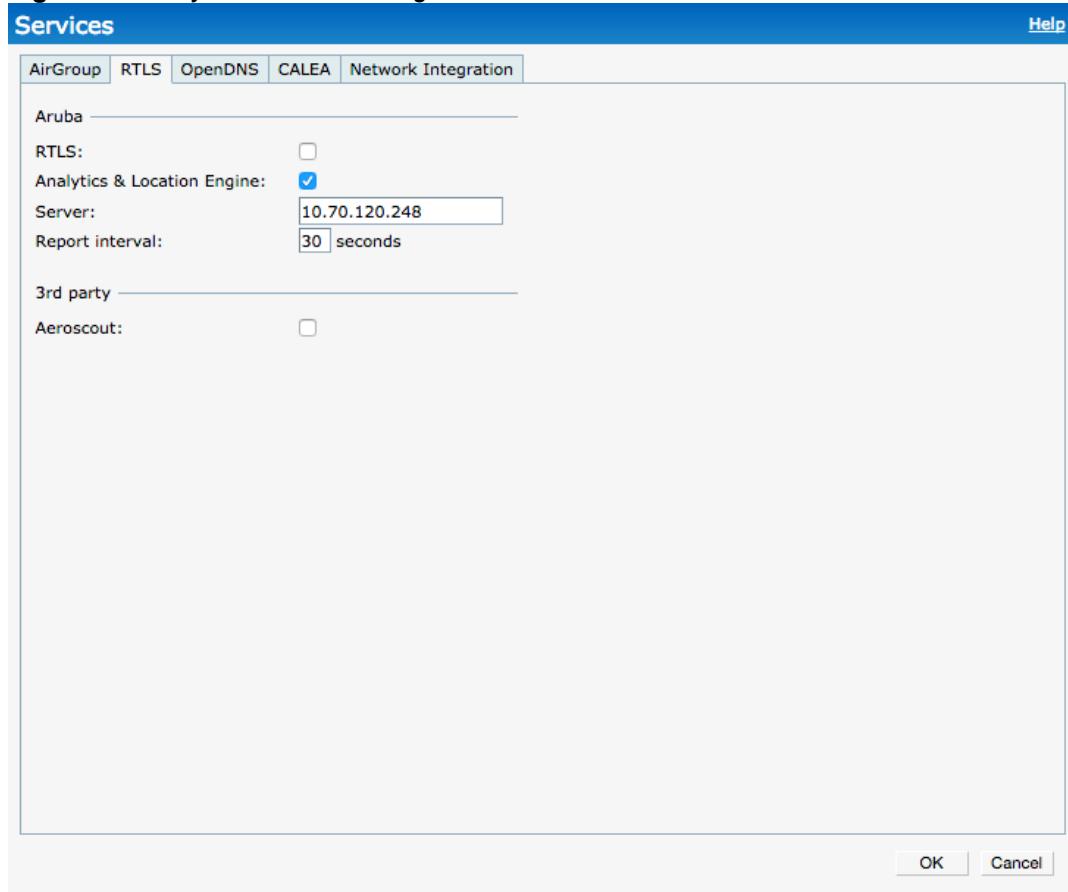
The screenshot shows the configuration interface for Analytics and Location Engines. On the left, a navigation tree includes General, Administration, Certificates, SNMP, Logging, Clock, Guest Provisioning, Captive Portal, SMTP, Bandwidth Calculator, Threshold, ADVANCED SERVICES, Redundancy, AirGroup, IP Mobility, Stateful Firewall, External Services, VPN Services, Wired Access, All Profiles, and BRANCH. Under BRANCH, there is a link to Smart Config and E-mail Support.

The main content area contains several configuration sections:

- Server Certificate:** Set to Default.
- IDP Server Certificate:** Set to Default.
- Configure Cipher LOW/MEDIUM/HIGH:** Set to High.
- LCD Menu:** A table showing various menu items with enable/disable checkboxes. Most items are set to enable.
- Configure Skype4B:** Set to HTTP.
- Mobility Manager Servers:** A table showing no results found. Buttons for New and Apply are present.
- AirWave Servers:** A table showing one entry for a primary server with profile default-amp. Buttons for New and Delete are present.
- Analytics and Location Engines:** A table showing one entry for a primary server with profile default-ale. Buttons for New and Delete are present.
- Commands:** Buttons for Apply and View Commands.

On the IAP VC: Under VC homepage > Expand on **More > Services > RTLS** tab > Verify whether the Analytics & Location Engine knob is checked and the ALE IP address is configured.

Figure 21 Analytics & Location Engine and IP Address



Check 5: Is NTP correctly set up on ALE and the controller/IAP?

On ALE: Under **Configuration > Options > NTP Server** > Verify whether an IP address is configured for the NTP server. If not, specify one and click on **Apply**.

Check 6: Is ALE receiving data from the controller/IAP?

For AMON feed from the controller:

```
[root@ale ~]# tcpdump port 8211
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
00:01:41.424288 IP 10.70.20.252.8211 > 10.70.120.248.8211: UDP, length 1370
00:01:41.424848 IP 10.70.20.252.8211 > 10.70.120.248.8211: UDP, length 1370
00:01:41.425149 IP 10.70.20.252.8211 > 10.70.120.248.8211: UDP, length 1370
```

For RTLS feed from the controller, filter on the port that was configured on the controller and ALE. In the example below, port 8855 was used. The source IP addresses in the output correspond to traffic from different APs.

```
[root@ale ~]# tcpdump port 8855
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
19:49:18.387301 IP 10.70.20.203.fuscript > 10.70.120.248.8855: UDP, length 392
19:49:18.515666 IP 10.70.20.198.fuscript > 10.70.120.248.8855: UDP, length 392
19:49:18.581172 IP 10.70.20.194.fuscript > 10.70.120.248.8855: UDP, length 392
19:49:18.806412 IP 10.70.20.205.fuscript > 10.70.120.248.8855: UDP, length 436
```

For traffic from an Instant AP cluster, filter on the destination port that was configured on the Instant VC. TCP 8088 is used in the example command below.

```
[root@ale ~]# tcpdump port 8088
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

Check 7: Are the APs on the controller/IAP cluster able to hear client devices in their vicinity?

In this example, the output has been divided into multiple sections to better fit on the pages of this document. In the actual command-line interface, it will appear in a single, long table.

On the controller:

```
(pfe-controller) # show ap monitor client-list ap-name POC-ALE-03-81:92

Monitored Client Table
-----
mac          bssid          essid          channel sta-type
---          ----          -----          -----
60:f8:1d:b7:64:48 ac:a3:1e:55:89:72 hpn-byod      48    interfering
84:3a:4b:34:1d:5e 94:b4:0f:5c:a2:b0 App-Test-JI     36    interfering
5c:e0:c5:89:2b:f0 ac:a3:1e:55:6f:51 ethersphere-wpa2 48    interfering
c8:e0:eb:39:26:bf 18:64:72:e5:8e:90 pfe-psk       36    valid
60:02:b4:85:31:99 18:64:72:40:b9:d1 Angel-Media   36    interfering
e8:b1:fc:87:89:76 40:e3:d6:d7:6c:b0 Hd-lab-k12    36    interfering
5c:c5:d4:c1:17:9f 40:e3:d6:d7:6c:b0 Hd-lab-k12    36    interfering

auth phy-type      dt/mt          ut/it snr rssi cl-delay
---- -----      ----          ----- --- --- ----- -----
no   80211a-VHT-40 559/559      1/0    14  81  -
yes  80211a-HT-40  756/756      3/2    10  85  -
no   80211a-VHT-40 1385/1385    3/43   13  82  -
yes  80211a-VHT-80 1258/1258    11/10  18  77  -
yes  80211a-HT-40  82897/82897  9/8    16  79  -
yes  80211a-VHT-80 3054933/3054933 2/1    11  84  -
yes  80211a-VHT-80 3044573/3044573 3/2    9   86  -
```

<output snipped>

On the IAP:

```

AP2# sh ap monitor sta-list

Monitored Client Table
-----
mac          bssid        essid      channel sta-type
---          ----        ----      ----- -----
ac:d1:b8:8a:6f:1f 40:e3:d6:d7:6c:b0 Hd-lab-k12 36   interfering
a4:5e:60:4a:4d:a3 94:b4:0f:5c:f7:50 IAP POC Lab 165   interfering
24:77:03:d5:ba:54 94:b4:0f:5c:f7:50 IAP POC Lab 165   interfering
a4:67:06:46:55:e7 94:b4:0f:5c:f7:50 IAP POC Lab 165   interfering
40:e2:30:1c:82:4b 40:e3:d6:d7:6c:b0 Hd-lab-k12 36   interfering
8c:29:37:e8:2d:70 94:b4:0f:5c:f7:50 IAP POC Lab 165   interfering

auth phy-type    dt/mt      ut/it      snr rssi cl-delay
---- -----    ----      -----      --- ---- -----
yes 80211a       296962/136372 7/17     17 78  -
yes 80211a-VHT-20 15260/7148   4/2      24 71  -
yes 80211a-HT-20 387504/176661 8/3      18 77  -
yes 80211a-HT-20 4048/1902    437/203  15 80  -
yes 80211a       385484/176819 18/46898 19 76  -
no 80211a        1318/628    579/272   27 68  -

<output snipped>

```

If you have verified all the above steps and still do not see any location events on ALE, please open a Support ticket.

Troubleshooting Location Issues For a Specific Client

Problem: Very little -to- no locations are being generated for a specific client device.

Check 1 (This check applies only when using Calibration mode on ALE): Check whether your device's location is being computed in the Nao Logger app on the device, when "Demo" mode is selected on the app. In "Demo" mode, the location algorithm on the device runs based on RSSI information recorded on the device and not based on network RSSI. These locations are based on the local PDB in Nao Cloud (and not the PDB that is published from Nao Cloud to ALE). If you see your location being computed on the app, it means that the PDB is good.

Check 2: Check the RSSI feed to see how many APs are contributing RSSI for the given client MAC. If no APs are reporting RSSI, then it might be an issue with ALE not being able to receive data from the WLAN (also see section [No Location Events Generated on ALE on page 46](#) for further troubleshooting).

If the source for RSSI is AMON:

```

[root@ale ~]# /opt/ale/bin/feed-reader -e tcp://localhost:7778 | grep 18:5e:0f:90:15:73 -
A16
Attempting to 'connect' to endpoint: tcp://localhost:7778
Connected to endpoint: tcp://localhost:7778
Subscribed to topic: ""

```

If the source for RSSI is RTLS:

```

[root@ale ~]# /opt/ale/bin/feed-reader -e tcp://localhost:7777 | grep 18:5e:0f:90:15:73 -
A16
Attempting to 'connect' to endpoint: tcp://localhost:7777
Connected to endpoint: tcp://localhost:7777
Subscribed to topic: ""

```

Check 3: If the RSSI feed shows any number of APs reporting the device MAC, look for what is happening to the device in the ale-location logs. If you can for a short while, turn up ale-location logs to TRACE level (if not TRACE, then at least DEBUG level). Then see what you can find out about this device by filtering on the MAC address of the device.

Figure 22 Trace Level

The screenshot shows the Aruba ALE 2.0.0.3 configuration interface. The left sidebar has tabs: Monitoring, Configuration, Mode, Source, Options (which is selected), Admin, and Maintenance. The main area is titled "ALE 2.0.0.3 (53845)". Under "Logging & Notifications", there is a "Logging Levels" section. A dropdown menu is open over the "location" row, showing levels: ERROR, WARNING, INFO, DEBUG, and TRACE, with TRACE selected. Other rows show "platform" (INFO), "webapp" (INFO), and "persistance" (INFO). Below the table are two checkboxes: "Syslog Server" and "Send email notification for alerts". At the bottom right are "Cancel" and "Apply" buttons. The top right corner shows "Admin".

```
[root@ale ~]# tail -F /opt/ale/ale-location/logs/ale-location.log | grep 185e0f901573

2016-02-19 10:41:53.077 PST [pool-1-thread-2] DEBUG c.a.a.l.records.StationRecord - Station 185e0f901573 scheduled with 14 RSSIs:[RssiRecord [rssi=-40, tsSecs=1455907829, bssid=aca31e5a2750, channel=0], RssiRecord [rssi=-70, tsSecs=1455907819, bssid=aca31e59a0d0, channel=0], RssiRecord [rssi=-39, tsSecs=1455907823, bssid=aca31e5a2750, channel=0], RssiRecord [rssi=-83, tsSecs=1455907788, bssid=186472e18bf0, channel=0], RssiRecord [rssi=-58, tsSecs=1455907735, bssid=aca31e59a6f0, channel=0], RssiRecord [rssi=-58, tsSecs=1455907734, bssid=aca31e59a6f0, channel=0], RssiRecord [rssi=-66, tsSecs=1455907785, bssid=aca31e5a4c30, channel=0], RssiRecord [rssi=-84, tsSecs=1455907805, bssid=aca31e59c3f0, channel=0], RssiRecord [rssi=-72, tsSecs=1455907826, bssid=aca31e59a2b0, channel=0], RssiRecord [rssi=-60, tsSecs=1455907827, bssid=aca31e599cb0, channel=0], RssiRecord [rssi=-60, tsSecs=1455907807, bssid=aca31e599cb0, channel=0], RssiRecord [rssi=-71, tsSecs=1455907820, bssid=aca31e59a2b0, channel=0], RssiRecord [rssi=-86, tsSecs=1455907811, bssid=aca31e59a2d0, channel=0], RssiRecord [rssi=-86, tsSecs=1455907810, bssid=aca31e59a2d0, channel=0]]]

2016-02-19 10:41:53.125 PST [engine-worker-4] DEBUG c.a.a.l.records.StationRecord - New location for station 185e0f901573: x=70.044075, y=81.055542, floor=7B27C59A75713E1F9D9B965F57497CA2, algo=ALGORITHM_ESTIMATION

2016-02-19 10:41:53.215 PST [pool-1-thread-2] TRACE c.a.a.l.handlers.BasicMatrixHandler - Station 185e0f901573 reported by unknown BSSID 6cf37feb690

2016-02-19 10:42:03.498 PST [pool-1-thread-2] DEBUG c.a.a.l.records.StationRecord - Station 185e0f901573 scheduled with 12 RSSIs:[RssiRecord [rssi=-70, tsSecs=1455907819, bssid=aca31e59a0d0, channel=0], RssiRecord [rssi=-39, tsSecs=1455907835, bssid=aca31e5a2750, channel=0], RssiRecord [rssi=-70, tsSecs=1455907835, bssid=aca31e59a0d0, channel=0], RssiRecord [rssi=-83, tsSecs=1455907838, bssid=186472e18bf0, channel=0], RssiRecord [rssi=-83, tsSecs=1455907788, bssid=186472e18bf0, channel=0], RssiRecord [rssi=-58, tsSecs=1455907735, bssid=aca31e59a6f0, channel=0], RssiRecord [rssi=-84, tsSecs=1455907805, bssid=aca31e59c3f0, channel=0], RssiRecord [rssi=-60, tsSecs=1455907827, bssid=aca31e599cb0, channel=0], RssiRecord [rssi=-66, tsSecs=1455907786, bssid=aca31e5a4c30, channel=0], RssiRecord [rssi=-72, tsSecs=1455907832, bssid=aca31e59a2b0, channel=0], RssiRecord [rssi=-72, tsSecs=1455907838, bssid=aca31e59a2b0, channel=0], RssiRecord [rssi=-86, tsSecs=1455907811, bssid=aca31e59a2d0, channel=0]]]

2016-02-19 10:42:03.512 PST [pool-1-thread-2] TRACE c.a.a.l.handlers.BasicMatrixHandler - Station 185e0f901573 reported by unknown BSSID 6cf37feb690

2016-02-19 10:42:03.536 PST [engine-worker-5] DEBUG c.a.a.l.records.StationRecord - New location for station 185e0f901573: x=72.157089, y=74.203644, floor=7B27C59A75713E1F9D9B965F57497CA2, algo=ALGORITHM_ESTIMATION

2016-02-19 10:42:03.625 PST [pool-1-thread-2] TRACE c.a.a.l.handlers.BasicMatrixHandler - Station 185e0f901573 reported by unknown BSSID 6cf37feb690
```

In the example above, you can notice a couple of things:

1. The client is being detected by a bunch of APs and its location (x,y) is also being computed by ALE.
2. The station is also being reported by an unknown BSSID (indicated by the third, fifth and seventh entries in the log). This means that ALE does not recognize the BSSID that is reporting the client device in question, possibly because the AP has not yet been identified in AirWave or placed on the VisualRF floor. Since ALE does not have any information on this particular AP, it discards any information reported by the AP.

Very Few Location Events on ALE Compared to Device Traffic on the WLAN

Problem: Cannot see as many location updates on ALE as there are associated devices on the WLAN.

There could be multiple reasons for this observation:

- AP/AM coverage is not sufficient and ALE cannot compute location based on the number of received RSSIs. In that case, you can enable the 'Single AP' feature that will force ALE to compute a new location on a less regular basis even though there's not a lot of RSSIs. Doing so ensures that clients that cannot be heard by 3 or more APs (for triangulation purposes) will be tied to the x,y of the AP that hears the client the best. It is recommended that 'Single AP' mode be always turned on. This allows the algorithm to be less strict on the criteria of using a minimum number of APs reporting the RSSI and other such factors, such as whether the AP reporting the RSSI was learned during the AP-AP RSSI tuning phase. It will generate the location that may be less precise but more frequent.
- Also note that client devices that are associated to the WLAN may not probe as frequently as other unassociated devices. To add to this behavior, for associated devices that are idle, there may not be enough data frames that are being exchanged over the air. Due to these reasons, ALE may not receive enough client RSSIs to compute location. Adding AM density helps in the case of associated devices.
- Ensure that NTP is configured on the controller/IAP as well as on ALE. A difference in timestamps between ALE and the WLAN may cause ALE to treat incoming data from the WLAN as stale and potentially discard it.
- Your system might be running low in CPU/Memory and ALE cannot store any more locations in the internal DB.
- If you are comparing locations shown in AirWave VisualRF with locations shown on the ALE map, keep in mind that the VisualRF map shows locations that have been historically recorded. The ALE map represents a recent snapshot and shows locations of devices as seen over the last few minutes. So if VisualRF shows a higher number of locations, it is possible that some of those are old ones.

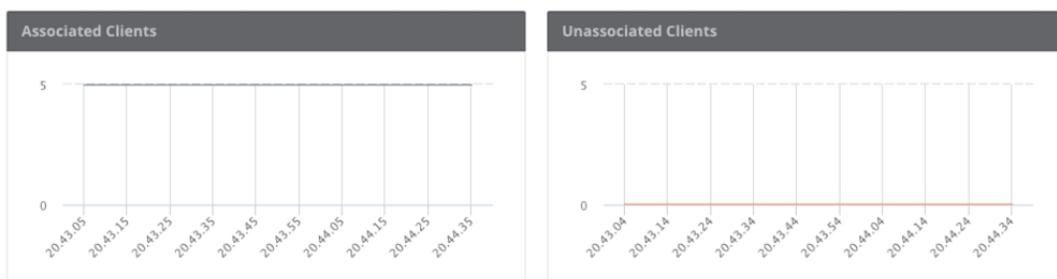
In low AP-density networks, you can also consider switching to Context-only mode that gives you a proximity message rather than an actual location. It tells you that you are in the proximity of a particular AP.

No Presence/Location Events for Unassociated devices on the WLAN

Problem: Controller source on ALE is configured as RTLS. ALE is generating data for associated client devices but no data is observed for unassociated devices.

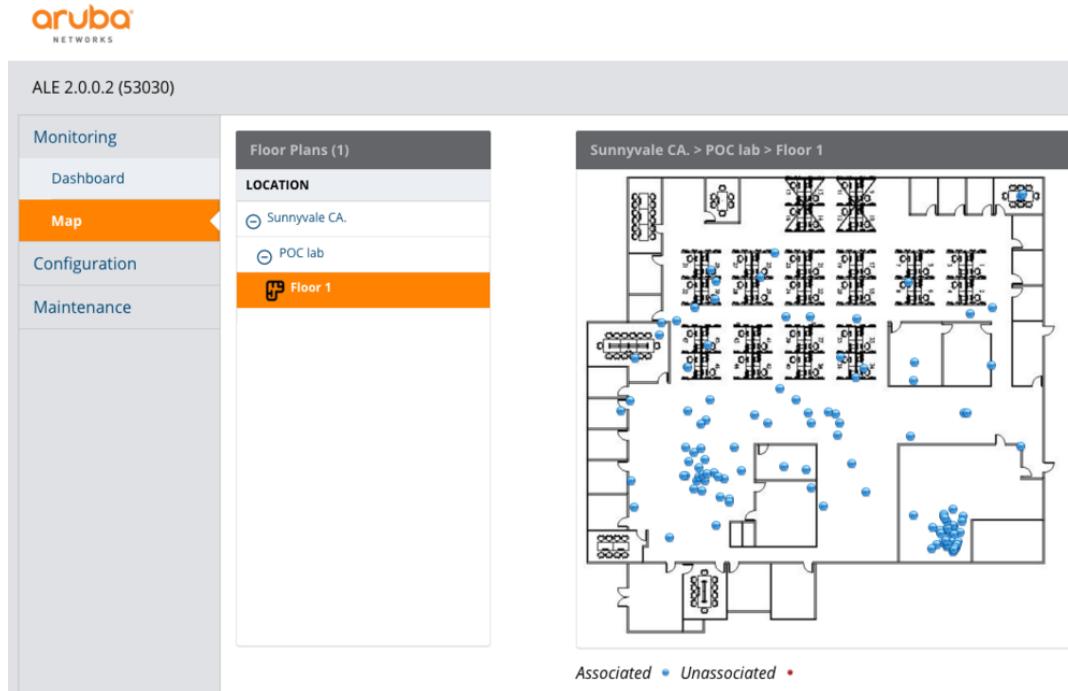
- ALE does not generate presence/location data for unassociated devices.
- ALE UI dashboard does not show unassociated devices.

Figure 23 No Data Observed for Unassociated Devices



- ALE dashboard map does not show red dots (unassociated devices).

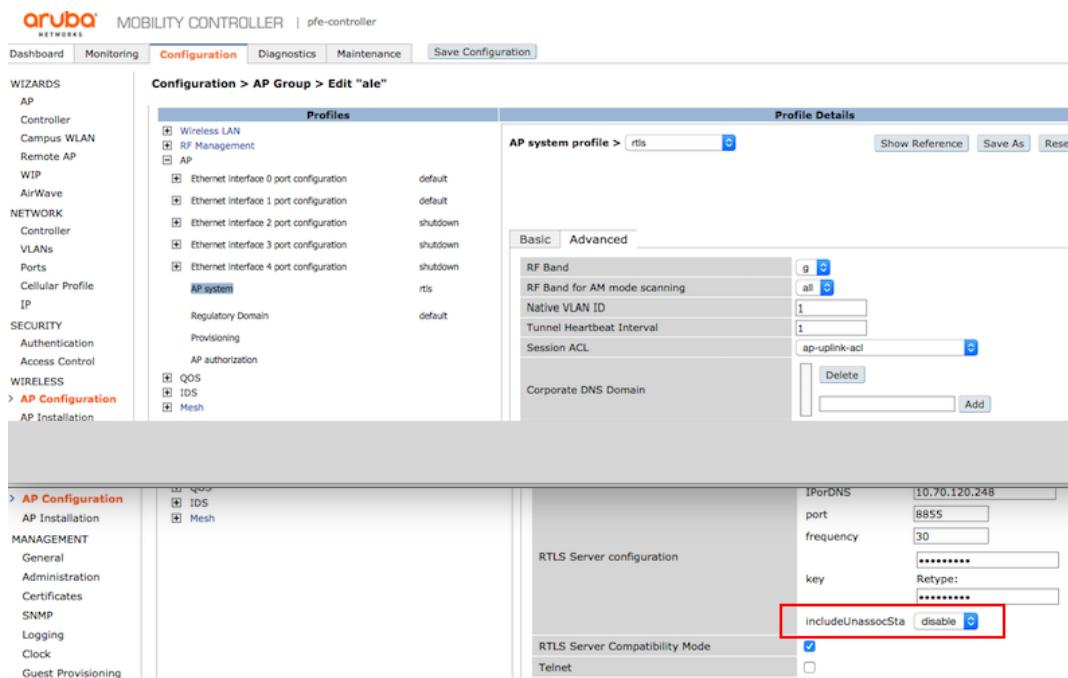
Figure 24 No Unassociated Devices



Check 1: Is the **includeUnassocSta** knob enabled on the controller?

- Under Configuration > Select AP Group > Select AP System Profile > Advanced tab > RTLS Server Configuration:

Figure 25 *includeUnassocSta* Knob



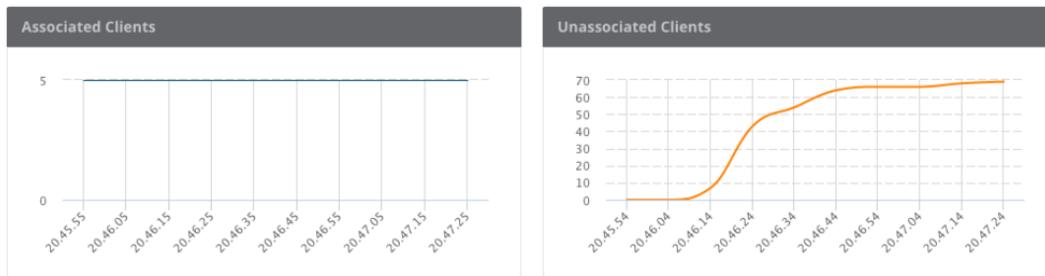
- Enable the **includeUnassocSta** knob and click on **Apply**.

Figure 26 Enable includeUnassocSta Knob

- Verify the presence of unassociated devices on the ALE dashboard.

On the ALE UI > Monitoring:

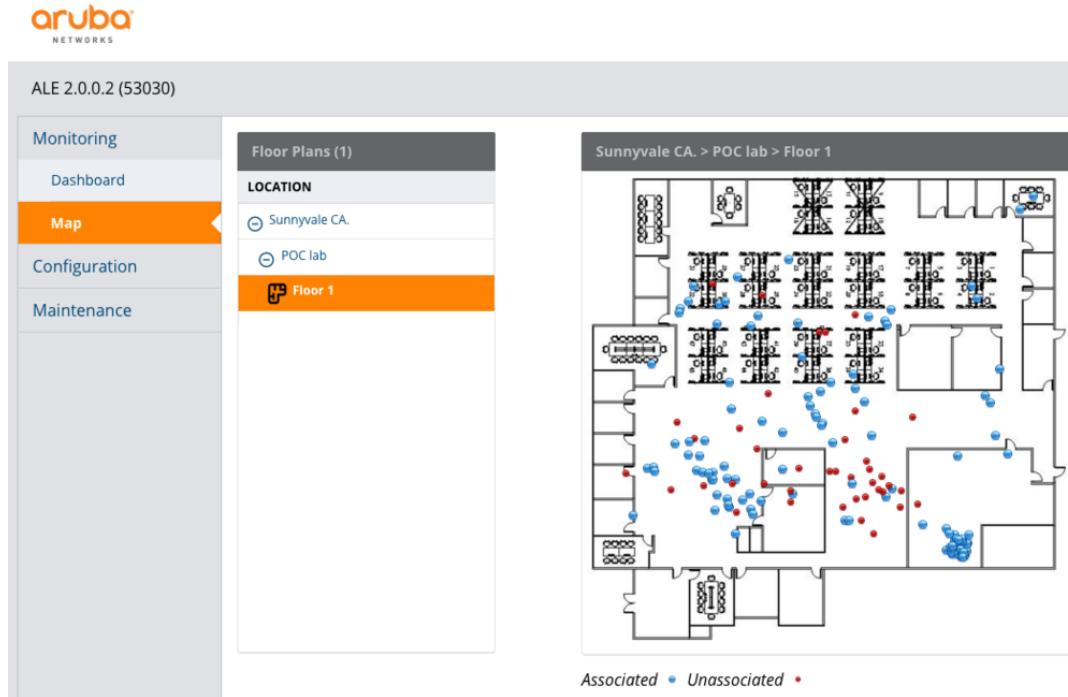
Figure 27 Unassociated Devices Present



- The same can be verified on the ALE Maps dashboard.

On the ALE UI > Monitoring > Map:

Figure 28 ALE Maps Dashboard



- You can also check the output of the presence API to confirm you can see unassociated devices.

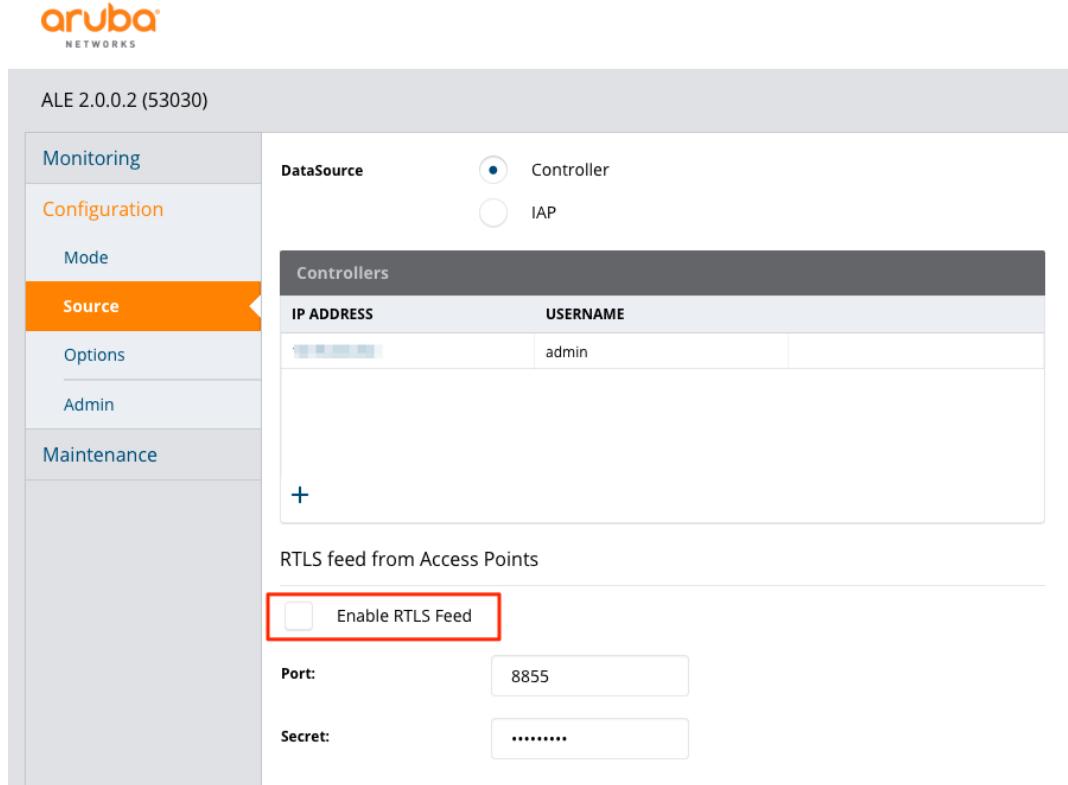
The value for the **associated** field will be **false** for unassociated devices.

```
[root@ale ~]# /opt/ale/bin/feed-reader -f presence
Attempting to 'connect' to endpoint: tcp://localhost:7779
Connected to endpoint: tcp://localhost:7779
Subscribed to topic: "presence"

[1] Recv event with topic "presence"
seq: 9288
timestamp: 1453927575
op: OP_ADD
topic_seq: 172
source_id: 005056852F31
presence {
    sta_eth_mac {
        addr: 34:68:95:ec:f0:db
    }
    associated: false
    hashed_sto_eth_mac: 046AD455D891E4ED49E9FA2285383B6B8608EE6F
    ap_name: POC-ALE-05-83:3c
    radio_mac {
        addr: 18:64:72:e8:33:d0
    }
}
```

Check 2: Alternatively, consider using AMON as the data source on ALE instead of RTLS (recommended) and verify whether you can see events for unassociated client devices.

Figure 29 Enabling AMON



ALE-AirWave Issues

This section includes the following topics:

- [ALE UI Error: "Error Fetching Sites from AirWave" on page 58](#)
- [Client Devices Being Incorrectly Located on the ALE Map on page 61](#)
- [Troubleshooting AP Placement Inconsistencies Resulting in Inadequate/Incorrect Location Events on ALE on page 68](#)

ALE UI Error: "Error Fetching Sites from AirWave"

An error is seen when trying to connect ALE to AirWave.

Under Configuration > Mode > Add AirWave Server:

Figure 30 Adding AirWave

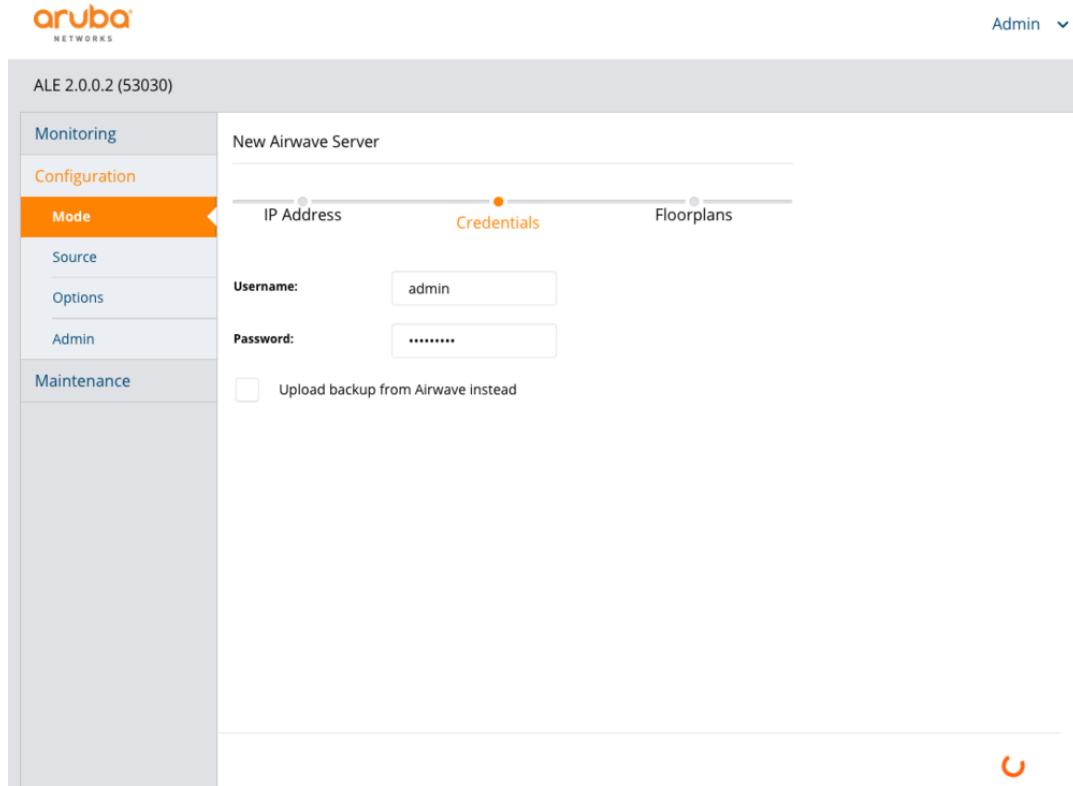
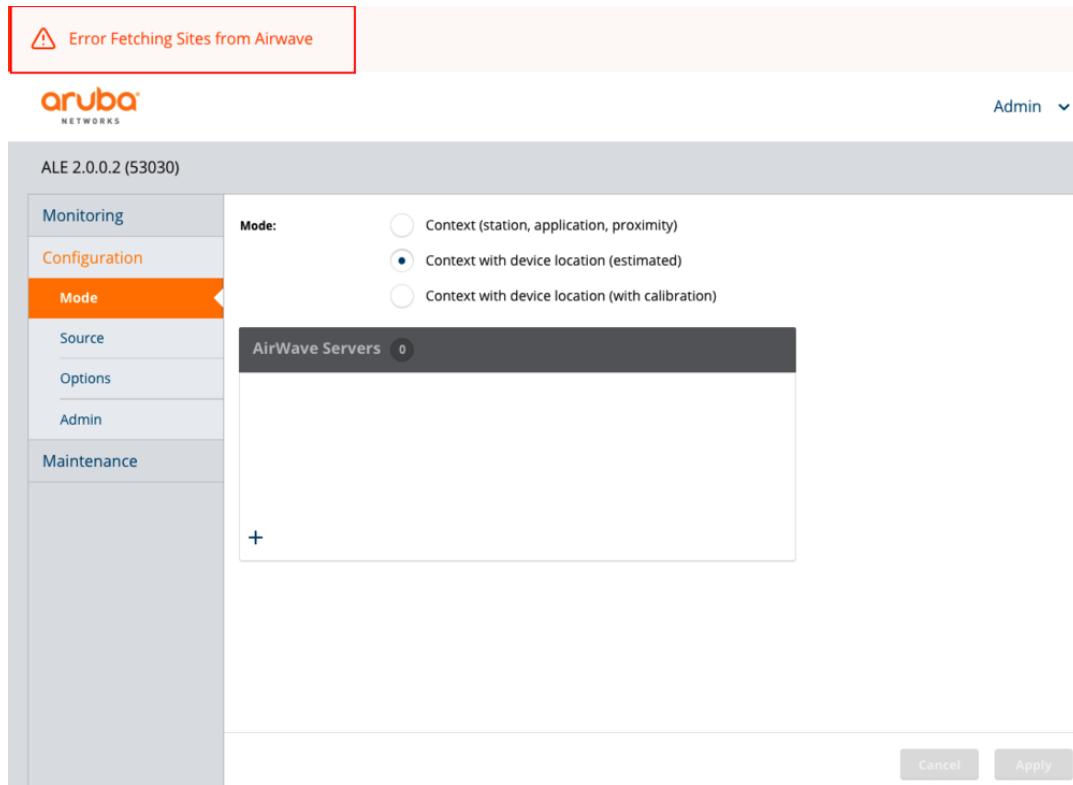


Figure 31 Error Fetching Sites from AirWave



Check 1: Are there any communication issues between ALE and AirWave?

Try performing a Telnet connection to the AirWave server on port 443 (the port that ALE uses to communicate with AirWave).

If you see this:

```
[root@ale ~]# telnet 10.70.120.251 443
Trying 10.70.120.251...
telnet: connect to address 10.70.120.251: No route to host
[root@ale ~]#
```

Or this:

```
[root@ale ~]# telnet 10.70.120.251 443
Trying 10.70.120.251...
telnet: connect to address 10.70.120.251: Connection refused
[root@ale ~]#
```

The above outputs indicate that ALE cannot communicate with AirWave either due to a routing issue, or there is a firewall on the network that is blocking connections to AirWave. Check for end-to-end network connectivity and also the firewall settings in your network to make sure AirWave is reachable.

You can also use a port scanning utility such as Nmap to further investigate firewall issues. Nmap is not installed on ALE by default.

Here's an example:

```
[root@ale ~]# nmap -p 22,443 10.70.120.251

Starting Nmap 5.51 ( http://nmap.org ) at 2016-01-28 21:07 UTC
Nmap scan report for 10.70.120.251
Host is up (0.00050s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
443/tcp   open  https
MAC Address: 00:50:56:85:7D:21 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.51 seconds
[root@ale ~]#
```

Check 2: Is the VisualRF engine running on AirWave?

Figure 32 Enabling VisualRF Engine

Server Settings

Enable VisualRF Engine:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Enable HTML5-based UI:	<input checked="" type="radio"/> Yes <input type="radio"/> No
Enable Multi-floor Bleed Through:	<input checked="" type="radio"/> Yes <input type="radio"/> No
Dynamic Attenuation:	<input checked="" type="radio"/> Yes <input type="radio"/> No
Use Metric Units:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Memory Allocation:	768 MB
Core Threads:	6
Location Caching Threads:	4
UI Threads:	4
Synchronization Timer:	10 minutes
Restrict visibility of empty floor plans to the role of the user who created them:	<input type="radio"/> Yes <input checked="" type="radio"/> No

Wall Attenuation Settings

Add	New Wall Attenuation		
<input type="checkbox"/>	Concrete	15	Red
<input type="checkbox"/>	Cubicle	4	Green
<input type="checkbox"/>	Drywall	6	Yellow
<input type="checkbox"/>	Glass	2	Blue

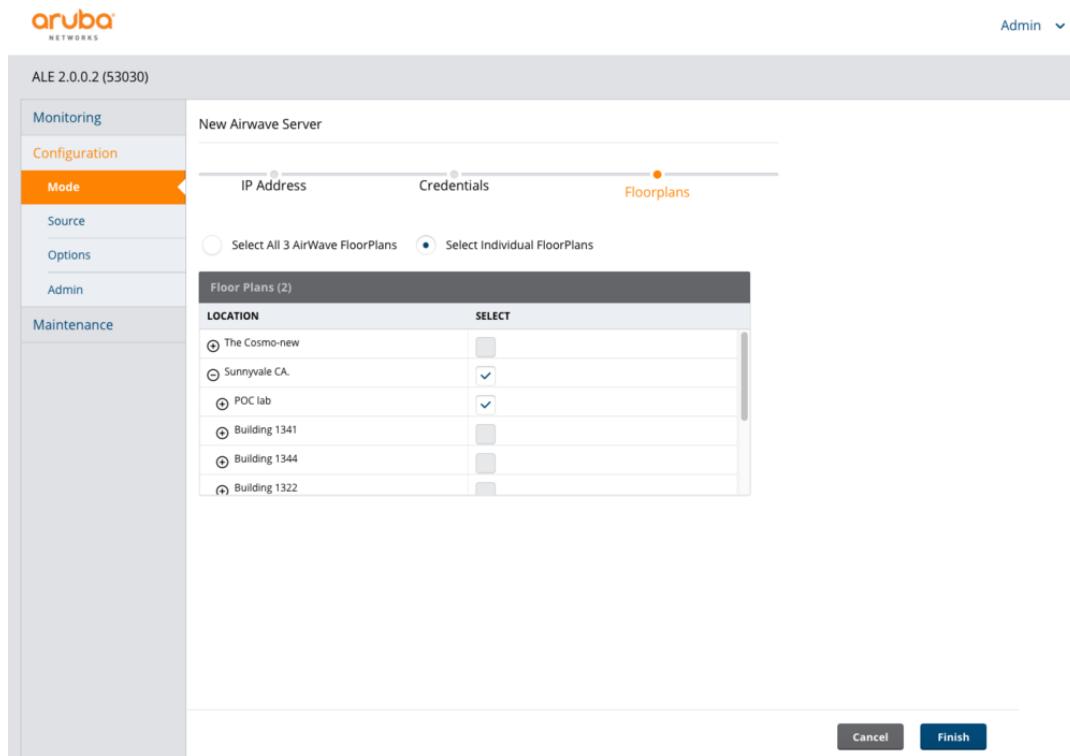
4 Wall Attenuations
Select All - Unselect All
Delete

Save Revert

Enable VisualIRF. Under **VisualIRF > Setup > Enable VisualIRF Engine** > select **Yes**. Click on **Save**.

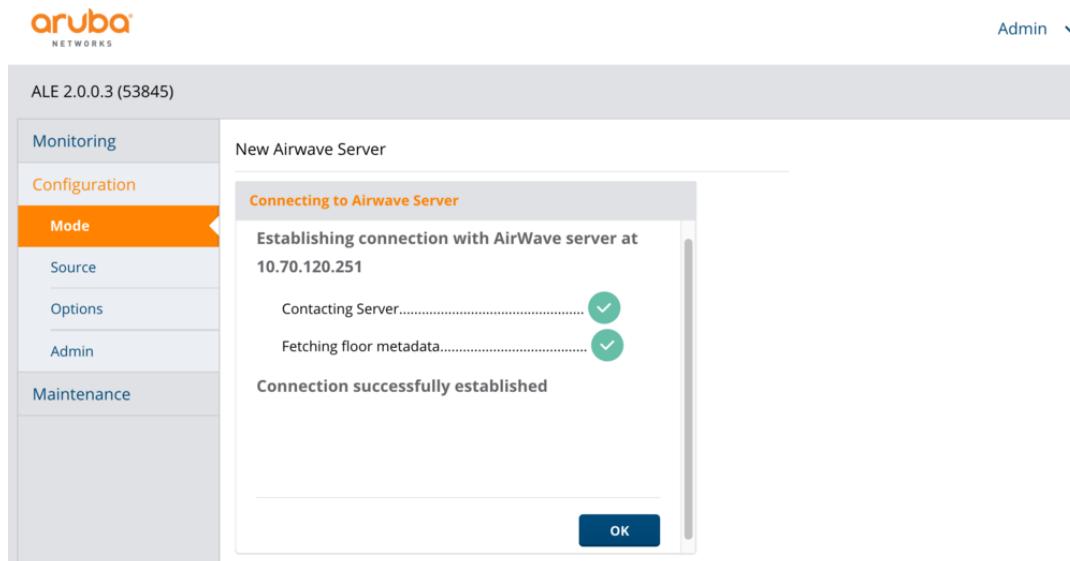
After a few minutes, try adding the AirWave IP on ALE: Under **ALE Configuration > Mode > Add AirWave server IP** > select floors.

Figure 33 Selecting AirWave Floors on ALE



The following output indicates that ALE was able to communicate with AirWave successfully.

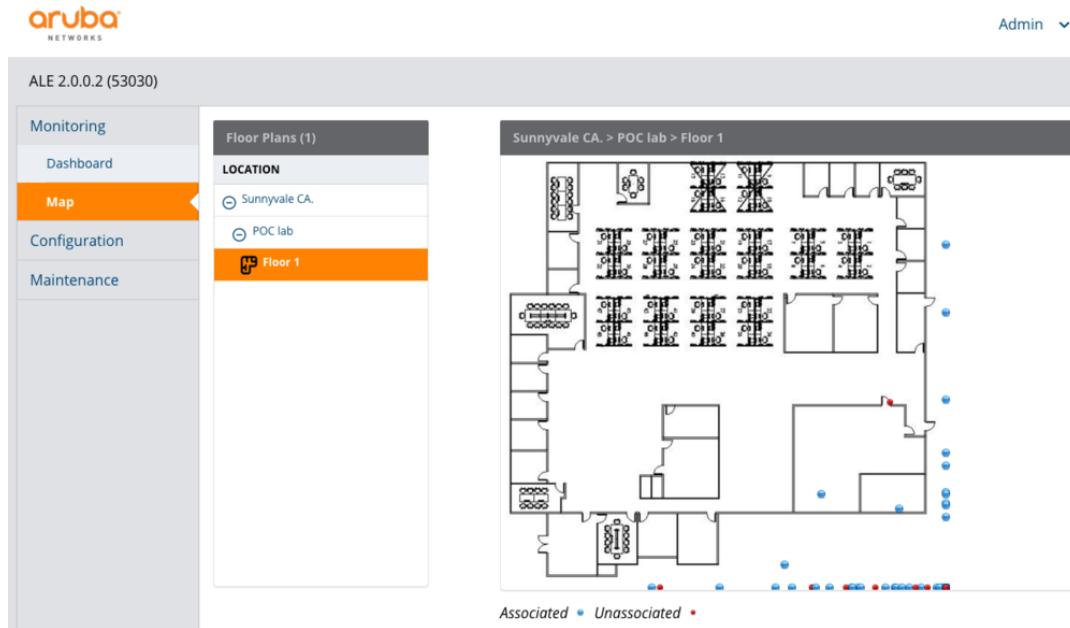
Figure 34 ALE to AirWave Connection Successful



Client Devices Being Incorrectly Located on the ALE Map

Problem: ALE shows clients being incorrectly placed on the floor map.

Figure 35 Dots Incorrectly Laid Out



Check 1: Are you running version 2.0.0.3 or newer? With ALE versions older than 2.0.0.3, there is an issue where configuring different metrics on ALE and AirWave results in clients being placed on the bottom and sides of the floor. The workaround was to use only "feet" metrics on AirWave and ALE but this issue is resolved in ALE 2.0.0.3.

```
[root@ale2 ~]# rpm -qa |grep ale
ale-persistence-2.0.0.2-53030.x86_64
ale-location-2.0.0.2-53030.x86_64
ale-utils-2.0.0.2-53030.x86_64
ale-monitconfig-2.0.0.2-53030.noarch
ale-jwebapp-2.0.0.2-53030.x86_64
ale-platform-2.0.0.2-53030.x86_64
ale-wstunnel-2.0.0.2-53030.x86_64
[root@ale2 ~]#
```

If you are running an older version of ALE, upgrade to 2.0.0.3 or newer:

```
[root@ale-2 ~]# sh ale-2.0.0.3-53845.run
```

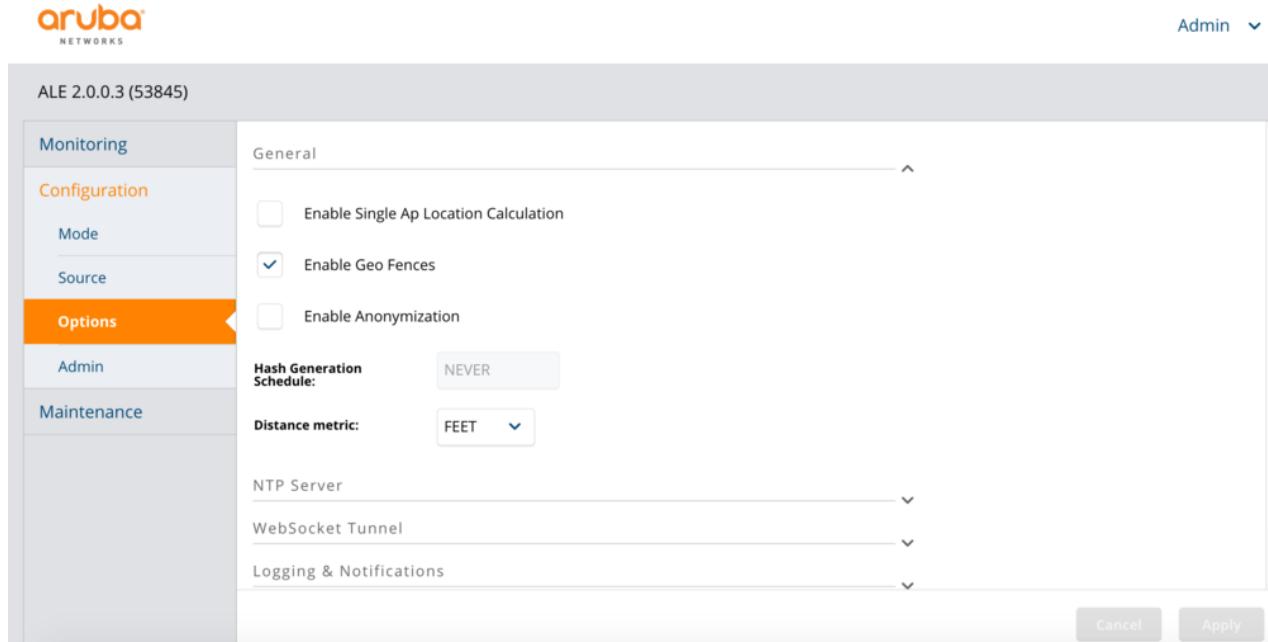
The upgrade may take a few minutes to complete.

Check 2: Were the distance metrics recently changed on AirWave?

It is okay to have different metrics configured on AirWave and ALE, but if the metric unit was recently changed on AirWave, this may cause locations to be incorrectly computed on ALE, as shown above. This issue can be resolved by restarting the VisualRF Engine, re-adding the AirWave entry on ALE, and regenerating the PDB.

Under **ALE Configuration > Options > General >** Is the distance metric configured in FEET or METER?

Figure 36 ALE Distance Metric



- Now check the distance metric configured on AirWave.

Under VisualRF > Setup > Use Metric Units:

Figure 37 Use Metric Units

The screenshot shows the Aruba VisualRF setup interface. The top navigation bar includes New Devices: 21, Up: 19, Down: 44, Mismatched: 29, Rogue: 9, Clients: 4, Alerts: 0, and a Search bar. The main tabs are Home, Groups, APs/Devices, Clients, Reports, System, Device Setup, AMP Setup, RAPIDS, and VisualRF (which is selected). Below these are sub-tabs: Floor Plans, Setup (which is selected), Import, and Audit Log. The 'Server Settings' tab contains various configuration options like Enable VisualRF Engine, Enable HTML5-based UI, and Dynamic Attenuation. The 'Use Metric Units' checkbox is highlighted with a red border. The 'Wall Attenuation Settings' tab shows a table of materials and their attenuation values: Concrete (15, Red), Cubicle (4, Green), Drywall (6, Yellow), and Glass (2, Blue). Buttons for Add, Delete, Save, and Revert are at the bottom.

- In the above example, ALE is configured in FEET whereas AirWave is configured in METERS. If AirWave was recently configured to use different metric units from ALE, this is likely throwing off location calculation and causing clients to be wrongly displayed on the map.
- Toggle the VisualRF engine.

Under VisualRF > Setup > Select **Enable VisualRF Engine** as **No** > Click on **Save**.

Figure 38 Set Enable VisualRF Engine to No

Material	Attenuation	Color
Concrete	15	Red
Cubicle	4	Green
Drywall	6	Yellow
Glass	2	Blue

- Now enable the VisualRF engine.

Under **VisualRF > Setup > Select Enable VisualRF Engine as Yes > Click on Save.**

Figure 39 Set Enable VisualRF Engine to Yes

Material	Attenuation	Color
Concrete	15	Red
Cubicle	4	Green
Drywall	6	Yellow
Glass	2	Blue

It might take a few minutes for the VisualRF Engine to start, so you will not be able to access the VisualRF page during this time.

- Re-add the AirWave entry on ALE.

Under **ALE Configuration > Mode > Click on the delete icon > Add AirWave back > Select the same floor and complete the site-fetch process.**

Figure 40 Delete AirWave Entry on ALE

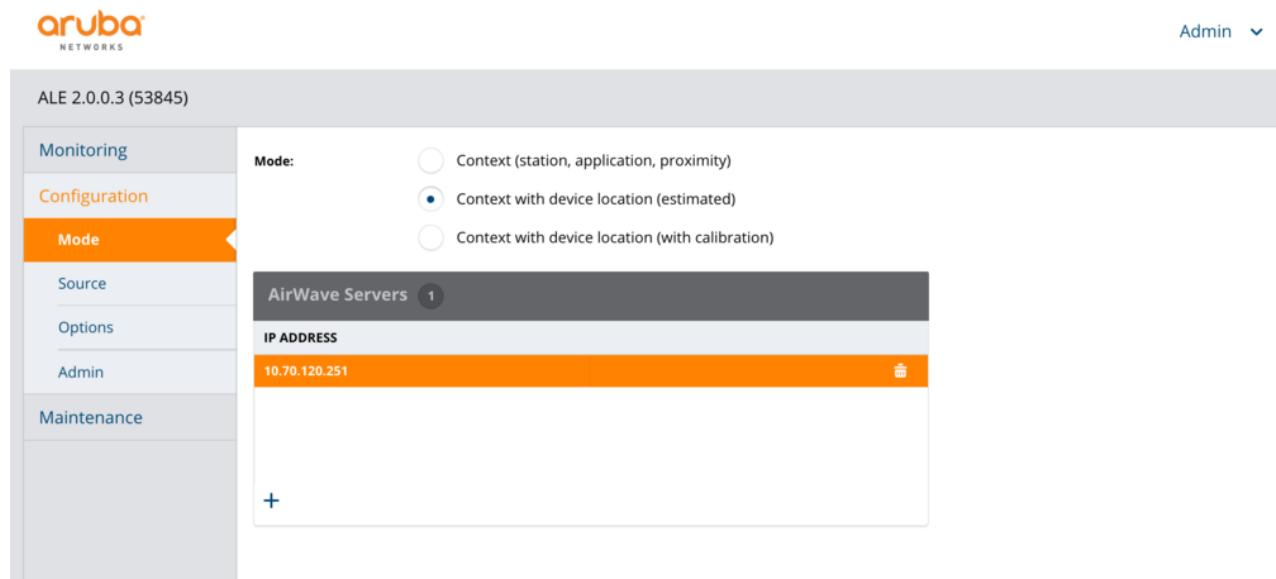


Figure 41 Adding an AirWave Entry on ALE.

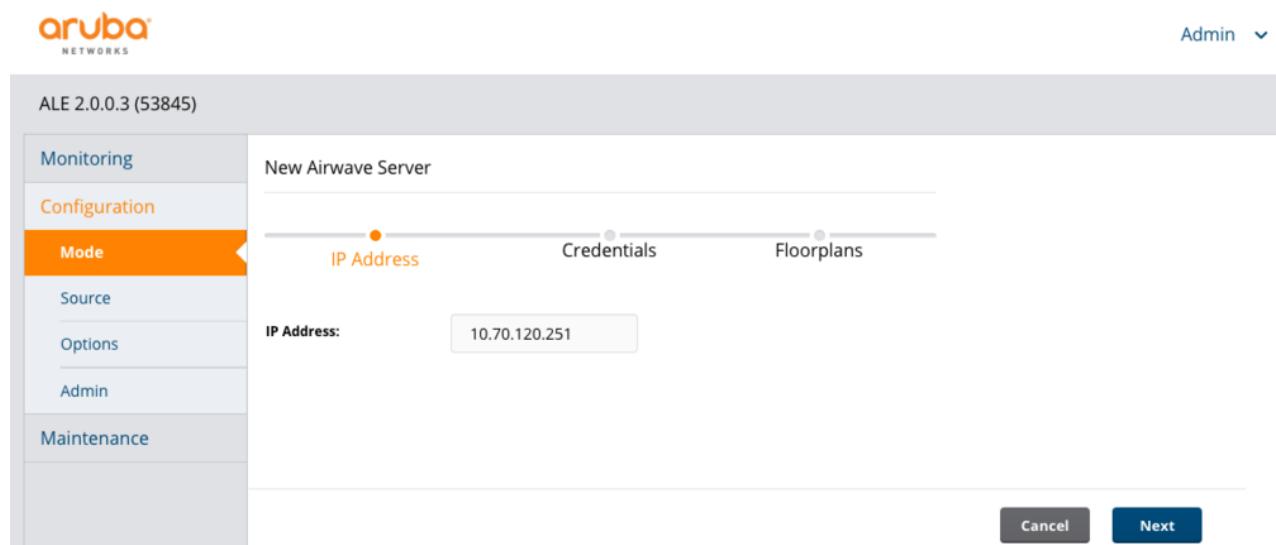


Figure 42 Specifying AirWave Credentials on ALE

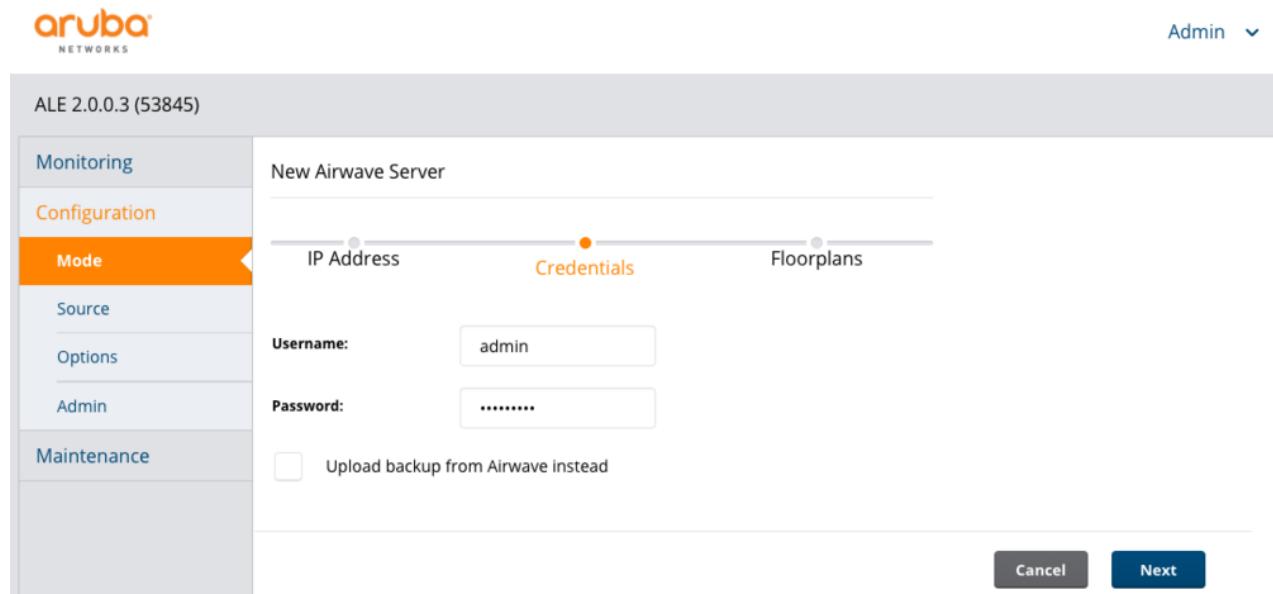


Figure 43 Selecting Floors from AirWave

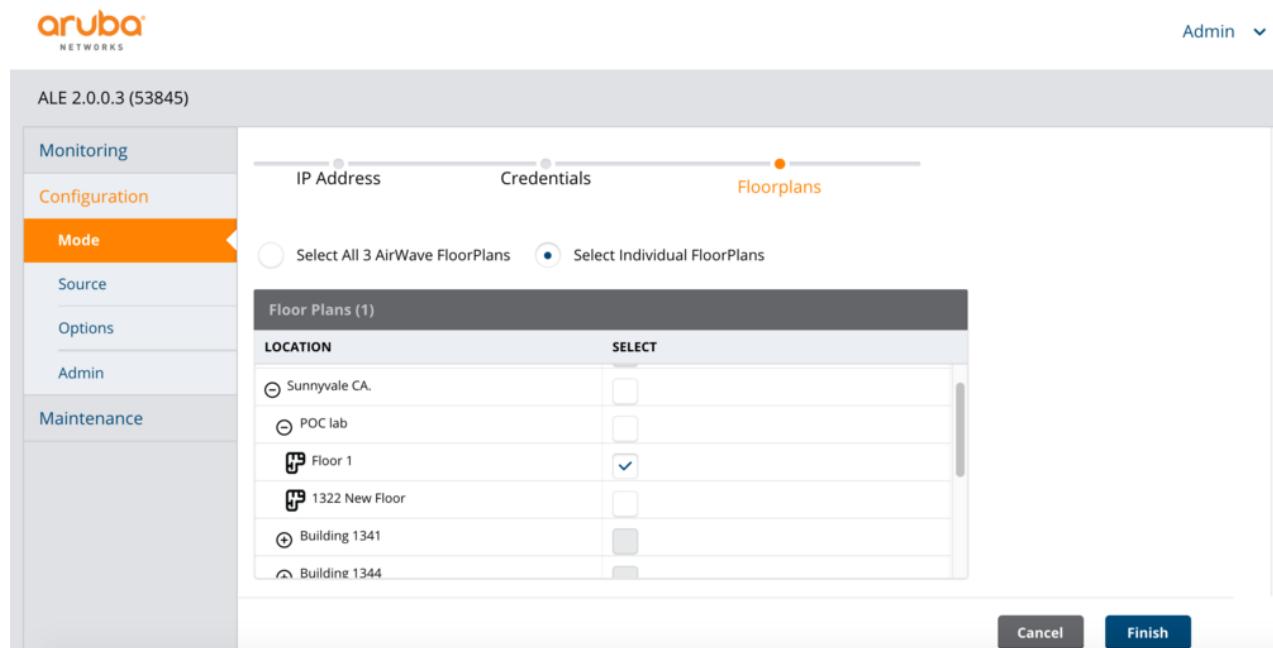
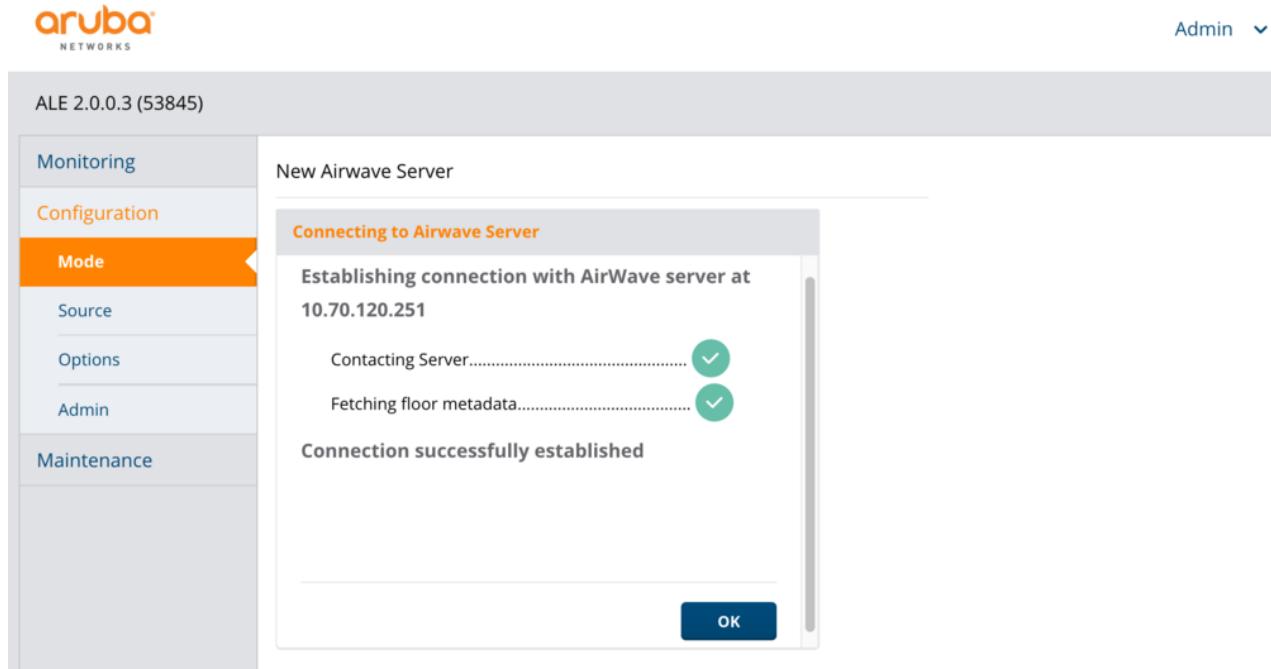


Figure 44 ALE to AirWave Connection Successful



- Regenerate the PDB.

Under ALE Maintenance > Regenerate PDB > Generate Database Now.

Figure 45 Regenerating PDB

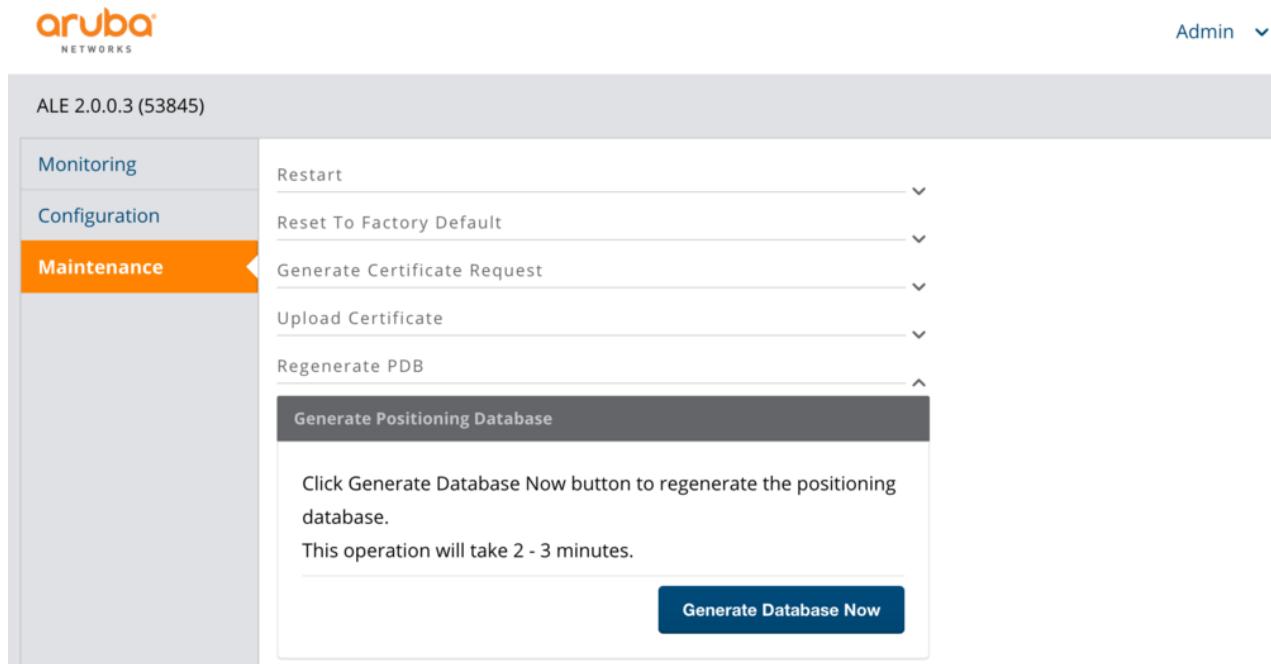
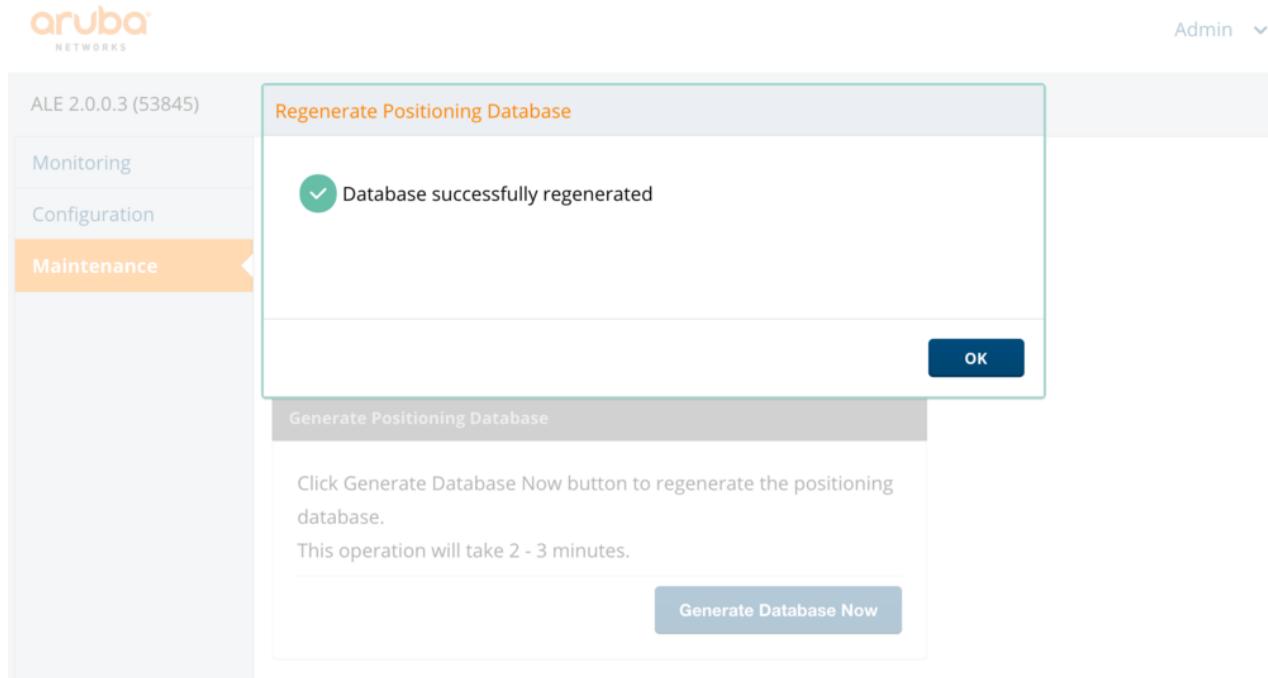


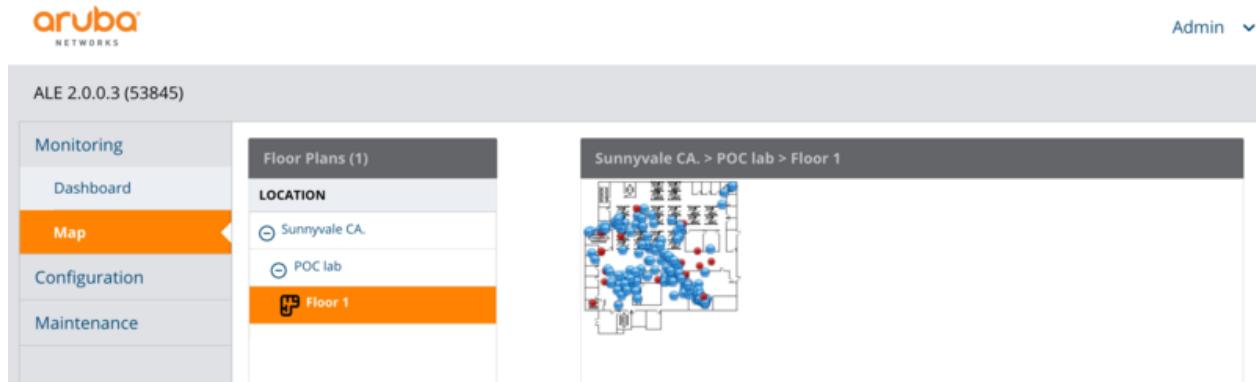
Figure 46 Database Successfully Generated



- Verify that the ALE map now displays client locations correctly.

Under **ALE > Monitor > Map > Select the same floor.**

Figure 47 ALE Map



Troubleshooting AP Placement Inconsistencies Resulting in Inadequate/Incorrect Location Events on ALE

Problem: How can I verify if the AP placement on AirWave is consistent with the APs that are sending RSSI feeds to ALE?

It is crucial to ensure that APs are correctly placed in AirWave, because if ALE receives RSSI information from an AP that it hasn't learnt about from AirWave, the location engine will ignore these APs when computing client locations.

Look at the following logs to understand if this is a problem in your deployment:

- The following counters in the **ale-location** log (INFO level) indicate that ALE is receiving RSSI information from unknown BSSIDs:

```

2016-02-19 12:03:58.783 PST [metrics-logger-reporter-1-thread-1] INFO
c.a.a.c.metrics.RegistryManager - type=COUNTER,
name=com.aruba.ale.location.handlers.BasicMatrixHandler.rtls.bssidNotFound, count=0

2016-02-19 12:03:58.783 PST [metrics-logger-reporter-1-thread-1] INFO
c.a.a.c.metrics.RegistryManager - type=COUNTER,
name=com.aruba.ale.location.handlers.BasicMatrixHandler.vbr.bssidNotFound, count=810305

```

The above log indicates that there is a high incidence of AMON-based RSSI coming from APs that the location engine has not learned from AirWave.

- When **ale-location** logs are set to TRACE level, the following log entry indicates this concern:

```

2016-02-19 10:42:03.625 PST [pool-1-thread-2] TRACE c.a.a.l.handlers.BasicMatrixHandler -
Station 185e0f901573 reported by unknown BSSID 6cf37febfb690

```

Corrective action would be to add this AP on AirWave, if not already present.

WebSocket Connection Issues

ALE Cannot Initiate a Connection to the WebSocket Server

Problem: ALE cannot establish the WebSocket connection.

This issue can be noticed:

- From the ALE UI dashboard:

Figure 48 *WebSocket Connection Failed*

The screenshot shows the ALE 2.0.0.2 (53030) dashboard. The left sidebar has tabs for Monitoring, Dashboard (which is selected), Map, Configuration, and Maintenance. The main area is divided into three sections: Health, Info, and License. In the Health section, under Network Data, there is a yellow warning icon next to the text "WebSocket connect failed". A tooltip over this icon provides more details: "WebSocket connect failed 08:31 PM 2016 UTC to: 10.70.120.234". The Info section shows Mode: Context + Location (Estimated), Campuses: 1, Buildings: 1, Floors: 1, and APs: 13. The License section shows License Type: PERMANENT. The top right corner shows the user is Admin.

- Via ALE logs:

```
[root@ale bin]# tailf /opt/ale/ale-wstunnel/logs/ale-wstunnel.log
```

```
2016-01-29 20:29:01.915 UTC [main] INFO com.aruba.ale.wstunnel.TunnelClient - Connecting with client ID of 005056852F31
2016-01-29 20:29:02.288 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.client.TunnelConnector - WebSocket connecting to wss://10.70.120.234:4433
2016-01-29 20:29:02.957 UTC [vert.x-eventloop-thread-0] ERROR c.a.a.w.client.TunnelConnector - WebSocket connection to 10.70.120.234:4433 failed. Will retry in 0 seconds...
2016-01-29 20:29:02.962 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.client.TunnelConnector - WebSocket connecting to wss://10.70.120.234:4433
2016-01-29 20:29:02.984 UTC [vert.x-eventloop-thread-0] ERROR c.a.a.w.client.TunnelConnector - WebSocket connection to 10.70.120.234:4433 failed. Will retry in 1 seconds...
2016-01-29 20:29:03.986 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.client.TunnelConnector - WebSocket connecting to wss://10.70.120.234:4433
```

- Via the logs on the WebSocket server. The output of this log file will show no incoming connections from ALE.

```
[root@websocket-server ale-wstunnel]# tailf logs/ale-wstunnel.log
```

Check 1: Is the WebSocket server (FQDN) reachable via a ping test? If the server's FQDN is not resolvable on the network, add it manually on ALE.

```
[root@ale ~]# vi /etc/hosts
```

```
127.0.0.1      localhost.localdomain  localhost.localdomain
localhost4      localhost4.localdomain4  localhost        ale-2
```

```
1      localhost.localdomain  localhost.localdomain  localhost6
localhost6.localdomain6  localhost        ale-2
```

```
10.70.120.234 tme-websocket.arubanetworks.com
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
-- INSERT --                                     <<<----- Type 'i' to edit. To save and
quit, type Esc -> :wq -> Enter
```

Check 2: Is the WebSocket server listening for incoming ALE requests on TCP port 4433 (default port is TCP 443)?

You can also use a port scanning utility such as Nmap to further investigate firewall issues. Nmap is not installed on ALE by default.

```
[root@ale ~]# yum install nmap
```

Here's an example of Nmap usage:

```
[root@ale ~]# nmap -p 4433 tme-websocket.arubanetworks.com

Starting Nmap 5.51 ( http://nmap.org ) at 2016-01-30 00:18 UTC
Nmap scan report for tme-websocket.arubanetworks.com (10.70.120.234)
Host is up (0.00047s latency).

PORT      STATE SERVICE
4433/tcp  open  unknown
MAC Address: 00:50:56:85:5D:0A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
[root@ale ~]#
```

- If the port state is **filtered**, there is a firewall on the network or on the WebSocket server that is blocking the connection requests.
- If the port state is **closed**, there is no application on the other end that is listening on port 4433. Make sure you start the WebSocket service on the other end.
- If the port state is **open**, there is an application that is listening on port 4433. This is the desired port state.

Check 3: Verify that the WebSocket server is running:

On the WebSocket server:

```
[root@websocket-server ale-wstunnel]# ps aux |grep Tunnel
root      5509  0.0  0.0 103244   868 pts/0    S+   00:23   0:00 grep Tunnel
[root@websocket-server ale-wstunnel]#
```

The example output above shows that the WebSocket server hasn't been started yet.

To start the service:

```
[root@websocket-server ~]# /opt/ale/ale-wstunnel/bin/startup-server.sh
[root@websocket-server ~]#
```

Verify the service is running (indicated by **com.aruba.ale.wstunnel.TunnelServer** in the output below):

```
[root@websocket-server ~]# ps aux |grep Tunnel
root      5589  1.0  1.5 4065264 91344 ?        S1   00:25   0:02 /usr/java/jdk1.8.0_31/bin/java -cp /opt/ale/ale-wstunnel/conf:/opt/ale/ale-wstunnel/lib/ale-wstunnel-0.0.1.jar -Dlogger_basedir=/opt/ale/ale-wstunnel/logs -Djava.library.path=/opt/ale/lib64 -Dlogger.console=false com.aruba.ale.wstunnel.TunnelClient
root      5634 42.0  1.9 3982820 114080 pts/0    S1   00:29   0:02 /usr/bin/java -cp /opt/ale/ale-wstunnel/conf:/opt/ale/ale-wstunnel/lib/ale-wstunnel-0.0.1.jar -Dlogger_basedir=/opt/ale/ale-wstunnel/logs -Dlogger.console=false com.aruba.ale.wstunnel.TunnelServer
root      5652  0.0  0.0 103244   872 pts/0    S+   00:29   0:00 grep Tunnel
[root@websocket-server ~]#
```

Check 4: Ensure that the FQDN of the remote endpoint specified on ALE matches the Common Name (CN) on the server's certificate.

If the FQDN and CN do not match, then ALE cannot authenticate the WebSocket server and the connection will fail.

Check for the Websocket remote endpoint configured on ALE:

Figure 49 Remote Endpoint as FQDN on ALE

LOCAL ENDPOINT	PORT #
localhost	8080
localhost	7779
localhost	443

REMOTE ENDPOINT URL	PORT #
tme-websocket.arubanetworks.com	4433

Use OpenSSL on ALE to verify the CN on the server certificate:

```
[root@ale ~]# openssl s_client -connect tme-websocket.arubanetworks.com:4433
CONNECTED(00000003)
depth=0 C = US, ST = CA, L = Sunnyvale, O = HPE Aruba, OU = TME, CN = tme-
websocket.arubanetworks.com, emailAddress = tme@hpe.com
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, ST = CA, L = Sunnyvale, O = HPE Aruba, OU = TME, CN = tme-
websocket.arubanetworks.com, emailAddress = tme@hpe.com
verify return:1
---
<output snipped>
```

Check 5: On ALE, is the remote endpoint configured as the WebSocket server's FQDN or the IP address?

It is recommended that the FQDN is specified as the remote endpoint on ALE and not the IP address. If the IP address is specified as the remote endpoint, the connection may fail. This means that the CN on the WebSocket server must also be configured to be the server's FQDN.

Example of a WebSocket remote endpoint incorrectly configured on ALE:

Figure 50 *Remote Endpoint*

The screenshot shows the ALE 2.0.0.2 (53030) interface. On the left, a vertical navigation menu includes options like Monitoring, Configuration, Mode, Source, Options (which is selected and highlighted in orange), Admin, and Maintenance. The main content area is divided into two sections: 'Local End-Points' and 'Remote End-Points'.
Local End-Points: A table with columns 'LOCAL ENDPOINT' and 'PORT #'. It lists three entries: localhost (port 8080), localhost (port 7779), and localhost (port 443).
Remote End-Points: A table with columns 'REMOTE ENDPOINT URL' and 'PORT #'. It lists one entry: 10.70.120.234 (port 443), which is highlighted with a red border.

ALE Troubleshooting Logs

This section includes the following topics:

- [ALE Logs on page 73](#)
- [ALE Logging Levels on page 74](#)
- [WebSocket Logs on page 75](#)
- [Tech Support Logs on page 76](#)

ALE Logs

There are five different logs that you can monitor in order to troubleshoot issues on ALE.

- **Platform logs** (ale-platform): Use these logs to troubleshoot issues between the WLAN and ALE (for example, incoming data).
- **Location logs** (ale-location): Use these logs to troubleshoot issues related to location events, such as troubleshooting specific client devices, APs being ignored during location calculations, etc.
- **Persistence logs** (ale-persistence): Use these logs to troubleshoot northbound API events being published by ALE.
- **Webapp logs** (ale-jwebapp): Use these logs to troubleshoot communication issues between ALE and AirWave.
- **WebSocket logs** (ale-wstunnel): Use these logs to troubleshoot WebSocket communication issues between ALE and external analytics applications.

All five logs are accessible from the `/opt/ale/` file path on ALE.

```
[root@ale-2 ~]# cd /opt/ale/
[root@ale-2 ale]# ls
ale-jwebapp ale-location ale-persistence ale-platform ale-wstunnel bin include lib64
license naocloud noscan
[root@ale-2 ale]#
```

Below is an example command to monitor ale-location logs in real-time:

```
[root@ale-2 ale]# tailf /opt/ale/ale-location/logs/ale-location.log
```

ALE Logging Levels

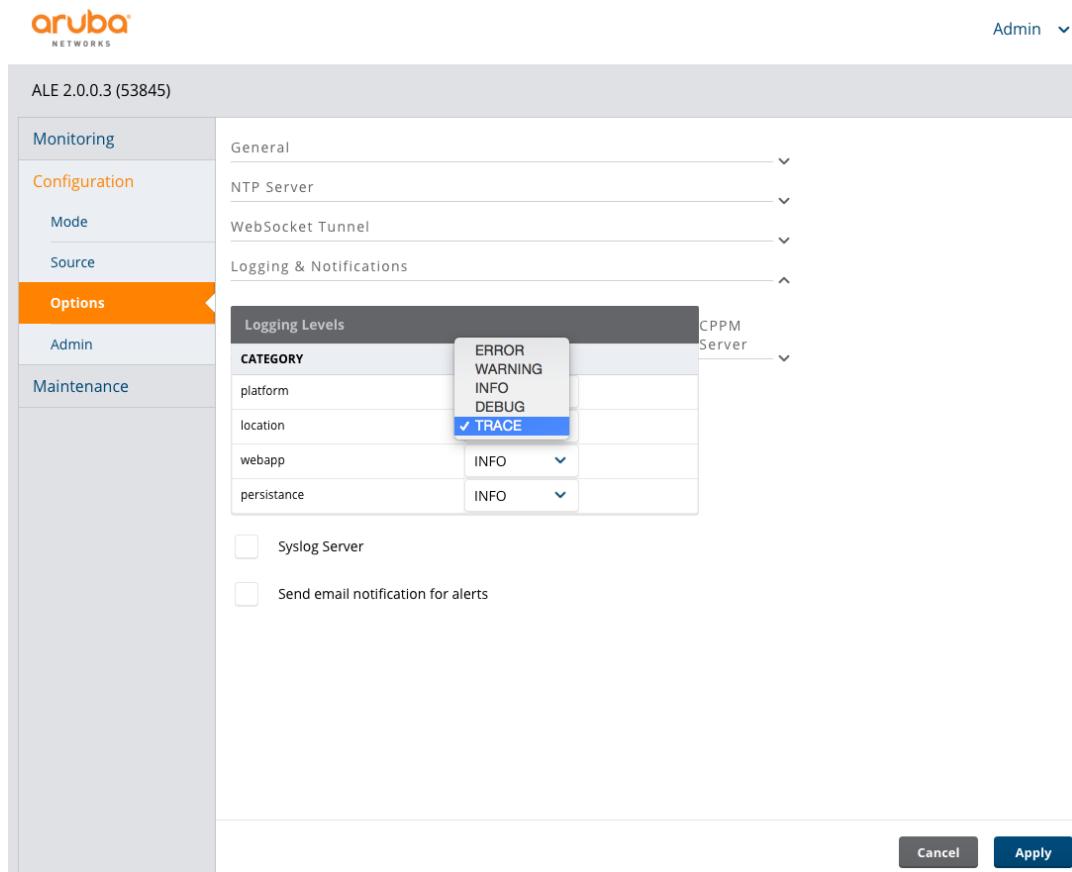
There are five logging levels associated with each of these logs:

- ERROR
- WARNING
- INFO (default for all logs)
- DEBUG
- TRACE

The TRACE logging level enables the most verbose logs, followed by DEBUG and INFO.

Depending on the issue you are troubleshooting, it is recommended to change the logging level on the corresponding logs to TRACE for maximum verbosity.

Figure 51 Enabling TRACE Level on ALE Logs



WebSocket Logs

You can use the **ale-wstunnel** logs to troubleshoot issues with your WebSocket connection.

On the WebSocket Client (ALE)

On ALE, the logs are available in the following path:

```
/opt/ale/ale-wstunnel/logs/ale-wstunnel.log
```

Issue the following command to view the status of the WebSocket connection in real time:

```
[root@ale-2 ~]# tailf /opt/ale/ale-wstunnel/logs/ale-wstunnel.log
```

The following output indicates that a connection was initiated to the WebSocket server at the time indicated by the timestamp:

```
2016-03-10 21:59:00.884 UTC [main] INFO com.aruba.ale.wstunnel.TunnelClient - Connecting with client ID of 005056852F31  
2016-03-10 21:59:02.143 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.client.TunnelConnector - WebSocket connecting to wss://tme-websocket.arubanetworks.com:4433
```

The following log indicates a successful WebSocket connection:

```
2016-03-10 21:59:02.321 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.client.TunnelConnector - WebSocket connected to tme-websocket.arubanetworks.com:4433
```

ALE will publish the status of established WebSocket connections every two minutes, indicated by the following output.

```
2016-03-10 22:01:00.878 UTC [metrics-logger-reporter-1-thread-1] INFO c.a.a.c.metrics.RegistryManager - type=GAUGE, name=com.aruba.ale.wstunnel.client.TunnelConnector.tme-websocket.arubanetworks.com, value=true
```

On the WebSocket Server

On the WebSocket server, the logs are available in the following path:

```
<file-path-where-websocket-package-was-installed>/ale-wstunnel-0.0.1/logs/ale-wstunnel.log
```

Issue the following command to view the status of the WebSocket connection in real time:

```
tailf <file-path>/ale-wstunnel-0.0.1/logs/ale-wstunnel.log
```

The following output indicates a successful WebSocket connection:

```
2016-03-10 21:59:02.280 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.s.WebSocketServerTunnel - Client '005056852F31' connected.  
2016-03-10 21:59:02.280 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12000 -> localhost:8080  
2016-03-10 21:59:02.280 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12001 -> localhost:7779  
2016-03-10 21:59:02.281 UTC [vert.x-eventloop-thread-0] INFO c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12002 -> localhost:443
```

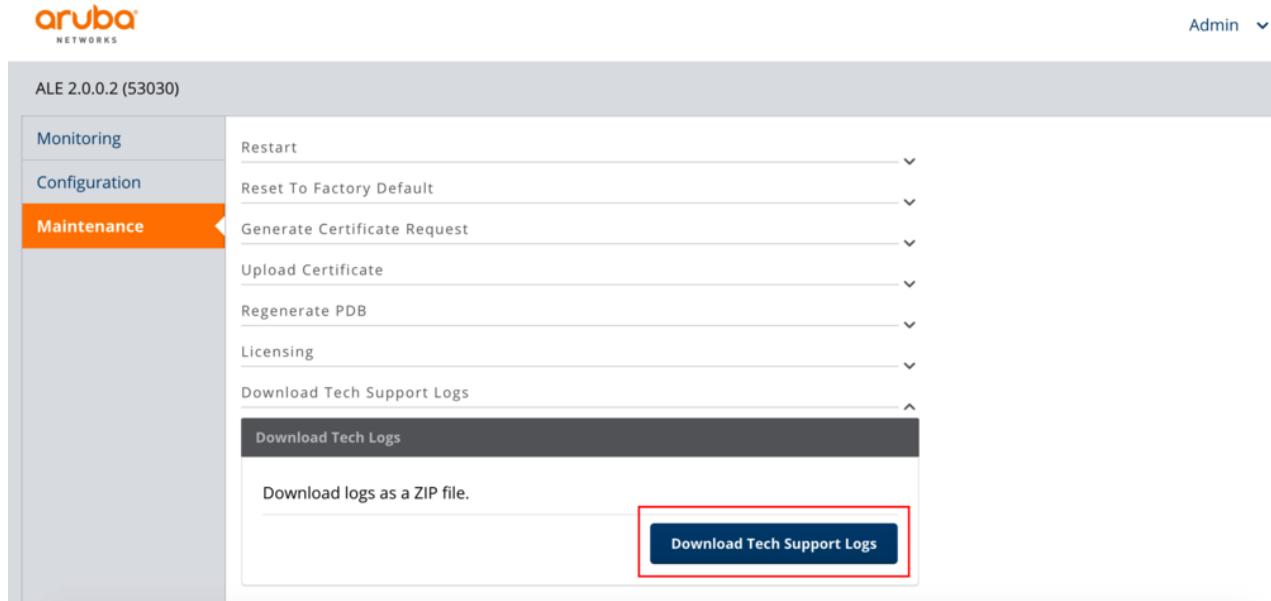
Where 005056852F31 is the ID of the ALE server that initiated the WebSocket connection.

Tech Support Logs

If you have exhausted all troubleshooting methods and the issue still persists, it may help to open a support ticket with HPE Aruba. Please remember to include the ALE Tech Support logs as part of the ticket.

They are available to download from the ALE UI under **Maintenance > Download Tech Support Logs**.

Figure 52 ALE Tech Support Logs



This Appendix includes the following topics:

- [ALE Feed Reader Reference Code \(Java\) on page 77](#)
- [ALE Feed Reader Reference Code \(C++\) on page 77](#)
- [IAP-ALE Bandwidth Calculator on page 77](#)
- [Configuring and Testing a WebSocket Tunnel on page 77](#)
- [Using the ALE Demonstrator Application on page 95](#)

ALE Feed Reader Reference Code (Java)

Contact Support or your local Aruba account team for access to this content.

ALE Feed Reader Reference Code (C++)

Contact Support or your local Aruba account team for access to this content.

IAP-ALE Bandwidth Calculator

Contact Support or your local Aruba account team for access to this content.

Configuring and Testing a WebSocket Tunnel

This section walks you through the process of configuring a WebSocket tunnel between ALE and analytics applications. Please use this section in conjunction with information on WebSocket tunnels documented in the ALE 2.0 User Guide.

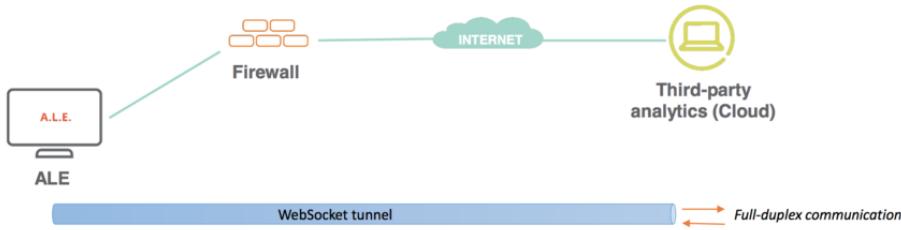
In most deployments, ALE resides behind a firewall and the analytics applications that subscribe to the ALE feed may either be co-located with ALE or deployed in the cloud. If the application server is co-located, then it can subscribe to the ALE feed without the need of a WebSocket connection to ALE.

However, if the customer is using the services of a third-party analytics partner, then the analytics application in all likelihood will reside in the cloud. Because ALE is behind a firewall, the application cannot connect directly to ALE, hence the need of a secure WebSocket connection.

Once ALE initiates a WebSocket connection to the application, the application can now use the REST APIs or the Publish/Subscribe APIs to retrieve contextual data from ALE.

A WebSocket connection is secure since it uses HTTPS. Having a WebSocket connection can be especially beneficial in network environments that block non-web Internet connections.

Figure 53 WebSocket Connections



This section includes the following topics:

- [Ports Used for ALE and WebSocket Server Communication on page 78](#)
- [Configuration Requirements on page 78](#)
- [Configuration Steps on page 78](#)
- [Test the Polling and Publish/Subscribe APIs on page 89](#)

Ports Used for ALE and WebSocket Server Communication

Please ensure that ALE can communicate with the analytics application (WebSocket server) on the port that the application is listening for WebSocket connection requests. By default, TCP 443 is used by the WebSocket software package. This port is configurable, as shown in step 4 of the section [Set the WebSocket Server's Properties on page 82](#).

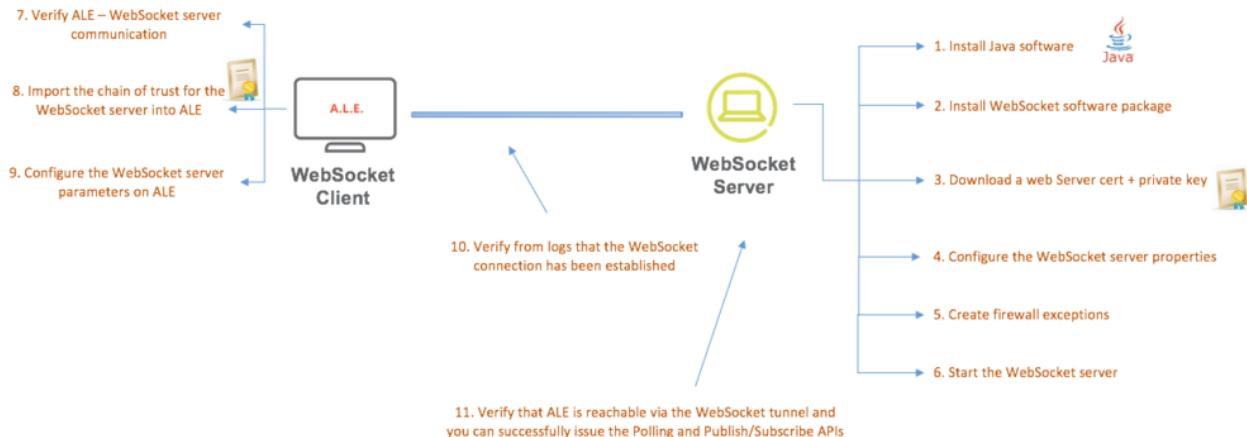


ALE needs an Internet connection; it should be able to perform DNS resolution to be able to successfully establish WebSocket connections with cloud-based analytics applications.

Configuration Requirements

From a terminology perspective, ALE is the WebSocket client (since it is the end-point that will initiate the connection) and the analytics application server will be the WebSocket server.

Figure 54 WebSocket Workflow



Configuration Steps

The following sections describe how to configure a WebSocket connection between ALE and a Linux CentOs-based server. For more details on WebSocket connections, please refer to the ALE 2.0 User Guide.

WebSocket Server

Install Java

1. Check your Java version:

```
[root@websocket-server ~]# java -version  
java version "1.8.0_31"  
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)  
[root@websocket-server ~]# javac -version  
javac 1.8.0_31  
[root@websocket-server ~]#
```

If the latest version is installed, proceed to step 2.

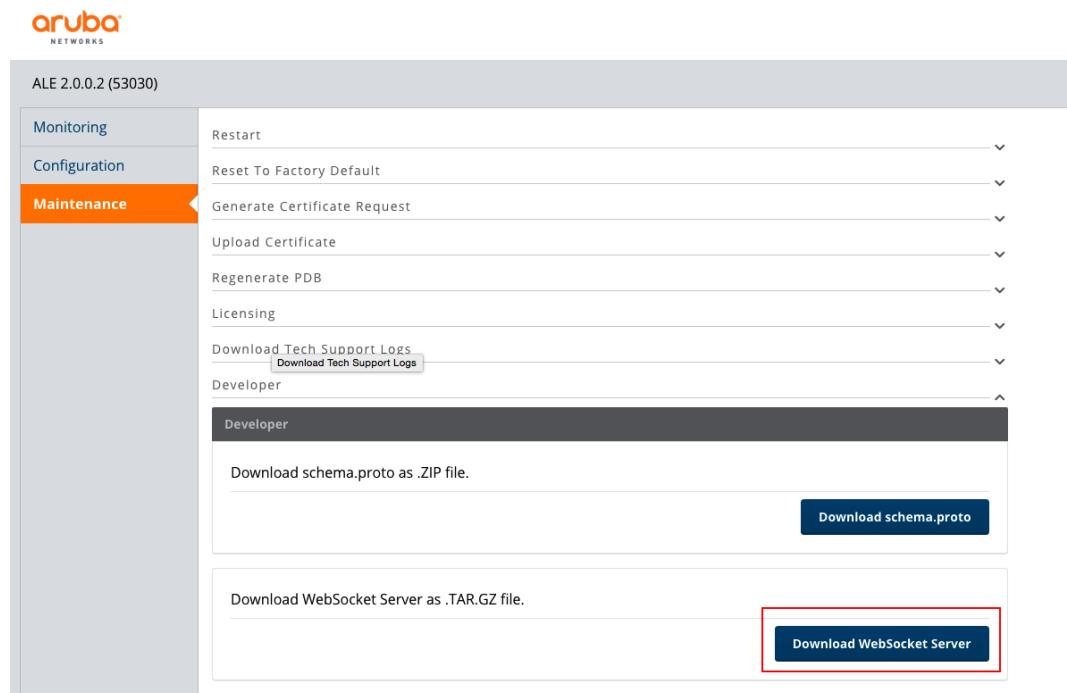
Install the latest package, if not already installed:

```
[root@websocket-server ~]# yum install java-1.8.0-openjdk.x86_64
```

Install the WebSocket Software Package

2. Download the WebSocket software package on your WebSocket server. The package is available to download from ALE under **Maintenance > Developer**.

Figure 55 Download WebSocket Software



Download and untar the file to a folder path of your choice.

```
[root@websocket-server ~]# tar -xzf WebSocketServer.tar.gz -C /opt/ale
```

Navigate to the Websocket folder. Here's an example:

```
[root@websocket-server ale-wstunnel]# pwd  
/opt/ale/ale-wstunnel  
[root@websocket-server ale-wstunnel]# ls  
bin conf lib logs  
[root@websocket-server ale-wstunnel]#
```

The **conf** folder holds the configuration files and the **bin** folder holds the startup scripts.

Create a Server Certificate

3. Create and download a server certificate. In the example below, a self-signed certificate was created using the `openssl` tool for demonstration purposes. In an actual deployment, it is recommended that the certificate be created by a trusted certificate authority.

```
[root@websocket-server ale-wstunnel]# yum install openssl
```

- Create a private key (enter a passphrase of your choice when prompted):

```
[root@websocket-server ale-wstunnel]# openssl genrsa -des3 -out server.key 1024  
Generating RSA private key, 1024 bit long modulus  
.....+++++  
.....+++++  
e is 65537 (0x10001)  
Enter pass phrase for server.key:  
Verifying - Enter pass phrase for server.key:  
[root@websocket-server ale-wstunnel]#
```

- Create a Certificate Signing Request (CSR) using the private key. Enter the passphrase (that you created above), when prompted and fill in the certificate details.

Important: For the Common Name (CN), it is recommended that you use a hostname and not an IP address.

```
[root@websocket-server ale-wstunnel]# openssl req -new -key server.key -out server.csr  
Enter pass phrase for server.key:  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
----  
Country Name (2 letter code) [XX]:US  
State or Province Name (full name) []:CA  
Locality Name (eg, city) [Default City]:Sunnyvale  
Organization Name (eg, company) [Default Company Ltd]:HPE Aruba  
Organizational Unit Name (eg, section) []:TME  
Common Name (eg, your name or your server's hostname) []:tme-websocket.arubanetworks.com  
Email Address []:tme@hpe.com  
  
Please enter the following 'extra' attributes  
to be sent with your certificate request  
A challenge password []:  
An optional company name []:  
[root@websocket-server ale-wstunnel]#
```

- Remove the passphrase from the key (important):

```
[root@websocket-server ale-wstunnel]# cp server.key server.key.tmp
[root@websocket-server ale-wstunnel]# openssl rsa -in server.key.tmp -out server.key
Enter pass phrase for server.key.tmp:
writing RSA key
[root@websocket-server ale-wstunnel]#
```

- Generate the server certificate. It is recommended that the server certificate be generated in the .crt format.

```
[root@websocket-server ale-wstunnel]# openssl x509 -req -days 365 -in server.csr -signkey
server.key -out server.crt
Signature ok
subject=/C=US/ST=CA/L=Sunnyvale/O=HPE Aruba/OU=TME/CN=tme-
websocket.arubanetworks.com/emailAddress=tme@hpe.com
Getting Private key
[root@websocket-server ale-wstunnel]#
```

- Verify the certificate has been created:

```
[root@websocket-server ale-wstunnel]# ls
bin  conf  lib  logs  server.crt  server.csr  server.key  server.key.tmp
[root@websocket-server ale-wstunnel]#
```

- If you can read the certificate contents, then the certificate has been properly generated:

```
[root@websocket-server ale-wstunnel]# openssl x509 -in server.crt -text
Certificate:
Data:
    Version: 1 (0x0)
    Serial Number: 11884188135103923222 (0xa4ed1e24ccc80416)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=US, ST=CA, L=Sunnyvale, O=HPE Aruba, OU=TME, CN=tme-
websocket.arubanetworks.com/emailAddress=tme@hpe.com
    Validity
        Not Before: Jan 29 01:13:24 2016 GMT
        Not After : Jan 28 01:13:24 2017 GMT
    Subject: C=US, ST=CA, L=Sunnyvale, O=HPE Aruba, OU=TME, CN=tme-
websocket.arubanetworks.com/emailAddress=tme@hpe.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        Public-Key: (1024 bit)
        Modulus:
            00:d5:3d:27:6f:cb:bb:1a:88:86:df:1e:ed:b4:2e:
            dd:31:e8:53:bd:c7:b8:7a:f0:72:cb:2c:fa:ad:01:
            29:57:68:05:8d:a9:24:04:0e:bb:47:35:22:88:0f:
            58:fd:49:6a:40:38:f2:6b:15:34:3d:24:c2:4f:e5:
            39:ec:2e:92:8d:1e:4b:e3:18:79:45:6c:a8:f6:16:
            57:d1:be:4b:9e:4f:03:43:57:04:7d:6a:c2:d6:76:
            e0:d2:56:cb:fa:7b:4d:ba:95:1c:36:e6:8e:2a:18:
            9d:d0:a1:fc:a3:04:d6:08:80:07:89:9a:d2:8a:aa:
            4c:f5:2a:b7:a4:cf:9c:db:03
        Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
        43:18:6b:c8:12:71:f1:6c:42:f5:9a:45:2d:1e:b8:b6:9e:92:
        32:6e:e6:43:24:f6:02:08:ec:a2:ff:fe:be:16:c8:e9:82:4b:
        f2:a7:a4:d5:bb:a8:52:1e:bb:2d:a7:3e:7f:80:7a:37:1e:17:
        53:a4:b8:6a:bf:22:71:7f:62:18:a9:17:7c:18:33:e6:54:73:
        61:c2:a5:69:05:e5:83:12:8d:2e:e3:7a:8c:6b:29:fe:a2:1c:
        80:a8:a4:6f:87:5a:50:fe:29:0b:f4:06:ed:0c:84:8f:98:30:
        5b:46:81:0e:bc:eb:70:8a:3c:5d:42:13:57:e6:4e:01:47:bd:
        81:9c
-----BEGIN CERTIFICATE-----
MIICpTCCAg4CCQCk7R4kzMgEFjANBgkqhkiG9w0BAQUFADCB1jELMAkGA1UEBhMC
VVMxCzAJBgNVBAgMAkNBMRlIwEAYDVQQHDA1TdW5ueXZhbGUxEjAQBgNVBAoMCUhQ
RSBBcnViYTEMMAoGA1UECwwDVE1FMSgwJgYDVQQDDB90bWUtd2Vic29ja2V0LmFy
dWJhbmV0d29ya3MuY29tMRowGAYJKoZIhvcNAQkBFgt0bWVAaHB1LmNvbTAeFw0x
NjAxMjkwMTEzMjRaFw0xNzAxMjgwMTEzMjRaMIGWMQswCQYDVQQGEwJVUzELMAkG
A1UEAwCQ0ExEjAQBgNVBAcMCVN1bm55dmFsZTESMBAGA1UECgwJSFBFIEFydWJh
MQwwCgYDVQQLDANUTUUxDKAmBgNVBAMMH3RtZS13ZWJzb2NrZXQuYXJ1YmFuZXR3
b3Jrcy5jb20xGjAYBgkqhkiG9w0BCQEWC3RtZUBocGUuY29tMIGfMA0GCSqGSIb3
DQEBAQUAA4GNADCBiQKBgQDVPSdvy7saiIbfHu20Lt0x6FO9x7h68HLLPqtAS1X
aAWNqSQEDrtHNSKID1j9SwpAOPJrFTQ9JMJP5TnsLpKNhvqjGH1FbKj2F1fRvkue
TwNDVwR9asLWduDSVsv6e0261Rw25o4qGJ3QofyjBNYIgAeJmtKKqkz1Krekz5zb
AwIDAQABMA0GCSqGSIb3DQEBBQAA4GBAEMYa8gScffFsQvWaRS0euLaekjJu5kMk
9gII7KL//r4WyOmCS/KnpNW7qF1euy2nPn+AejceF1OkuGq/InF/YhipF3wYM+ZU
c2HCpWkF5YMSjS7jeoxrKf6iHICopG+HW1D+KQv0Bu0MhI+YMftGgQ6863CKPF1C
E1fmTgFHvYGc
-----END CERTIFICATE-----
[root@websocket-server ale-wstunnel]#
```

Set the WebSocket Server's Properties

4. Edit the **server.properties** file that is part of the WebSocket software package.

There are two fields that you need to edit here - the file paths for **server.crt** and **server.key**.

You can also optionally edit the port on which the WebSocket server will listen for incoming WebSocket connection requests. In the example below, port 4433 is used instead of the default port 443. If you are using an ALE server as the WebSocket server for testing purposes, it is recommended that the listening port be changed to other than the default value (443), since another application on the ALE server is already listening on port 443 by default.

If using the **vi** editor, you need to press the letter **i** before you can make any changes to the file. Once changes have been made, press **Esc** and then type **:wq** to save and exit. Use **:q!** to exit without saving your changes.

```
[root@websocket-server ale-wstunnel]# vi conf/server.properties
```

Make Firewall Exceptions

5. Ensure that the firewall on the WebSocket server allows incoming requests on TCP port 4433 (or whatever you have configured in the previous step).

```
[root@websocket-server ale-wstunnel]# vi /etc/sysconfig/iptables
```

Add the port entry and save your changes:

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 4000 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 7779 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8088 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 4433 -j ACCEPT <<----- Add the
following entry
-A INPUT -m state --state NEW -m udp -p udp --dport 8211 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
~
~
~
~
~
~
:wg      <<----- Save your config changes
```

Restart the firewall:

```
[root@websocket-server bin]# /etc/init.d/iptables restart
iptables: Setting chains to policy ACCEPT: filter      [  OK  ]
iptables: Flushing firewall rules:                   [  OK  ]
iptables: Unloading modules:                         [  OK  ]
iptables: Applying firewall rules:                  [  OK  ]
[root@websocket-server bin]#
```

Start the WebSocket Service

6. Start the WebSocket server from the **bin** path.

```
[root@websocket-server ale-wstunnel]# ls
bin  conf  lib  logs  server.crt  server.csr  server.key  server.key.tmp
[root@websocket-server ale-wstunnel]# cd bin
[root@websocket-server bin]# ls
startup-client.sh  startup-server.sh
[root@websocket-server bin]# ./startup-server.sh
[root@websocket-server bin]#
```

Verify that the tunnel is active:

```
[root@websocket-server bin]# ps aux |grep Tunnel
root      2807  0.0  1.7 4069488 106960 ?        S1   Jan28   0:06 /usr/java/jdk1.8.0_31/bin/java -cp /opt/ale/ale-wstunnel/conf:/opt/ale/ale-wstunnel/lib/ale-wstunnel-0.0.1.jar -Dlogger.basedir=/opt/ale/ale-wstunnel/logs -Djava.library.path=/opt/ale/lib64 -Dlogger.console=false com.aruba.ale.wstunnel.TunnelClient
root      4421  2.4  1.5 3964372 93276 pts/0    S1   02:00   0:02 /usr/bin/java -cp /opt/ale/ale-wstunnel/conf:/opt/ale/ale-wstunnel/lib/ale-wstunnel-0.0.1.jar -Dlogger.basedir=/opt/ale/ale-wstunnel/logs -Dlogger.console=false com.aruba.ale.wstunnel.TunnelServer
root      4451  0.0  0.0 103244   872 pts/0     S+   02:02   0:00 grep Tunnel
[root@websocket-server bin]#
```

WebSocket Client (ALE)

Verify WebSocket Server Reachability

7. On ALE, verify that you can resolve the CN of the server certificate, as ALE will use the CN to initiate the WebSocket connection.

In the previous steps, **CN=tme-websocket.arubanetworks.com** was specified when creating the WebSocket server certificate. If this value is not publicly resolvable, then create an entry on ALE to locally resolve **tme-websocket.arubanetworks.com**.

```
[root@ale /]# vi /etc/hosts

127.0.0.1      localhost.localdomain  localhost.localdomain  localhost4
localhost4.localdomain4  localhost        ale-2
::1      localhost.localdomain  localhost.localdomain  localhost6
localhost6.localdomain6  localhost        ale-2
10.70.120.234 tme-websocket.arubanetworks.com      <<<-----Add

~
~
~
~
~
~
~
~

:wq      <<<----- Save your config changes
```

Verify that you can ping the WebSocket server:

```
[root@ale /]# ping tme-websocket.arubanetworks.com
PING tme-websocket.arubanetworks.com (10.70.120.234) 56(84) bytes of data.
64 bytes from tme-websocket.arubanetworks.com (10.70.120.234): icmp_seq=1 ttl=64 time=0.526 ms
64 bytes from tme-websocket.arubanetworks.com (10.70.120.234): icmp_seq=2 ttl=64 time=0.345 ms
^C
--- tme-websocket.arubanetworks.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1267ms
rtt min/avg/max/mdev = 0.345/0.435/0.526/0.092 ms
[root@ale /]#
```

Trust the WebSocket Server's Identity

- The next step is to import the WebSocket server's chain of trust into ALE. In this example, the WebSocket server has a self-signed certificate that can be imported into ALE.

There are a number of ways to pull the certificate from the WebSocket server, including Secure Copy (SCP), File Transfer Protocol (FTP), or manually copying the certificate. In the example below, we will attempt to pull the certificate directly from the WebSocket server using **Openssl**.

```
[root@ale ~]# echo -n | openssl s_client -connect tme-websocket.arubanetworks.com:4433 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /root/websocket-server.crt
depth=0 C = US, ST = CA, L = Sunnyvale, O = HPE Aruba, OU = TME, CN = tme-
websocket.arubanetworks.com, emailAddress = tme@hpe.com
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, ST = CA, L = Sunnyvale, O = HPE Aruba, OU = TME, CN = tme-
websocket.arubanetworks.com, emailAddress = tme@hpe.com
verify return:1
DONE
[root@ale ~]#
```

Verify that the server certificate was imported.

```
[root@ale ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog  websocket-server.crt
[root@ale ~]#
```

You can also issue the following command to verify that certificate contents can be read:

```
[root@ale ~]# openssl x509 -in websocket-server.crt -text
```

Now that the server certificate has been copied on ALE, import it into the trusted root certificate store on ALE.

```
[root@ale ~]# cd /usr/java/jdk1.8.0_31/bin/
[root@ale bin]# ./keytool -list -keystore ../jre/lib/security/cacerts -importcert -alias
poclab-websocket-server-cert -file /root/websocket-server.crt
Enter keystore password:                                     <<<<----- Type
password "changeit"
Owner: EMAILADDRESS=tme@hpe.com, CN=tme-websocket.arubanetworks.com, OU=TME, O=HPE Aruba,
L=Sunnyvale, ST=CA, C=US
Issuer: EMAILADDRESS=tme@hpe.com, CN=tme-websocket.arubanetworks.com, OU=TME, O=HPE Aruba,
L=Sunnyvale, ST=CA, C=US
Serial number: a4ed1e24ccc80416
Valid from: Fri Jan 29 01:13:24 UTC 2016 until: Sat Jan 28 01:13:24 UTC 2017
Certificate fingerprints:
    MD5: F2:F4:F1:07:40:1D:A7:9E:17:3F:3D:F7:C7:24:E3:A8
        SHA1: 14:A5:64:9C:0D:A8:8A:4E:F5:17:DF:99:0A:2D:98:54
        :3D:D2:83:39
    SHA256: 91:8C:49:C1:81:AD:22:8B:0F:7A:13:8B:C3:1B:BA:
A3:B5:CD:8E:A3:B3:AE:D8:76:C5:9B:F8:78:A4:2F:BC:D9
        Signature algorithm name: SHA1withRSA
        Version: 1
Trust this certificate? [no]: yes                         <<<<----- Type yes
Certificate was added to keystore
[root@ale bin]#
```

To verify the import was successful:

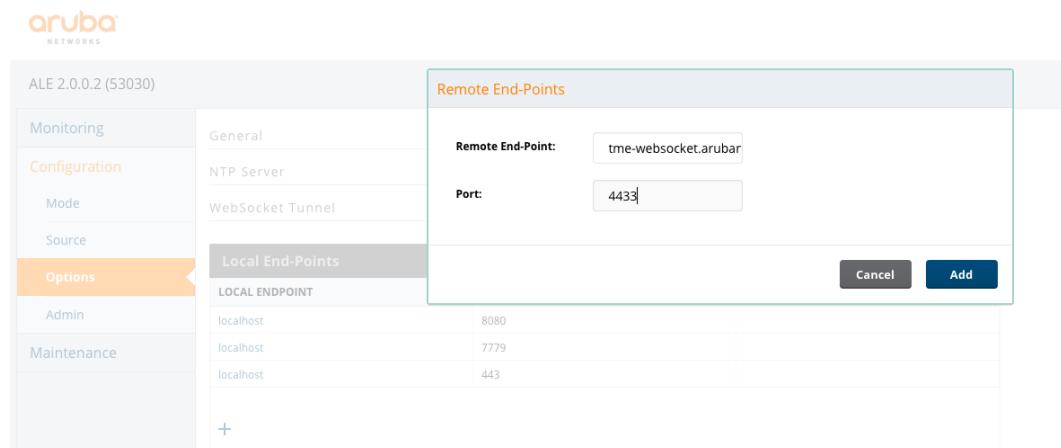
```
[root@ale bin]# ./keytool -list -keystore ../jre/lib/security/cacerts |grep poclab-
websocket-server-cert -A1
Enter keystore password: changeit
poclab-websocket-server-cert, Jan 29, 2016, trustedCertEntry,
Certificate fingerprint (SHA1): 14:A5:64:9C:0D:A8:8A:4E:F5:17:DF:99:0A:2D:98:54:3D:D2:83:39
[root@ale bin]#
```

Configure the WebSocket Entry On ALE

- On the ALE UI, specify the WebSocket server as the WebSocket end-point to connect to.

Under **Config > Options > WebSocket Tunnel > Remote End-Points** > Click + to add the remote end-point.

Figure 56 *Remote End-Point*



Click **Add**. On the WebSocket Tunnel page, click **Apply**.

Verify the Connection

- You can view the “ale-wstunnel” log to verify on ALE that the connection was successfully established:

```
[root@ale logs]# tailf /opt/ale/ale-wstunnel/logs/ale-wstunnel.log
```

The following output indicates a successful WebSocket connection:

```
2016-01-29 03:06:50.289 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.client.TunnelConnector - WebSocket connecting to wss://tme-
websocket.arubanetworks.com:4433
2016-01-29 03:06:50.783 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.client.TunnelConnector - WebSocket connected to tme-
websocket.arubanetworks.com:4433
```

Similarly, on the WebSocket server, you can monitor for incoming WebSocket connections from ALE:

```
[root@websocket-server logs]# tailf /opt/ale/ale-wstunnel/logs/ale-wstunnel.log
```

The following output indicates a successful WebSocket connection:

```

2016-01-29 02:01:01.858 UTC [vert.x-eventloop-thread-0] INFO
com.aruba.ale.wstunnel.TunnelServer - Listening for clients on wss://0.0.0.0:4433
2016-01-29 02:01:01.867 UTC [vert.x-eventloop-thread-0] INFO
com.aruba.ale.wstunnel.TunnelServer - Listening for web/API requests on
http://localhost:8700
2016-01-29 03:06:50.724 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.s.WebSocketServerTunnel - Client '005056852F31' connected.
2016-01-29 03:06:50.730 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12000 -> localhost:8080
2016-01-29 03:06:50.731 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12001 -> localhost:7779
2016-01-29 03:06:50.731 UTC [vert.x-eventloop-thread-0] INFO
c.a.a.w.s.WebSocketServerTunnel - - Mapping localhost:12002 -> localhost:443

```

Test the Polling and Publish/Subscribe APIs

Before testing the APIs on the WebSocket server, verify that ALE:WebSocket-server's {remote:local} port mappings are in place:

```

[root@websocket-server ~]# curl -v http://localhost:8700/clients
* About to connect() to localhost port 8700 (#0)
*   Trying ::1... Connection refused
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8700 (#0)
> GET /clients HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:8700
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Content-Length: 339
<
* Connection #0 to host localhost left intact
* Closing connection #0
{"clients": [{"id": "005056852F31", "uptime": 57898177, "endpoints": [{"local_host": "localhost", "local_port": 12000, "remote_host": "localhost", "remote_port": 8080}, {"local_host": "localhost", "local_port": 12001, "remote_host": "localhost", "remote_port": 7779}, {"local_host": "localhost", "local_port": 12002, "remote_host": "localhost", "remote_port": 443}]}]}
[root@websocket-server ~]#

```

The above output gives the following information:

- The **id** field shows the ALE server that the WebSocket server is connected to, via the tunnel.
- Remote port 8080 (used for REST APIs) is mapped to local port 12000.
- Remote port 7779 (used for Pub/Sub APIs) is mapped to local port 12001.
- Remote port 443 is mapped to local port 12002.

With ALE 2.0, all Polling API requests need to pass authentication. Just so we do not have to authenticate every time an API request is placed, let's store the credentials in a temporary cookie file.

```

curl -v -k --data "j_username=admin&j_password=welcome123" --cookie-jar /tmp/cook.txt
http://localhost:12000/api/j_spring_security_check

```

Where **admin/welcome123** are the ALE credentials.

Polling APIs

Now issue any of the Polling APIs. Below are a few examples.

Building API

```
[root@websocket-server ~]# curl -v -k --cookie /tmp/cook.txt
http://localhost:12000/api/v1/building
* About to connect() to localhost port 12000 (#0)
*   Trying ::1... connected
* Connected to localhost (::1) port 12000 (#0)
> GET /api/v1/building HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:12000
> Accept: */*
> Cookie: JSESSIONID=0DC5EB94A185EAD018F2E2C7FA8D97A8
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json; charset=UTF-8
< Content-Language: en-US
< Content-Length: 175
< Date: Fri, 29 Jan 2016 19:25:16 GMT
<
* Connection #0 to host localhost left intact
* Closing connection #0
{"Building_result": [{"msg": {"building_id": "659685E876A6325EB3F974FBBA530F8", "building_name": "POC lab", "campus_id": "FCCD5881918E3C8D8628130773403AA6"}, "ts": 1453948738}]}
[root@websocket-server ~]#
```

Floor API

```
[root@websocket-server ~]# curl -v -k --cookie /tmp/cook.txt
http://localhost:12000/api/v1/floor
* About to connect() to localhost port 12000 (#0)
*   Trying ::1... connected
* Connected to localhost (::1) port 12000 (#0)
> GET /api/v1/floor HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:12000
> Accept: */*
> Cookie: JSESSIONID=0DC5EB94A185EAD018F2E2C7FA8D97A8
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json; charset=UTF-8
< Content-Language: en-US
< Content-Length: 407
< Date: Fri, 29 Jan 2016 19:24:08 GMT
<
* Connection #0 to host localhost left intact
* Closing connection #0
{"Floor_result": [{"msg": {"floor_id": "8B5207D8FDB1319AB35A72A32C7E5937", "floor_name": "Floor 1", "floor_latitude": 0.0, "floor_longitude": 0.0, "floor_img_path": "/download/10.70.120.251/images/7f67cc85-81c7-4e2d-9b95-0f385d0bc4c8.jpg", "floor_img_width": 152.0, "floor_img_length": 147.0, "building_id": "659685E876A6325EB3F974FBBA530F8", "floor_level": 1.0, "units": "ft", "grid_size": 10.0}, "ts": 1453948738}]}
[root@websocket-server ~]#
```

Station API

```
[root@websocket-server ~]# curl -v -k --cookie /tmp/cook.txt
http://localhost:12000/api/v1/station
* About to connect() to localhost port 12000 (#0)
*   Trying ::1... connected
* Connected to localhost (::1) port 12000 (#0)
> GET /api/v1/station HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.16.2.3 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:12000
> Accept: */*
> Cookie: JSESSIONID=0DC5EB94A185EAD018F2E2C7FA8D97A8
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json; charset=UTF-8
< Content-Language: en-US
< Content-Length: 1737
< Date: Fri, 29 Jan 2016 19:25:39 GMT
<
{"Station_result": [{"msg": {"sta_eth_mac": {"addr": "ACBC32B8599F"}, "role": "authenticated", "bssid": {"addr": "186472E81930"}, "device_type": "OS X", "hashed_sto_eth_mac": "D540B41E28ED584DE9753BB998D0755E126029B8", "ap_name": "POC-ALE-03-81:92"}, "ts": 1454080155}, {"msg": {"sta_eth_mac": {"addr": "7831C1B8B3D8"}, "role": "authenticated", "bssid": {"addr": "186472E58E90"}, "device_type": "OS X", "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "10.70.20.129"}, "hashed_sto_eth_mac": "64D57D97954F40423B11B96DDE68412C5CDB50D1"}, "ts": 1454090074}, {"msg": {"sta_eth_mac": {"addr": "60F81DB76448"}, "role": "authenticated", "bssid": {"addr": "186472E8A810"}, "device_type": "OS X", "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "10.70.20.227"}, "hashed_sto_eth_mac": "FBD61BE219A039C83698488040552393D2CD49B9"}, "ts": 1454094479}, {"msg": {"sta_eth_mac": {"addr": "247703D5B250"}, "role": "authenticated", "bssid": {"addr": "186472E7E9B0"}, "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "192.168.1.221"}, "hashed_sto_eth_mac": "33F29AEA2CF514F153058AE2869A085B27517579"}, "ts": 1454093728}, {"msg": {"sta_eth_mac": {"addr": "0C3E9F5CEA01"}, "role": "authenticated", "bssid": {"addr": "186472E58E90"}, "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "10.70.20.217"}, "hashed_sto_eth_mac": "7B35C0B8EF0F2050410AD5E5909A5EA0D5F17A61"}, "ts": 1454091368}], "m*: Connection #0 to host localhost left intact
* Closing connection #0
sg": {"sta_eth_mac": {"addr": "247703D5B250"}, "role": "authenticated", "bssid": {"addr": "186472E7E9B0"}, "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "192.168.1.221"}, "hashed_sto_eth_mac": "33F29AEA2CF514F153058AE2869A085B27517579"}, "ts": 1454093728}, {"msg": {"sta_eth_mac": {"addr": "0C3E9F5CEA01"}, "role": "authenticated", "bssid": {"addr": "186472E58E90"}, "sta_ip_address": {"af": "ADDR_FAMILY_INET", "addr": "10.70.20.217"}, "hashed_sto_eth_mac": "7B35C0B8EF0F2050410AD5E5909A5EA0D5F17A61"}, "ts": 1454091368}]} [root@websocket-server ~]#
```

You can also test the Polling APIs via a Web browser.

Figure 57 WebSocket Authentication

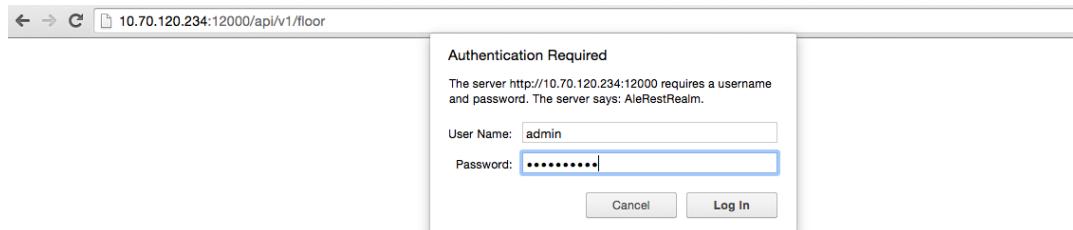
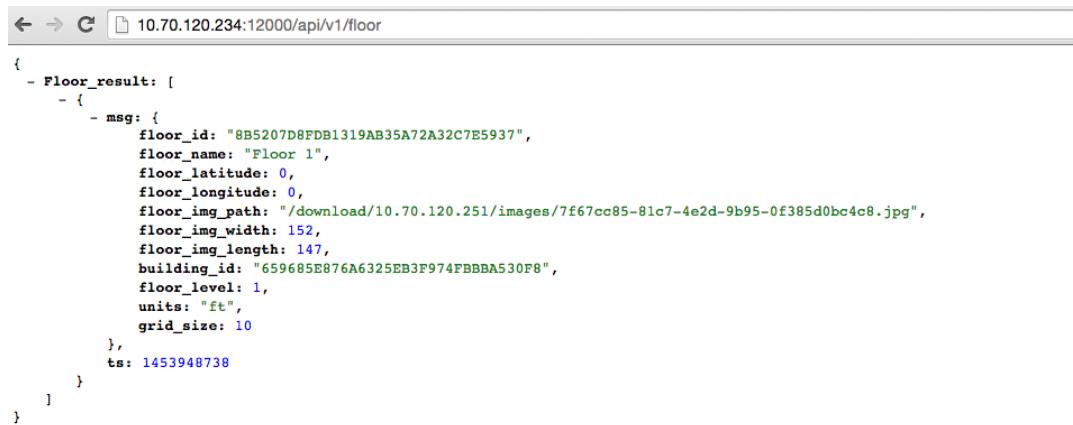


Figure 58 *WebSocket Floor API*



The screenshot shows a browser window with the URL `10.70.120.234:12000/api/v1/floor`. The page displays a JSON response with the following structure:

```
{  
  - Floor_result: [  
    - {  
      - msg: {  
          floor_id: "8B5207D8FDB1319AB35A72A32C7E5937",  
          floor_name: "Floor 1",  
          floor_latitude: 0,  
          floor_longitude: 0,  
          floor_img_path: "/download/10.70.120.251/images/7f67cc85-81c7-4e2d-9b95-0f385d0bc4c8.jpg",  
          floor_img_width: 152,  
          floor_img_length: 147,  
          building_id: "659685E876A6325EB3F974FBBA530F8",  
          floor_level: 1,  
          units: "ft",  
          grid_size: 10  
        },  
        ts: 1453948738  
      }  
    ]  
}
```

In the above example, a Google Chrome extension called JSONView was used to make the JSON output more readable.

Publish/Subscribe APIs

To subscribe to the ALE feed in real-time, the customer can build a ZMQ-based app based on a language of their choice.

If you have an ALE instance as your WebSocket server, you can use a native debugging tool called feed-reader in order to test the feed. The feed-reader code (available in Java and C++) is available by contacting Support or your local Aruba account team, and can be used by customers and partners when building their own ZMQ-based apps.

Continuing the same example, local port 12001 corresponds to remote port 7779.

```

[root@websocket-server ~]# /opt/ale/bin/feed-reader -e tcp://localhost:12001
Attempting to 'connect' to endpoint: tcp://localhost:12001
Connected to endpoint: tcp://localhost:12001
Subscribed to topic: ""
[1] Recv event with topic "location/40:e2:30:1c:93:4d"
seq: 331807
timestamp: 1454097056
op: OP_UPDATE
topic_seq: 242538
source_id: 005056852F31
location {
    sta_eth_mac {
        addr: 40:e2:30:1c:93:4d
    }
    sta_location_x: 113.759
    sta_location_y: 112.824
    error_level: 24
    associated: true
    campus_id: FCCD5881918E3C8D8628130773403AA6
    building_id: 659685E876A6325EB3F974FBBBA530F8
    floor_id: 8B5207D8FDB1319AB35A72A32C7E5937
    hashed_sta_eth_mac: 8A5C4F9F50A25B5B4E779DC27E562A1295FAD79B
    geofence_ids: 09B034AE4CE93995AC92895E91C390FF
    loc_algorithm: ALGORITHM_ESTIMATION
    unit: FEET
}
[2] Recv event with topic "location/d0:e1:40:9f:02:52"
seq: 331808
timestamp: 1454097058
op: OP_UPDATE
topic_seq: 242539
source_id: 005056852F31
location {
    sta_eth_mac {
        addr: d0:e1:40:9f:02:52
    }
    sta_location_x: 27.247
    sta_location_y: 41.2553
    error_level: 30
    associated: true
    campus_id: FCCD5881918E3C8D8628130773403AA6
    building_id: 659685E876A6325EB3F974FBBBA530F8
    floor_id: 8B5207D8FDB1319AB35A72A32C7E5937
    hashed_sta_eth_mac: D850DF23FF1B9A7B168AB80A32E2BECE76FA5E5B
    loc_algorithm: ALGORITHM_ESTIMATION
    unit: FEET
}
<output snipped>

```

You can subscribe to specific topics such as location, station, presence, etc.

```
[root@websocket-server ~]# /opt/ale/bin/feed-reader -e tcp://localhost:12001 -f station
Attempting to 'connect' to endpoint: tcp://localhost:12001
Connected to endpoint: tcp://localhost:12001
Subscribed to topic: "station"
[1] Recv event with topic "station"
seq: 332615
timestamp: 1454097231
op: OP_ADD
topic_seq: 223
source_id: 005056852F31
station {
    sta_eth_mac {
        addr: fc:c2:de:aa:94:bc
    }
    role: authenticated
    bssid {
        addr: 18:64:72:e5:8e:80
    }
    device_type: Android
    hashed_sto_eth_mac: 49C6C19587FAEF786B53CD7CF46B3C7C110D2355
    ap_name: POC-ALE-08-58:e8
}
<output snipped>
```

Using the ALE Demonstrator Application

If you have deployed ALE in Estimation or Calibration mode and would like to quickly test locations in the absence of an analytics application backend, you can do so using an Android-based app called ALE Demonstrator.

ALE Demonstrator can also be used by analytics partners to understand how the REST and Publish/Subscribe APIs are used to pull static data like AP placements and streaming location data respectively. The functional source code for ALE Demonstrator is available on GitHub for reference.

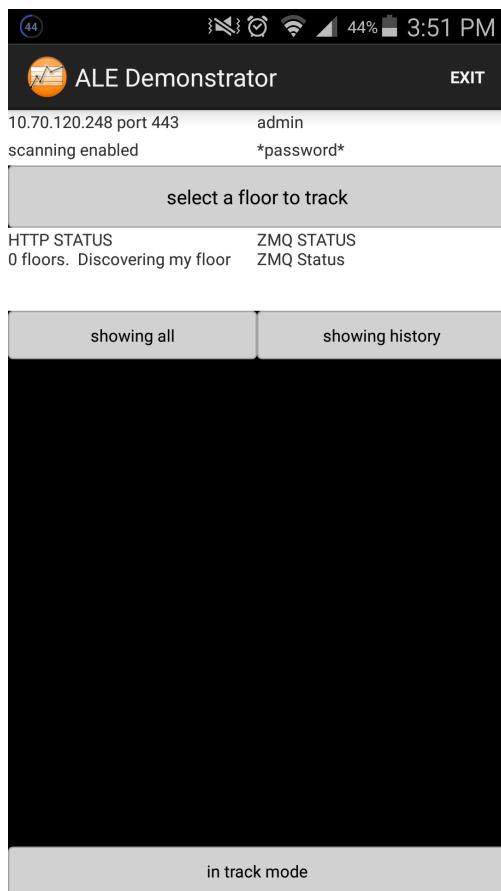
<https://github.com/pthornycroft/ALE-Demonstrator>

Below are some example screenshots that show how to set up and use ALE Demonstrator. Before you start setting up the app, ensure that Wi-Fi is enabled on your device and the ALE server that you are subscribing to is reachable.

Before using ALE Demonstrator, please ensure TCP connections to ports 443 and 7779 on ALE are allowed by the network firewall.

1. Launch the ALE Demonstrator app and click on the white bar on the top.

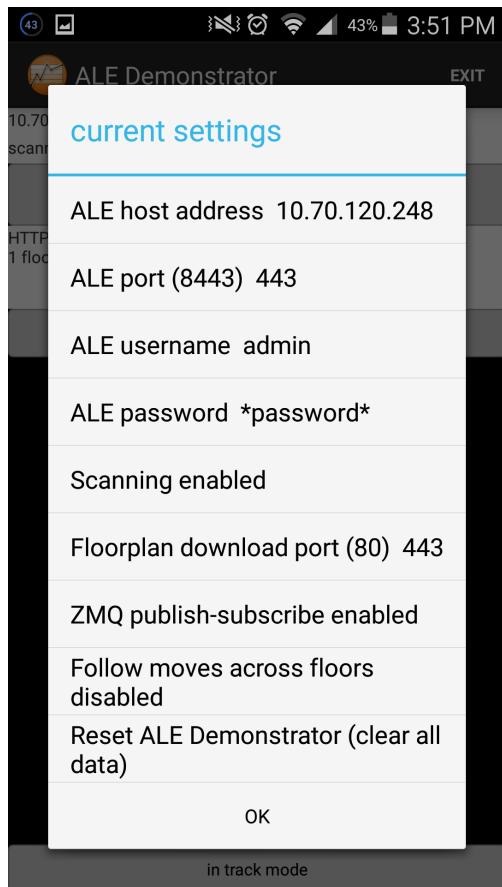
Figure 59 ALE Demonstrator



2. On the “Current Settings” screen, enter the following and tap on OK:

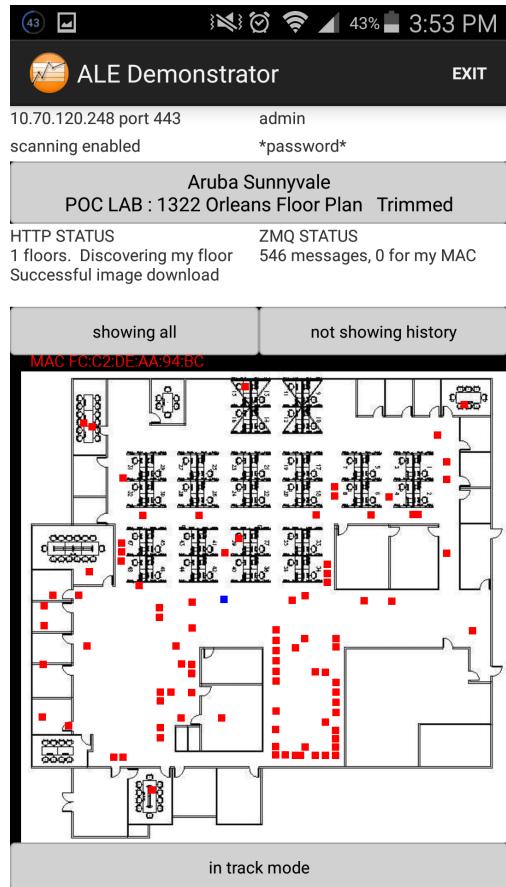
- ALE server’s IP address
- Port number = 443
- ALE username/password for login
- Ensure scanning is enabled
- Floor plan download port = 443
- Ensure ZMQ publish-subscribe is enabled

Figure 60 ALE Demonstrator - Current Settings



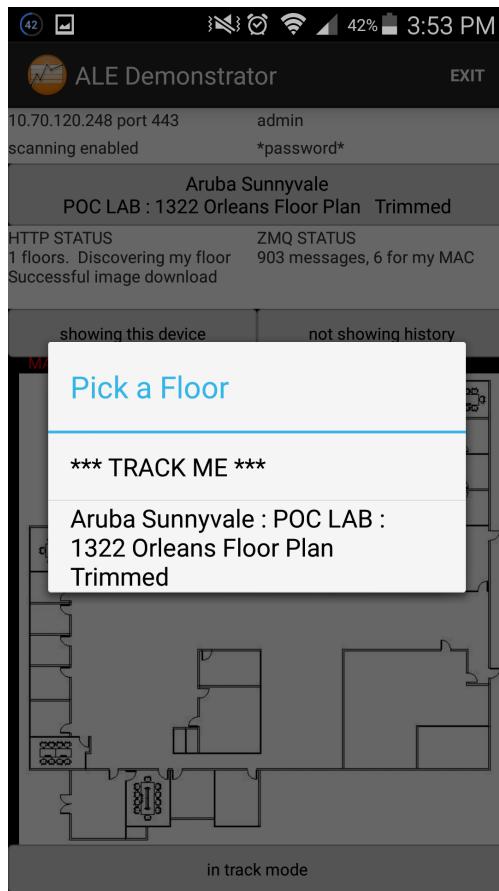
3. You will see the floor plan being downloaded and populated with clients on the screen. The main page also shows the number of ZMQ messages being published by ALE and how many belong to your device.
On the main page, tap on the top gray section of the page (where map information is displayed).

Figure 61 ALE Demonstrator - Map Information



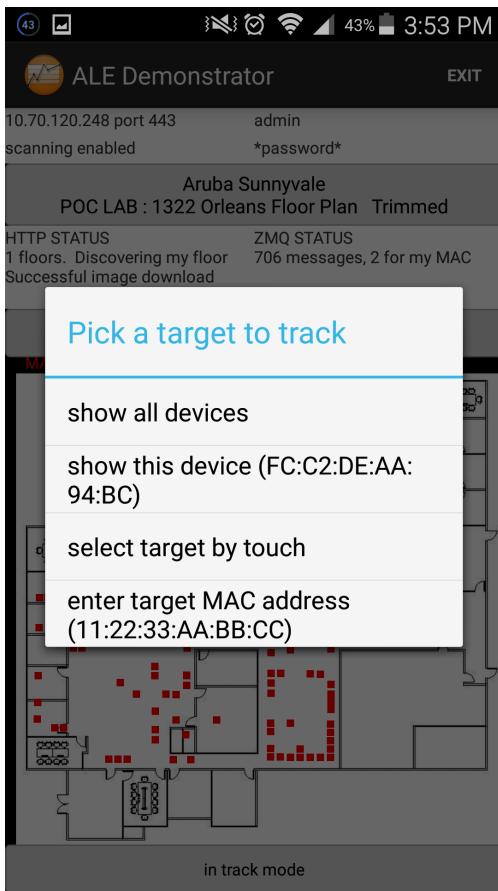
4. On the “Pick a Floor” screen, select “*** TRACK ME ***” to automatically load the floor where your device is present.

Figure 62 ALE Demonstrator - Pick a Floor



5. On the main screen, you can also select to only track the MAC address of your device by tapping on the “show all” button in the second gray section and selecting “show this device <MAC-address>.”

Figure 63 ALE Demonstrator - Pick a target to track



6. You can also track how your device is being historically located on the floor by toggling the “show history” button in the second gray section on the main screen.