User Guide

Ayla RBAC API Specification

Version: 5.0

Date Released: August 17, 2017

Document Number: AY006URB6-5

Ayla Networks

Referenced Documents

Ayla Networks does not supply all documents that are referenced in this document with the equipment. Ayla Networks reserves the right to decide which documents are supplied with products and services.

Contact Information

Ayla Networks TECHNICAL SUPPORT and SALES

Contact Technical Support: https://support.aylanetworks.com
or via email at support@aylanetworks.com

Contact Sales: https://www.aylanetworks.com/company/contact-us

Ayla Networks REGIONAL OFFICES

GREATER CHINA
Shenzhen
Room 310-311
City University of Hong Kong
Research Institute Building
No. 8 Yuexing 1st Road
High-Tech Industrial Park
Nanshan District
Shenzhen, China
Phone: 0755-86581520

HEADQUARTERS
Silicon Valley
4250 Burton Drive, Suite 100
Santa Clara, CA 95054
United States
Phone: +1 408 830 9844
Fax: +1 408 716 2621

JAPAN
Wise Next Shin
Yokohama, 2-5-14
Shnyokohama, Kohokuku
Yokohama-shi, Kanagawa-ken
Yokohoma, 222-0033 Japan

EUROPE
London
30 Great Guildford St
London SE1 0HS
United Kingdom

TAIWAN
Taipei
5F No. 250 Sec. 1
Neihu Road, Neihu District
Taipei 11493, Taiwan

For a Complete Contact List of Our Offices in the US, China, Europe, Taiwan, and Japan:
https://www.aylanetworks.com/company/contact-us

Table of Contents

Revision History

Revision	Date	Author	Change Description
1	2015.09.24	M.D.	Initial document
2	2015.12.	L. Boling	Updated format; added API's
3	2016.05	D.Fulton	Updated API commands
4	2017.05		Updated format, fixed DOC-166 JIRA.
5	2017-08		Added RBAC Use Cases and linked to API Browser

1 Introduction

This document presents an overview of the Role Based Access Control (RBAC) system.

There are two parts to role-based access.

- Administrative: This involves setting up roles, and the privileges associated with them. This is done using the OEM Dashboard at https://dashboard.aylanetworks.com
- Operational: This involves creating users with specified roles and associating resources with them using the Ayla Service APIs as specified in this document.

1.1 Audience

This document is written for engineers and others that are familiar working with API's.

1.2 Related Documentation

Ayla Service API (AY006USA0)

1.3 Terminology

- Role User: A user in the system with a specific role. This type of user typically plays a role other than being a consumer of a device.

  Examples of role users:

  Employees of an OEM who have responsibility to manage users and devices

  Dealers who install devices and maintain an ongoing relationship with customers
- Role Label: A key value pair qualifying a role user. The role label further classifies users who have the same role. Depending on the use case, a role label can be unique or multiple users can share the same role label.

  Example of role labels:

  {"dealer_id":"dealer1"}

  {"location":"APAC"}

2 RBAC Overview

For OEMs building connected devices, obtaining data from their devices offers a wealth of information. Typical use cases include: analytics, auditing, and troubleshooting. An OEM administrator needs to have control over the users in their organization. It is helpful to have different levels of access that are based on their role in the company. For privacy and safety, it is essential that access control policies be enforced when users interact with their devices.

A role is associated with a set of privileges to specific resources. Each user in the system is assigned a role. The role determines the privileges the user has to a resource. Typical resources are devices, users, firmware images, and device templates.

The key features of the RBAC system

- Addresses enterprise and consumer concerns
- Provides flexible, granular controls based on roles
- Supports a hierarchy of roles

2.1 Role Based Access

2.1.1 Usage and Scenarios

There are two parts to role-based access.

- Administrative: This involves setting up roles, and the privileges associated with them. This is done using the OEM Dashboard at https://dashboard.aylanetworks.com
- Operational: This involves creating users with specified roles and associating resources with them using the Ayla Service APIs as specified in this document.

2.1.2 Roles based on Organizational Structure

In a company, organizations are structured using various criteria, such as, product responsibility, location, and so on. In many instances, employees are granted privileges based on the organization they belong to. For example, an employee in the Heating, Ventilation, and Air Conditioning (HVAC) division of a company may be given access to oversee all connected thermostats, but not other wirelessly enabled products, such as, smoke and CO detectors belonging to the safety division.

Using the Ayla RBAC system, an OEM admin can create a role for thermostat admins to have superuser access to all the thermostat models that the company has in the field. The admin can also create a thermostat technician role with privileges to only view part of the data that a thermostat admin can view.

This flexibility allows for policies to be defined that follow a company's organizational structure.

2.1.3 Dealer Distribution

Many products sold through commercial channels are distributed through dealers. Examples are installations in commercial buildings, hospitals, or residences. The dealer typically performs the installation and has a maintenance contract with the customer. With this distribution model, there are many considerations. The dealer should have enough access to a unit after its installation that would allow for its monitoring. However, the customer may want to limit how much of the data the dealer can access due to privacy or other considerations.

2.1.4 Retail Distribution

In this model, customers purchase products directly from retailers. The customer may own a mix of devices from the same OEM, some purchased through a dealer and some from a retailer. In this case, the customer would want to access all owned devices from a single application.

The Ayla RBAC system supports a hierarchical model where the customer can access all owned devices, a dealer can only access allowed information on serviced devices, and the OEM can access all manufactured devices.

2.2 Device Sharing

There are use cases that warrant sharing of a device among many users. An example is any device in a home being used by multiple family members. In this situation, the primary owner of the device shares it with the other family members or with a guest. In a more commercial setting, a hotel could have connected locks installed with access granted to hotel guests for a period of time. Temporary or permanent privileges can be granted using the Ayla service and can be revoked at any time by the owner.

2.3 Role User and Role Label

2.3.1 Role User

A user in the system with a specific role. This type of user typically plays a role other than being a consumer of a device.

Examples of role users:

- Employees of an OEM who have responsibility to manage users and devices
- Dealers who install devices and maintain an ongoing relationship with customers

2.3.2 Role Label

A key value pair qualifying a role user. The role label further classifies users who have the same role. Depending on the use case, a role label can be unique or multiple users can share the same role label.

Examples of role labels:

- {“dealer_id”:”dealer1”}
- {“location”:”APAC”}

3 User Role Management

This set of APIs allows the creation and association of end users with user roles. The association is established by assigning the role label of the role user to the end user.

This is useful in situations where, for example, a dealer wants to be able to create a customer account and associate that customer with self.

For API code and details, log into the Development Portal. Then go to the API Browser > Role Management APIs.

API	Description
PUT /api/v1/users/associate_role_user	This API enables a user to associate self with another role user. It can also be used by OEM Admin to associate a user to another role user.
GET /api/v1/users/role_users	This API lists all role users associated with a user. i.e., to get all the labels associated with a particular user. For example: get all the dealers associated with a user.
PUT /api/v1/users/disassociate_role_user	This API enables a user to disassociate a role user from self. After completion, the role user no longer has access to the user object corresponding to self. This API is also used by OEM admins to disassociate a role user from the user.
PUT /api/v1/users/disassociate_user	This API enables a role_user to disassociate self from a user. After completion, the role user will no longer have access to the user object. For example, a dealer can disassociate a customer from his/her account.

4 Admin User Role Management

Pre-requisite: The role already exists in the system. This can be achieved through the administrative step stated above.

Each user with a role has an associated role label. A role label is a key value pair that further qualifies a role user.

For example, for a given role, OEM::Company::Dealer, one dealer could have a label "dealer_id: dealer1" and another dealer could have a label "dealer_id: dealer2".

For API code and details, log into the Development Portal. Then go to the API Browser > Role Management APIs.

API	Description
GET /api/v1/users/index_by_role	This API gets a list of all users, based on the role of User.
GET /api/v1/users/index_by_oem	This API lists all users of an OEM. This includes all the customers of the OEM. This is intended to be invoked with the credentials of an OEM admin or a user that belongs to the OEM organization.
POST /api/v1/users/create_role_user	This API creates a user with a role by an OEM admin. After completion, the system sends a welcome email to the created user.
PUT /api/v1/users/assign_role	This API assigns a role to an existing user.
PUT /api/v1/users/revoke_role	This API revokes a role from an existing user.
PUT /api/v1/users/assign_label	This API modifies the role label to a user with the given role.
PUT /api/v1/users/revoke_label	This API removes a role label from a user with the given role.
GET /api/v1/users/index_by_label	This API lists all the users associated with a role user. For example: get all users associated with a particular dealer.
POST /api/v1/users/create_associated_user	This API creates a user and associates the user with self. After completion, the system sends a welcome email to the created user. After completion, the invoker of the API continues to have access to the created user object.

5 Device Role Management

This set of APIs enables the association of a role user to a device.

Steps to associate a device with a user:

1. Role enable a device by creating a role label on the device. This is when the device will accept an association from a user with that role. This step can be also be performed through a template such that all devices with a given model are role-enabled.
2. Associate a role user with the device. This assigns the role label of the user to the device, which establishes the association between the device and the user.

For API code and details, log into the Development Portal. Then go to the API Browser > Role Management APIs.

API	Description
GET /apiv1/devices/:device_id/role_labels	This API retrieves a list of enabled roles for a given device.
GET /apiv1/role_labels/:id	This API retrieves a given role label and related tags.
GET /apiv1/devices/find_by_role_label	This API retrieves all devices associated with a role, with the given label (key-value pair).
PUT /apiv1/devices/:device_id/role_labels/associate	This API associates a role user to the given device.
PUT /apiv1/devices/:id/role_labels/disassociate	This API disassociates a role user from the given device.

6 Admin Device Role Management

For API code and details, log into the Development Portal. Then go to the API Browser > Role Management APIs.

API	Description
POST /apiv1/devices/:device_id/role_labels	Used to enable a role on a device. After completion, the device will accept a role user with that role to be associated.
PUT /apiv1/role_labels/:id	This API updates a role label on the given device.
DELETE /apiv1/role_labels/:name	This API disables a role on the given device.

7 Share APIs

This set of APIs allows a registered user of a device to share it with other users. The user can also create a share based on time, read/write privileges or revoke a share at any time.

For API code and details, log into the Development Portal. Then go to the API Browser > Share APIs.

API	Description
GET /api/v1/users/shares/	This API returns a list of owned shares.
GET /api/v1/users/shares/:id	This API returns a single share.
POST /api/v1/users/shares/	This API creates a new share. At the end of this operation, the share is granted to the user identified by “user_email”. An email is sent to that user about the share event. This email can be customized with an email template that can also be passed as a URL parameter.
PUT /api/v1/users/shares/:id	This API creates a new share with the provided parameters.
DELETE /api/v1/users/shares/:id	This API deletes a specified share.

9 RBAC Use Cases

9.1 Introduction to RBAC Processes

These are conventions used in RBAC processes.

9.1.1 Role Processes

Role

- Assign (assign_role.json) [Admin]

  Assign OEM::Abenika::Dealer to Role User
- Revoke (revoke_role.json) [Admin]

  This revokes the Role from Role User (if present)

Role Label

- Assign (asssign_label.json) [Admin]

  dealer_id (key) with 101 (value)
  - Assign the Role Label to Role User

    Example: {"dealer_id":"101"}
  - Revoke Role Label from Role User (revoke_label.json) [Admin]

9.1.2 Device & Role Associations

Associate Device (allow a Role User access to a device)

- Enables a Role to be associated with a device
  - Enable Role (/device/device_id/role_labels.json [Admin]
    - (OEM::Abenika::Dealer)
  - Associate Role Label with a device (/devices/device_id/role_labels/associate.json) [End User/Admin]

    (dealer_id, 101)
  - Disassociate Role Label from Device (/devices/device_id/role_labels/disassociate.json) [End User/Admin]

    (i.e., removes dealer_id, 101)
  - Removes Role User (OEM::Abenika::Dealer) and Role Label (dealer_id, 101) from Device (/role_labels/role_id.json [Admin]

Associate End User (allow a Role User to have access to the End User)

- Associate End user with Role User
  - End User self-associates or OEM::Admin does association

    /users/associate_role_user.json (OEM::Abenika::Dealer, dealer_id, 101) [End User/Admin]

- Disassociate Role User from End User
  - End User dis-associates or OEM::Admin does dis-association

    /users/disassociate_role_user.json (OEM::Abenika::Dealer, dealer_id, 101) [End User/Admin]

  Dealer can also disassociate as the End User Dealer
  - /disassociate_user.json [Role User or OEM::Admin]

9.2 Manage Role User (i.e., Dealer)

Roles are used within a company to provide special users the ability to access user and device information. Role Users are users with an assigned role and role label which assigned to specified read/write permissions based on the configuration of the role.

For this example:

- The user is **dealer@company.com**
- The user role is **OEM::Company::Dealer**
- The role label is **dealer_id** = **101**

9.2.1 Assign Role

This assigns the user dealer@company.com to the role of a dealer with an ID.

NOTE: When a dealer is created and assigned an ID, the OEM Admin can assign devices.

High Level

3. Assign dealer@company.com to the OEM::Company::Dealer **role**
4. Assign dealer@company.com to the dealer_id = 101 role label
5. Verify from a listing Role Users with OEM::Company::Dealer **role and role label**

cURL Flow

Step 1: Assign dealer@company.com to the OEM::Company::Dealer role.

This (and several other API endpoints) do not return a response when successful. For confirmation, use the -v flag and look for a successful status code. This ensures a completed action.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "user_email": "dealer@company.com",
   "role": "OEM::Company::Dealer"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_role'
```

Step 2: Assign role label dealer_id=101 to dealer@company.com

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
```

```
-d '{
   "user_email": "dealer@company.com",
   "role": "OEM::Company::Dealer",
   "label": {
      "dealer_id": "101"
   }
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_label'
```

Step 3: To verify the role assignment, list all users with the OEM::Company::Dealer **role** and the role label dealer_id=101.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/api/v1/users/index_by_role?role=OEM::Company::Deale
r&key=dealer_id&value=101'
```

The listing should include dealer@company.com as an OEM::Company::Dealer with role label dealer_id=101.

9.2.2 Revoke Role

There may be a time when a user assigned a Dealer role needs to be removed from the Dealer role and the associated permissions. This example explains the process to accomplish this action.

High Level

1. Revoke the role label dealer_id=101 from the user with OEM::Company::Dealer role
2. Revoke the OEM::Company::Dealer role from the user
3. Verify, by listing all users associated with dealer_id=101

cURL Flow

Step 1: Revoke the role label of dealer_id=101 from the user.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "user_email": "dealer@company.com",
   "role": "OEM::Company::Dealer",
   "label_key": "dealer_id"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/revoke_label'
```

Step 2: Revoke the OEM::Company::Dealer role from the user.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "user_email": "dealer@company.com",
```

```
    "role": "OEM::Company::Dealer"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/revoke_role'
```

Step 3: To verify the change, list all users with the role OEM::Company::Dealer and the role label dealer_id=101. Ensure that dealer@company.com is not on the list.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/api/v1/users/index_by_role?role=OEM::Company::Deale
r&key=dealer_id&value=101'
```

The user dealer@company.com is now removed from the role of OEM::Company::Dealer (role label dealer_id=101) and no longer has access to any assigned devices.

9.3 Manage Dealer Devices

This use case manages devices that are associated with a dealer.

- The user is **dealer@company.com**
- The user role is **OEM::Company::Dealer**.
- The role label **dealer_id** is **101**.

9.3.1 Associate Device

This process associates a device with a dealer.

High Level

1. Associate a device with a dealer role user
2. Associate dealer_id=101 with the device
3. Verify from a list of all devices associated with dealer_id = 101

cURL Flow

Step 1: Associate the dealer role with the device (device key 1234567).

```
curl -X POST -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "role_label": {
   "role_name": "OEM::Company::Dealer",
   "description": "The device dealer"
   }
}' 'https://ads-dev.aylanetworks.com/apiv1/devices/1234567/role_labels'
```

Step 2: Associate the device with the dealer with id 101 (dealer@company.com).

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
```

```
   "role_name": "OEM::Company::Dealer",
   "label": {
      "key": "dealer_id",
      "value": "101"
      }
}' 'https://ads-
dev.aylanetworks.com/apiv1/devices/1234567/role_labels/associate'
```

Step 3: To verify, list all devices associated with dealer_id=101.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/apiv1/devices/find_by_role_label?role=OEM::Company:
:Dealer&key=dealer_id&value=101'
```

The user dealer@company.com is associated with the device.

9.3.2 Disassociate Device

This example shows how to disassociate a device with a dealer.

High Level

1. Disassociate dealer_id=101 from the device
2. Disable the OEM::Company::Dealer role from the device
3. Check by listing device's role labels

cURL Flow

Step 1: Disassociate the device from the dealer with id 101.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "role_name": "OEM::Company::Dealer",
   "label": {
   "key": "dealer_id",
   "value": "101"
   }
}'
'https://ads-
dev.aylanetworks.com/apiv1/devices/1234567/role_labels/disassociate'
```

Step 2: Disable the OEM::Company::Dealer role on the device (device key = 1234567).

```
curl -X DELETE -H "Authorization: auth_token {oem_admin_token}" \
'https://ads-dev.aylanetworks.com/apiv1/role_labels/4708'
```

Step 3: To verify, list the device's enabled role labels to confirm dealer is not in the list.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-dev.aylanetworks.com/apiv1/devices/1234567/role_labels'
```

Now dealer@company.com is no longer associated with our device; hence cannot access this device.

9.4 Manage Role Users & Role Labels

This use case works with role users and role labels. For this example:

- The dealer user is dealer@company.com
  - The **dealer_id** is **101** with user role of **OEM::Company::Dealer**
- The user is user1@company.com

9.4.1 Associate User with a Role Label

In this situation, the enduser user1@company.com is associated with the user-dealer.

High Level

1. Associate enduser user1@company.com with the role user dealer@company.com
2. To verify, use one of these:
   - List all users associated with dealer=101
   - List all role users associated with enduser user1@company.com

cURL Flow

Step 1: Associate enduser user1@company.com with the role OEM::Company::Dealer and the role label dealer_id=101.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "role": "OEM::Company::Dealer",
   "label": {
      "dealer_id": "101"
      },
   "user_email": "user1@company.com"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/associate_role_user'
```

Step 2: To verify the association, list all endusers associated with dealer=101 and confirm enduser user1@company.com is listed.

Alternatively, list all role users associated with user1@company.com and confirm that dealer_id=101 is listed.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
```

```
'https://ads-
dev.aylanetworks.com/api/v1/users/index_by_label?role=OEM::Company::Deal
er&key=dealer_id&value=101'
```

OR

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/api/v1/users/role_users?user_email=user1@company.co
m'
```

Now enduser user1@company.com is associated with the role OEM::Company::Dealer and the role label dealer_id=101 (and thus with the role user dealer@company.com).

9.4.2 Disassociate User from a Role Label

In this example, the user, dealer@company.com has the role label dealer_id=101, and is associated with the enduser user1@company.com. The enduser user1@company.com is disassociated with the dealer.

High Level

1. Disassociate enduser user1@company.com from the role user dealer@company.com
2. To check, list all users associated with dealer=101. Alternatively, list all role users associated with enduser user1@company.com

cURL Flow

Step 1: Disassociate enduser user1@company.com from the role OEM::Company::Dealer and the role label dealer_id=101.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type:
application/json" \
-d '{
   "role": "OEM::Company::Dealer",
   "label_key": "dealer_id",
   "user_email": "user1@company.com"
}' 'https://ads-
dev.aylanetworks.com/api/v1/users/disassociate_role_user'
```

Step 2: Verify by listing all role users associated with dealer=101 and checking that enduser user1@company.com is not listed.

Alternatively, list all role users associated with enduser user1@company.com and checking that dealer_id=101 is not listed.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/api/v1/users/index_by_label?role=OEM::Company::Deal
er&key=dealer_id&value=101'
```

OR

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept:
application/json" \
'https://ads-
dev.aylanetworks.com/api/v1/users/role_users?user_email=user1@company.co
m'
```

Now enduser user1@company.com is no longer associated with the role OEM::Company::Dealer and the role label dealer_id=101. It is disassociated from the role user dealer@company.com.

9.5 Associate User with 2 Dealers

In this use case, an end user (user1@company.com) is associated with two role users (dealer1@company.com, dealer2@company.com), who both have the role OEM::Company::Dealer.

This is not directly possible. There is a workaround. Assign the two role users to slightly different roles and then associate both with the end user.

High Level

1. Assign dealer1@company.com to the role OEM::Company::Dealer1
2. Assign dealer1@company.com to the dealer_id of 101
3. Assign dealer2@company.com to the role OEM::Company::Dealer2
4. Assign dealer2@company.com to the dealer_id of 201
5. Associate user1@company.com with the role user dealer1@company.com
6. Associate user1@company.com with the role user dealer2@company.com
7. Confirm with a list of all role users associated with user1@company.com

cURL Flow

Step 1: Make dealer1@company.com an OEM::Company::Dealer.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "user_email": "dealer1@company.com",
   "role": "OEM::Company::Dealer1"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_role'
```

Step 2: Assign role label dealer_id=101 to dealer1@company.com

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
   "user_email": "dealer1@company.com",
   "role": "OEM::Company::Dealer1",
   "label": {
```

```
    "dealer_id": "101"
    }
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_label'
```

Step 3: Make dealer2@company.com an OEM::Company::Dealer.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
    "user_email": "dealer2@company.com",
    "role": "OEM::Company::Dealer2"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_role'
```

Step 4: Assign role label dealer_id=201 to dealer2@company.com

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
    "user_email": "dealer2@company.com",
    "role": "OEM::Company::Dealer2",
    "label": {
    "dealer_id": "201"
    }
}' 'https://ads-dev.aylanetworks.com/api/v1/users/assign_label'
```

Step 5: Associate user1@company.com with the role OEM::Company::Dealer1 and the role label dealer_id=101.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
    "role": "OEM::Company::Dealer1",
    "label": {
        "dealer_id": "101"
        },
    "user_email": "user1@company.com"
}' 'https://ads-dev.aylanetworks.com/api/v1/users/associate_role_user'
```

Step 6: Associate user1@company.com with the role OEM::Company::Dealer2 and the role label dealer_id=201.

```
curl -X PUT -H "Authorization: auth_token {oem_admin_token}" -H
"Content-type: application/json" \
-d '{
    "role": "OEM::Company::Dealer2",
    "label": {
        "dealer_id": "201"
```

```
        },
      "user_email": "user1@company.com"
  }' 'https://ads-dev.aylanetworks.com/api/v1/users/associate_role_user'
```

Step 7: To verify, list all role users associated with user1@company.com. Confirm that dealer_id=101 and dealer_id=201 are listed.

```
curl -X GET -H "Authorization: auth_token {oem_admin_token}" -H "Accept: application/json" \
'https://ads-dev.aylanetworks.com/api/v1/users/role_users?user_email=user1@company.com'
```

The user1@company.com is now associated with the role OEM::Company::Dealer1 and the role label dealer_id=101.

There is also an association with role OEM::Company::Dealer2 and the role label dealer_id=201. The user is associated with the role user dealer1@company.com and dealer2@company.com.

9.6 Associate Device with User

The user dealer@company.com wants to see and control device_1 with id 1234567 for service purposes.

Steps

1. Assign dealer@company.com the role of OEM::Company::Dealer and role label dealer_id=101 using the steps given in Assign Role.
2. Tag device with id 1234567 with the role label dealer_id=101 using the steps given in Associate Device.

Verification steps in the two sub-steps will show that the device is properly associated with dealer@company.com.

4250 Burton Drive, Santa Clara, CA 95054
Phone: +1 408 830 9844
Fax: +1 408 716 2621