API

Read through the AWS Infrastructure documentation first for better understanding.

Mobile App

Authentication

Use AWS Amplify

Amplify Framework Docs

Amplify Framework documentation - Learn how to use Amplify to develop and deploy cloud-powered mobile and web apps.

https://docs.amplify.aws/start

AMPLIFY

▼ You should **not** run any Amplify CLI commands (e.g. *amplify init*, *amplify configure,* or *amplify add*), you only need to install the library, use our current AWS configuration settings provided below, and implement the authentication using the APIs provided by Amplify (refer to the online documentation for each mobile app platform).

- **AWS Cognito Region:** eu-central-1
- **Cognito User Pool ID:** eu-central-1_ZQucC5j0b
- **Cognito User Pool Web Client ID:** 4jbvjg4135irr7h2fts2djf0vl

For Android projects, follow the steps here and choose the Manual Setup instead of the automated one. Similarly, follow the steps here and choose the Manual Setup for iOS projects. For JavaScript-based projects, refer to this section onwards.

💡 To improve our app security, you should always sign out **globally** before logging in

After a user is authenticated, Cognito will return the user session with the access and the identity token included. You need to include these tokens in the request header when accessing most of the APIs below.

REST

Register User-Device Binding

```
Name: register
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/register
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  serialNum: The serial number of the device
  signature: The digital signature of the device (in string form of the array of bytes)
Response Body: JSON String
  msg: The response message
  code: The response code
HTTP Code:
  200: User-Device binding registration process is successful
  401: The signature given in the request body is not valid
  403: The registration request is not authorized (e.g. the device is owned by other users)
```

This function should be called by the user from the mobile app to activate a newly bought Aromeo Sense device. **Response code 1** means that the device is already owned by another user and **response code 2** means that the binding of the current device and the current user has been registered before while **response code 3** means that the device signature included in the request body is invalid. **Response code 0** means the function is executed successfully. Note that the possible responses here are not the same as the ones covered in the Responses section.

▼ **Example**

```
axios.post('https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/register', {
  serialNum: "ASTESTDV2020001",
  // Some bytes of the signature are omitted to keep this short
  signature: "[12, 152, 52, 100, 76, 121, ...., 117, 103, 171, 214]",
}, {
  headers: {
    // Some parts of the authorization header are omitted to keep it short
    'Authorization': 'eyDAztU....dmeyA',
    'Content-Type': 'application/json',
  },
}).then(function(res) {
      console.log(res);
}).catch(function(err) {
    console.log(err.response.data.code + ': ' + err.response.data.msg);
});
```

Deregister User-Device Binding

```
Name: deregister
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/deregister
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  serialNum: The serial number of the device
Response Body: JSON String
  msg: The response message
  code: The response code
```

This function can only be called by the owner of the device and requires the owner to be the only controller of that device (no guest). Possible responses include response code 0, 2, and 6.

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"serialNum":"ASTESTDV2020001"}' 'https://syxvq38szb.execute-api.ap-sc
```

Get User Gender

```
Name: getGender
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/get-gender
Method: GET
Headers:
  Authorization: The identity token received from the authenticated user session
Response Body: JSON String
  gender: The gender of the user
```

This function is solely used for user profile management. The gender could be either **MALE**, **FEMALE**, or **OTHER**. The gender could also be null if the user account is new.

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X GET -H 'Authorization:eyDAztU....dmeyA' 'https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/get-gende
```

Set User Gender

```
Name: setGender
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/set-gender
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  gender: The gender of the user to be set [MALE, FEMALE, or OTHER (case sensitive)]
Response Body: JSON String
  msg: The response message
  gender: The new gender set
```

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"gender":"MALE"}' 'https://syxvq38szb.execute-api.ap-southeast-1.amaz
```

Add Device Guest

```
Name: addGuest
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/add-guest
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  serialNum: The serial number of the device
  guestEmail: The email of the guest to be added as the device controller
Response Body: JSON String
  msg: The response message
  code: The response code
```

The maximum number of controllers allowed per device is 5 users (including the owner). Response code 0 to 5 are the possible responses returned by this API.

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"serialNum":"ASTESTDV2020001","guestEmail":"abc@abc.com"}' 'https://s
```

List Device Guests

```
Name: listGuests
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/list-guests
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  serialNum: The serial number of the device the user controls
Response Body: JSON String
  msg: The response message
  code: The response code
```

If this function is executed successfully (response code 0), an array of guests (controllers of the device not including the owner) will be returned. Only the owner of the device can call this function. This API may return response code 0, 2, and 3 (See Responses section).

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"serialNum":"ASTESTDV2020001"}' 'https://syxvq38szb.execute-api.ap-sc
```

Remove Device Guest

```
Name: removeGuest
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/remove-guest
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
```

```
Request Body: JSON String
  serialNum: The serial number of the device
  guestEmail: The email of the guest to be removed from the device controllers list
Response Body: JSON String
  msg: The response message
  code: The response code
```

Only the owner of the device can call this function. Response code 0, 2, and 7 are the possible responses returned by this API.

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"serialNum":"ASTESTDV2020001","guestEmail":"abc@abc.com"}' 'https://s
```

Transfer Device Ownership

```
Name: transferOwnership
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/transfer-ownership
Method: POST
Headers:
  Authorization: The identity token received from the authenticated user session
Request Body: JSON String
  serialNum: The serial number of the device
  guestEmail: The email of the guest to whom you want to transfer the ownership
Response Body: JSON String
  msg: The response message
  code: The response code
```

Only the owner of the device can call this function. Response code 0, 2, and 7 are the possible responses returned by this API. Upon successful execution, the original owner will be a guest of the device while the guest identified as *guestEmail* will be the new owner of the device.

▼ **Example**

```
# The identity token below is not represented completely to keep it short
curl -X POST -H 'Authorization:eyDAztU....dmeyA' -d '{"serialNum":"ASTESTDV2020001","guestEmail":"abc@abc.com"}' 'https://s
```

Get Aromeo Sense Module Themes (with their Priorities)

```
Name: listSenseThemesPriorities
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/list-sense-themes-priorities
Base CDN URL: d2oylxegvg1bg.cloudfront.net/
Method: GET
Headers:
  Authorization: The identity token received from the authenticated user session
Response Body: JSON String (See below for sample response body)
```

▼ **Sample Response Body**

```
{
  "Items": [
    {
      "Priority": {
        "N": "1"
      },
      "ThemeName": {
        "S": "Chill"
      },
      "ThemeID": {
        "S": "chill"
      },
      "S3ImgKey": {
        "S": "chill-Relax/Relax_chill_preImg.jpeg"
      },
      "ThemeModule": {
        "S": "Relax"
      },
      "S3FootageKey": {
        "S": "chill-Relax/Relax_chill_portrait.mp4"
      },
      "S3SoundKey": {
        "S": "chill-Relax/Relax_chill_sound.mp3"
      }
    },{
      "Priority": {"N": "1"},
      "ThemeName": {"S": "Reflect"},
      "ThemeID": {"S": "reflect"},
      "S3ImgKey": {"S": "reflect-Sleep/Sleep_light_preImg.jpeg"},
      "ThemeModule": {"S": "Sleep"},
      "S3FootageKey": {"S": "reflect-Sleep/Sleep_light_portrait.mp4"},
      "S3SoundKey": {"S": "reflect-Sleep/Sleep_light_sound.mp3"}
    },{
      "Priority": {"N": "1"},
      "ThemeName": {"S": "Cleanse"},
      "ThemeID": {"S": "cleanse"},
      "S3ImgKey": {"S": "cleanse-Focus/Focus_rain_preImg.jpeg"},
      "ThemeModule": {"S": "Focus"},
      "S3FootageKey": {"S": "cleanse-Focus/Focus_rain_portrait.mp4"},
      "S3SoundKey": {"S": "cleanse-Focus/Focus_rain_sound.mp3"}
    }
  ],
  "Count": 3,
  "ScannedCount": 3
}
```

The response JSON is basically the very response returned by DynamoDB API. Refer to the AWS SDK reference page for more information. In the sample response above, **S** and **N** in the deepest level is the description of the data type. S means **String** while N means **Number**.

To get the URLs for these themes, append the base CDN URL (i.e. d2oylxegvg1bg.cloudfront.net/) with the keys of the objects (i.e. **S3ImgKey**, **S3SoundKey**, and **S3FootageKey**). For example, d2oylxegvg1bg.cloudfront.net/chill-Relax/Relax_chill_sound.mp3 is the URL for the sound of the **Chill** theme in **Relax** Module.

WebSocket

Establish Connection

```
Name: controlConnect
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
URL Parameter:
  token: The access token received from the authenticated user session
```

This function is used to establish a communication to the mobile app users with the Aromeo Sense devices they control. As the design of JavaScript WebSocket API does not allow developers to include HTTP headers (i.e. the Authorization header for access token), the token will be passed through the URL parameters.

▼ **Example**

```
const ws = new WebSocket('wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod?token=' + user.signInUserSession.a
```

Update Aromeo Sense State

```
Name: control
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
Message Format: JSON String
Message Content:
  serialNum: The serial number of the corresponding Aromeo Sense device
  localTime: The local time this message was sent
  payload: The desired state of the device shadow
```

This function is used when the mobile app user wants to update the Aromeo Sense state such as start diffusing, turning the light on, etc. The localTime in the message content should be formatted as discussed here. The format of the payload (i.e. the desired shadow state) should follow the format discussed here and refer to the AWS documentation on how to build the desired state object. Response code 0, 3, 8, 10, and 11 are the possible responses returned by this API.

▼ **Example**

```
var desired = {
  focus: "1,3,2,2",
  light: true,
  diffusionDuration: "3600",
  color: 18
};
var currTime = new Date();
var toBeSent = {
  "serialNum": "ASTESTDV2020001",
  "localTime": currTime.toISOString() + ' ' + (-1*currTime.getTimezoneOffset()),
  "payload": desired
};
ws.send(JSON.stringify(toBeSent));
```

Receive Aromeo Sense State

```
Name: aromeoSenseFeedback
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
Message Format: JSON String
Message Content:
  serialNum: The serial number of the corresponding Aromeo Sense device
  currState: The reported state of the device shadow
```

Every time the Aromeo Sense device a user control updates its state, the user through the mobile app WebSocket connection will receive this update. This update contains the serial number of the device to identify to which Aromeo Sense device the state belongs because a user may control more than 1 Aromeo Sense devices.

▼ **Example**

```
ws.onmessage = (event) => {
  window.console.log("WebSocket message received:", event.data);
  var data = JSON.parse(event.data);
  var serialNum = data.serialNum;
  var focusParam = data.currState.focusParam;
  var schedule = data.currState.schedule;
}
```

Receive Aromeo Sense Timer

```
Name: aromeoSenseTimer
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
Message Format: JSON String
Message Content:
  serialNum: The serial number of the corresponding Aromeo Sense device
  durationLeft: The diffusion duration left in seconds
  currentTime: The exact UTC time (in ISO format) right before the message was sent by the device
```

When an Aromeo Sense device starts diffusing, a timer of the diffusion duration left is sent to all controller users. The *currentTime* item is included in the message to achieve better precision. If needed, users may also request for the timer manually by sending a message with **{"action":"checkTimer","serialNum":"<serial_num>"}** as the content to this WebSocket connection.

▼ **Example**

```
ws.onmessage = (event) => {
  window.console.log("WebSocket message received:", event.data);
  var data = JSON.parse(event.data);
  var serialNum = data.serialNum;
  if(data.durationLeft) {
    var durationLeft = data.durationLeft;
    var timeSent = new Date(data.currentTime);
  }
}

var toBeSent = {
  "serialNum": "ASTESTDV2020001",
  "action": "checkTimer"
};
ws.send(JSON.stringify(toBeSent));
```

Ping

```
Name: aromeoSensePing and aromeoSensePong
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
Message Format: JSON String
Request Message Content:
  action: The value of this item should be "ping" (case sensitive)
  serialNum: The serial number of the device to be checked
Response Message Content:
  msg: The value returned will be "PONG" (case sensitive) if the device is connected
  serialNum: The serial number of the checked device
```

This function is used to check the connection status of the Aromeo Sense device, whether it is connected or not. Since it is used to check the connection, it will **not** return any response if the device is not connected. Therefore, the developer has to set a **timer** such that the device can be considered as **disconnected** if the timer stops. Note that the latency of this function may not be very short. Thus, the timer has to be set long enough, i.e. ~10-15 seconds.

▼ **Example**

```
ws.onmessage = (event) => {
  console.log("WebSocket message received:", event.data);
  var data = JSON.parse(event.data);
  if(data.msg === "PONG") {
    alert("Device " + data.serialNum + " is connected to the Internet");
  }
};

var toBeSent = {
  "serialNum": "ASTESTDV2020001",
  "action": "ping"
};
ws.send(JSON.stringify(toBeSent));
```

Manually Get Aromeo Sense State

```
Name: aromeoSenseGetState
Endpoint: wss://581kv6aur0.execute-api.ap-southeast-1.amazonaws.com/prod
Message Format: JSON String
Request Message Content:
  action: The value of this item should be "getState" (case sensitive)
  serialNum: The serial number of the device to be checked
```

This function is used when the mobile app user wants to get the Aromeo Sense state manually. The response can be received by using this API. Response code 0, 3, and 8 are the possible responses returned by this API.

▼ **Example**

```
var toBeSent = {
  "serialNum": "ASTESTDV2020001",
  "action": "getState",
};
ws.send(JSON.stringify(toBeSent));
```

Aromeo Sense

REST

Request for IoT Server Provision

```
Name: iotCoreProvision
Endpoint: https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/iot-core-provision
Method: POST
Request Body: JSON String
  serialNum: The serial number of the device
  signature: The digital signature of the device (in string form of the array of bytes)
Response Body: JSON String
  msg: The response message
  payload: The payload object of the response containing the following items
    endpointAddress: The address of the IoT Core server the device can connect to
    endpointRegion: The region of the IoT Core server provided
    certificatePem: The certificate required to connect to the IoT Core server
    privateKey: The private key required to connect to the IoT Core server
```

This function can only be invoked if the Aromeo Sense device has at least 1 controller user. Therefore, the user-device binding registration should be invoked first for newly bought devices. The response body items mentioned above are the important items. There are other items returned by the server as well but they can safely be ignored.

💡 Note that the **private key** and the **certificate** will be **given** only **once** when the device is given provision for a particular IoT Core server the first time. Therefore, the certificate and the private key should be kept safe once received.

▼ **Example**

```
axios.post('https://syxvq38szb.execute-api.ap-southeast-1.amazonaws.com/prod/iot-core-provision', {
  serialNum: "ASTESTDV2020001",
  // Some bytes of the signature are omitted to keep this short
  signature: "[12, 152, 52, 100, 76, 121, ...., 117, 103, 171, 214]",
}, {
  headers: {
    'Content-Type': 'application/json',
  },
}).then(function(res) {
  const endpoint = res.data.payload.endpointAddress;
  const region = res.data.payload.endpointRegion;
  if(res.data.payload.certificatePem) {
    const data = new Uint8Array(Buffer.from(res.data.payload.certificatePem));
    fs.writeFileSync('./certs/' + region + '-cert.pem', data);
    const data2 = new Uint8Array(Buffer.from(res.data.payload.privateKey));
    fs.writeFileSync('./certs/' + region + '-pk.pem', data2);
    console.log("Credentials saved");
  } else {
    console.log("This device has been provisioned before");
  }
  console.log("IoT Core Endpoint Address: " + endpoint);
  console.log("IoT Core Region: " + region);
}).catch(function(err) {
  console.log(err);
  if(err) console.log(err.response.data.msg);
});
```

MQTT

Use aws-iot-device-sdk-embedded-c

aws/aws-iot-device-sdk-embedded-C

We have released version 4.0.0 beta 1 of this SDK on the v4_beta branch and encourage everyone to give it a try. Version 4 is a new design, and therefore NOT backwards compatible with version 3.0.1. We will continue to fix bugs in v3.0.1 even after v4.0.0 is released, but we may not add new features to

https://github.com/aws/aws-iot-device-sdk-embedded-C

Device shadow service data flow

The Device Shadow service acts as an intermediary, allowing devices and applications to retrieve and update a device's shadow. To illustrate how devices and applications communicate with the Device Shadow service, this section walks you through the use of the AWS IoT MQTT client and the AWS CLI to simulate communication between an internet-connected light bulb, an application, and the Device Shadow service.

https://docs.aws.amazon.com/iot/latest/developerguide/device-shadow-data-flow.html

Refer to the format of the device shadow state and the topics to be used.

Connection

```
Topic Name: <serial number>/connection
Message Format: JSON String
Subscribe Message Content:
  check: The value will be "PING" (case sensitive)
Publish Message Content:
  check: The value should be "PONG" (case sensitive)
```

Whenever the device receives a message sent to the *connection* topic (i.e. {"check":"PING"}), the device has to publish a JSON string (i.e. {"check":"PONG"}) to the same topic to indicate that this device is connected to the server.

Timer

```
Topic Name: <serial number>/timer
Message Format: JSON String
Publish Message Content:
  durationLeft: The diffusion duration left in seconds
  currentTime: The exact UTC time (in ISO format) right before the message is going to be sent
```

Whenever an Aromeo Sense device starts diffusing, it must send this message. The exact time right before the device sends the message should also be included in the message for better precision. The message should be formatted as **{"durationLeft":XXXX,"currentTime":<UTC Time in ISO Format>}** with **XXXX** is the diffusion duration left in seconds. For example, **{"durationLeft":1800,"currentTime":"2020-03-07T11:10:43.515Z"}** is a correct message content stating that the diffusion duration is 30 minutes left at 11:10:43 7 March 2020 UTC.

In addition, the device also needs to send this message whenever a mobile app makes a request to get the duration info. The mobile app request will also be sent to this topic with the message content formatted as **{“action”:”checkTimer”}.**

If the device is currently diffusing when the user presses the manual diffusion button or a schedule starts, the durationLeft and currentTime has to be reset to the new duration left based on the new diffusion duration.

Responses

Below are the possible API responses mentioned before, including the response message, response code, and the HTTP code.

Responses

msg	code	HTTP Code
Success	0	200
The number of controllers of this device has reached its maximum	1	403
Unauthorized: Current user is not the owner of this device	2	403
Device is either not registered or not activated	3	403
Guest user is already a controller of this device	4	403
Current user is not found	5	403
You need to remove all guests first before deregistering	6	403
The email provided is not the guest of this device	7	403
User is not a controller of this device	8	403
Not a valid local time	10	400
Schedule has collision(s)	11	400