Evaluate IoT

These topics let you quickly get a test or demo environment running. They demonstrate how to configure ForgeRock® Access Management and run the IoT SDK and IoT Gateway examples.

About IoT

How ForgeRock IoT can help you register, authenticate, and authorize your IoT ecosystem.

Requirements

Install the prerequisite software and get the examples.

Register identities

Manually register an identity in AM, for a thing or the IoT gateway.

SDK examples

Use the SDK examples to authenticate and request an access token for a thing.

Gateway examples

Use the gateway examples to start and authenticate the IoT Gateway and connect a thing to it.

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

About IoT

Things are physical objects that can connect with each other, and with other systems through the Internet, without human intervention. Examples include smart home devices, such as window sensors and door locks, smart TVs, health and fitness monitors, and road and speed sensors.

To participate in a connected system, a thing needs an *identity* that it uses to authenticate. ForgeRock IoT enables dynamic registration of things with identities, without human intervention.

As soon as things connect to a network, they become a security concern. You need to be able to *trust* and *monitor* the things that are connected to your network, and accessing your services or APIs. The ForgeRock® Identity Platform, including ForgeRock IoT, provides standards-based authorization using the OAuth 2.0 authorization framework. It gives you a single view of all the identities in your system—customers, employees, things, and the relationships between them. ForgeRock IoT also lets you manage offline and constrained devices, and delivers identities to things at the *edge* of your network, where the data is being generated.

About ForgeRock IoT

ForgeRock IoT includes two components:

IoT SDK

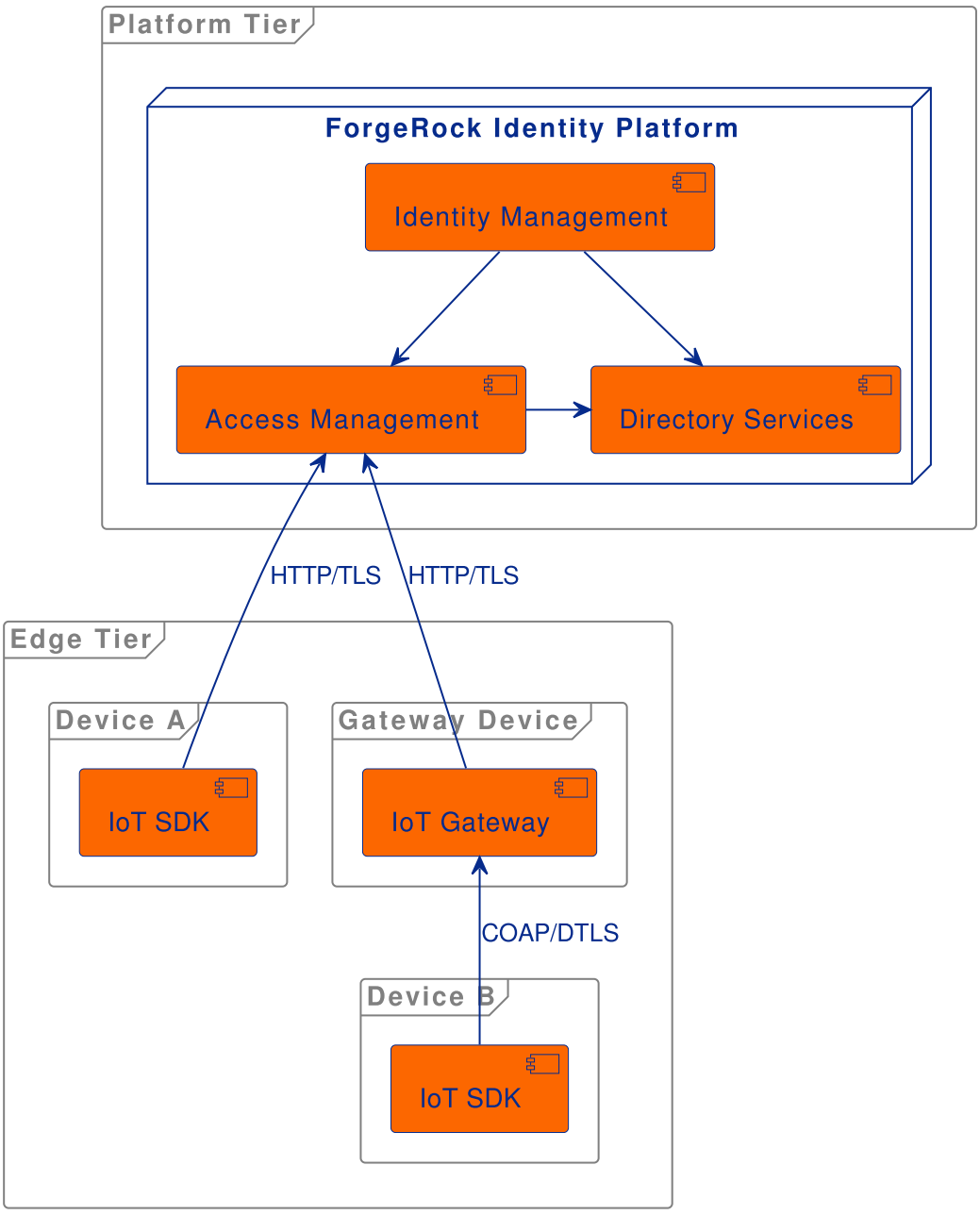
The IoT SDK lets a thing (either a physical device or a software service) register and authenticate without human interaction. When the thing is registered, it is represented by a digital identity in the ForgeRock Identity Platform, It can then authenticate itself to interact with the platform tier.

The IoT SDK can communicate directly with the platform, using HTTP(S), or through the IoT Gateway, using the Constrained Application Protocol (CoAP(S)).

IoT Gateway

The IoT Gateway is an application that lets *constrained devices* interact with the ForgeRock Identity Platform, by acting as a proxy between a thing and the Platform. A constrained device is usually a small device with limited CPU, memory, and power resources (such as sensors, smart objects, and smart devices).

This diagram shows the ForgeRock IoT architecture and components:



Prerequisites

These topics cover what you need to do before evaluating ForgeRock IoT.

Install the required software

Download the following software before you evaluate ForgeRock IoT, and test the examples:

- Go, version 1.21 or later.
- Git (to download the source code and run the examples).

Get the examples

1. Clone the `iot-edge` Git repository:

   ```
   git clone https://github.com/ForgeRock/iot-edge.git
   ```

   This command creates a directory named `iot-edge`.

2. Change to the `iot-edge` directory:

   ```
   cd /path/to/iot-edge
   ```

   The examples assume that this is your current working directory.

3. The examples also assume that you are working with version 7.4.0 of the code.

   Check out the `release/v7.4.0` branch:

   ```
   git checkout release/v7.4.0
   ```

Install and configure AM

Before you start, read the Evaluation topics in the AM documentation to set up an AM instance, with a *default configuration*.

The examples in this guide assume the following:

- AM is installed with the fully qualified domain name `am.localtest.me`, in a Tomcat container, listening on port `8080`.

  To configure AM, go to `http://am.localtest.me:8080/openam/`.

- AM is configured with the default configuration, with user `amAdmin` and password `changeit`.

When you have set up a default AM instance, log into the AM admin UI as user `amAdmin` with password `changeit`.

1. Add an IoT service.

   The IoT service configures the identity store, adding the required thing attributes to AM users (for all LDAPv3ForOpenDS and LDAPv3ForForgeRockIAM stores in the realm). For more information about this service, refer to IoT service in the *AM Reference*:

   - In the Top Level Realm, select **Services > Add a Service > IoT Service**, and click **Create**.

- Enable **Create OAuth 2.0 Client**.

  The IoT service creates an OAuth 2.0 Client with the given name and default configuration required to serve as the client for this service. The client is created without any scope(s), and is used by default for all things that request access tokens.

  ▼ *Advanced use*

  If a thing (or group of things) needs a client with different configuration to the default, you can create a custom client here, and add its name to the thing's `thingOAuth2ClientName` profile attribute.

- Enable **Create OAuth 2.0 JWT Issuer**.

  The service creates a Trusted JWT Issuer with the given name and default configuration required for the IoT Service to act as the Issuer when handling requests for access tokens.

  ▼ *Advanced use*

  If you configure the client manually, the JWT issuer must have the following settings:

  - `JWT Issuer: forgerock-iot-service`
  - `Consented Scopes Claim: scope`
  - `Resource Owner Identity Claim: sub`

  The signing/verification key used by this issuer is configured in the secrets store under `am.services.iot.jwt.issuer.signing`. It must use the HS256 algorithm.

- Click **Save Changes**.

2. Add an OAuth 2.0 provider service.

   The Top Level Realm includes and OAuth 2.0 provider service by default. If you are using a different realm, select **Services > Add a Service > OAuth2 Provider**, and click **Create**.

   ▼ *Advanced use*

   If your service will use the *introspection* feature of the SDK, change the following settings:

   - On the **Core** tab, enable **Use Client-Side Access & Refresh Tokens**.
   - On the **Advanced** tab, select an asymmetric key for the **OAuth2 Token Signing Algorithm**.

3. Configure the IoT OAuth 2.0 client:

   - Go to **Applications > OAuth 2.0 > Clients** and select the `forgerock-iot-oauth2-client`.

- In the **Scope(s)** field, add `publish` and `subscribe`.
- Save your work.

▼ *Advanced use*

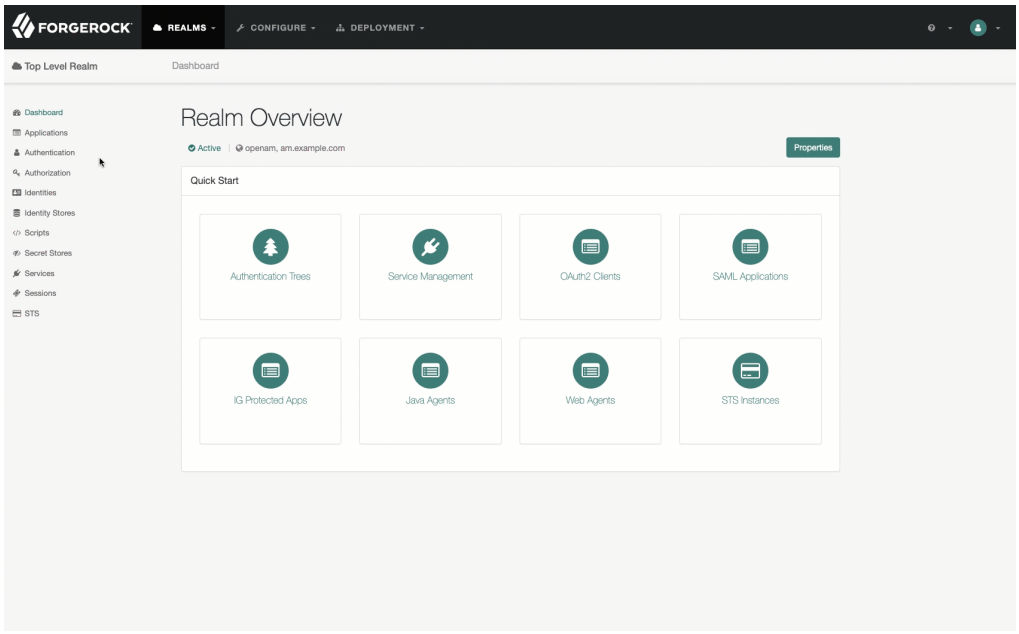
If you create your own OAUth2 client here, make sure that the client contains the `JWT Bearer`, `Device Code` and `Refresh Token` grant types, and has a strong generated password.

4. Create the following authentication trees:

a. A tree that handles authentication only, named `auth-tree`.

- Go to **Authentication > Trees** and click **Create Tree**.
- Type `auth-tree` in the **Name** field, and click **Create**.
- Add an `Authenticate Thing` node, with the default settings, and save your work.

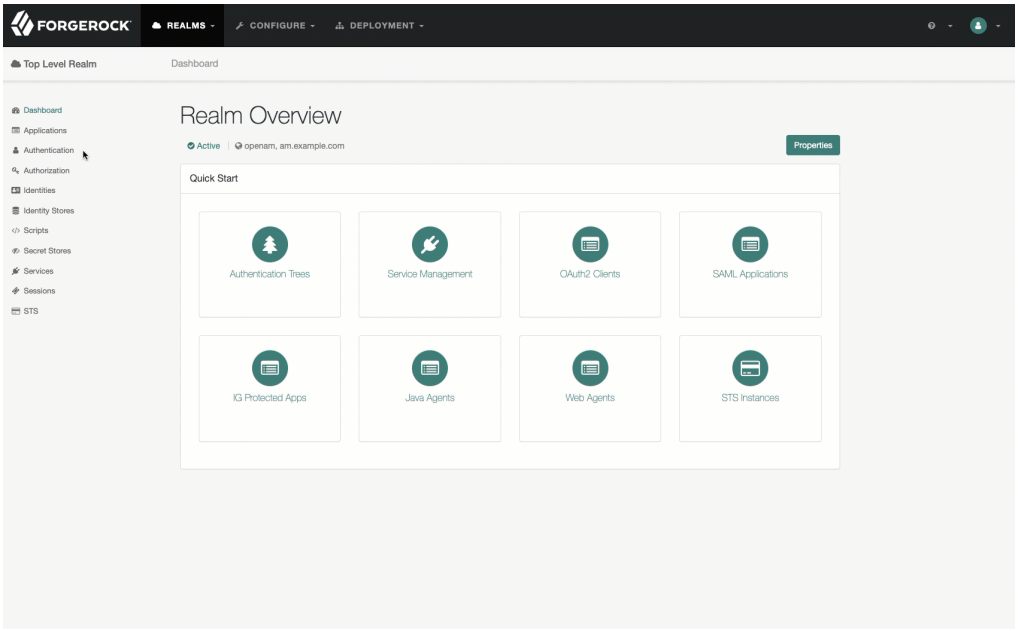
▼ *Show me*



b. A tree that handles registration only, named `reg-tree`:

- Go to **Authentication > Trees** and click **Create Tree**.
- Type `reg-tree` in the **Name** field, and click **Create**.
- Add a `Register Thing` node, and enable **Create Identity** on that node.
- Save your work.

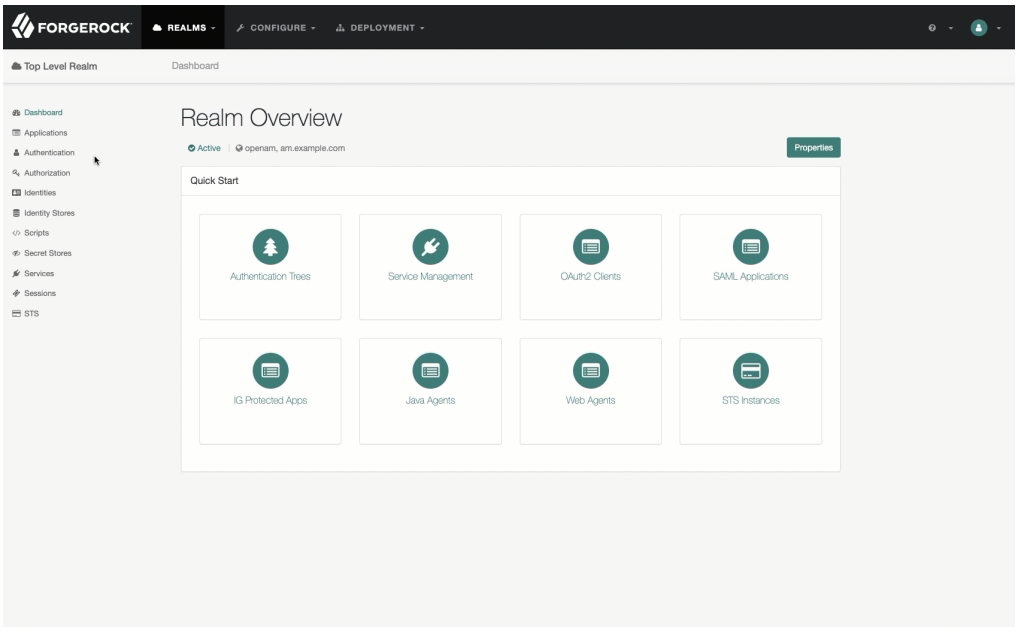
▼ *Show me*



c. A tree that handles authentication and registration, named `auth-reg-tree`:

- Go to **Authentication** > **Trees** and click **Create Tree**.
- Type `auth-reg-tree` in the **Name** field, and click **Create**.
- Add an `Authenticate Thing` node and a `Register Thing` node.
- On the `Register Thing` node, enable **Create Identity**.
- Save your work.

▼ *Show me*



5. Add a secret ID mapping.

- Go to **Configure** > **Secret Stores** and select the `default-keystore`.
- On the **Mappings** tab, click **Add Mapping**.
- In the **Secret ID** list, select `am.services.iot.cert.verification`.
- In the **Alias** field, type `es256test` and click **Add**.

This mapping indicates which key the Register Thing node should use when verifying the registration certificate. The CA certificate in this example (`es256test`) is one of the test certificates included by default in AM.

- Click **Create** to add the mapping.

For more information about mapping secret IDs, refer to Map and rotate secrets in the AM documentation.

6. Create a software publisher agent.
   - Go to **Applications > OAuth 2.0 > Software Publisher** and click **Add Software Publisher Agent**.
   - Enter these settings, then click **Create**:
     - **Agent ID**: `iot-software-publisher`
     - **Software publisher secret**: Leave blank
     - **Software publisher issuer**: `https://soft-pub.example.com`
   - Enter these settings, then save your work:
     - **Software statement signing Algorithm**: `ES256`
     - **Public key selector**: `JWKs`
     - **Json Web Key**: `{"keys":[{"use":"sig","kty":"EC","kid":"gLcQhotEZygUuVUrt3Z6azql3dVfqQS7lo3vereyU7Y=","crv":"P-256","alg":"ES256","x":"IUuXjru5zb3ixx23uM-qYsFX47eQNWJ6jTkHudFpVr4","y":"VDSoP-7XBc8KLSeVb2fwzg36458AV3a8MrBx1RZHNho"}]}`

Register identities

You can register identities in AM manually, over REST, or dynamically during the authentication process.

These examples show how to register identities manually. Dynamic registration is covered in the IoT SDK examples and IoT Gateway examples:

1. Before you can register an identity, get an admin SSO token from AM as follows:

```
curl \
--header 'Content-Type: application/json' \
--header 'X-OpenAM-Username: amAdmin' \
--header 'X-OpenAM-Password: changeit' \
--header 'Accept-API-Version: resource=2.0, protocol=1.0' \
--request POST \
'http://am.localtest.me:8080/openam/json/authenticate'
```

```
{
    "tokenId": "yLiS5J55N…lMxAAA.*",
    "successUrl": "/openam/console",
    "realm": "/"
}
```

2. Save the `tokenId` returned in this request as a variable, for example:

```
export tokenId=yLiS5J55N…lMxAAA.*
echo $tokenId
yLiS5J55N…lMxAAA.*
```

3. Set the `ID` of the thing or gateway you are registering as a variable. The examples use `manual-thing` and `manual-gateway` as IDs:

   ▼ *Set the ID for a thing*

   ```
   export ID=manual-thing
   echo $ID
   manual-thing
   ```

   ▼ *Set the ID for a gateway*

   ```
   export ID=manual-gateway
   echo $ID
   manual-gateway
   ```

4. Register an identity for the thing or gateway. These examples set a number of sample fields (`thingKeys`) for the thing or gateway you are registering:

   ▼ *Register a thing*

   ```
   curl \
   --request PUT \
   --header 'Content-Type: application/json' \
   --header 'Accept-Api-Version: resource=4.0, protocol=2.1' \
   --cookie "iPlanetDirectoryPro=${tokenId}" \
   --data '{
     "userPassword": "5tr0ngG3n3r@ted",
     "thingType": "device",
     "thingKeys": "{\"keys\":
   [{\"use\":\"sig\",\"kty\":\"EC\",\"kid\":\"cbnztC8J_l2feNf0a
   TFBDDQJuvrd2JbLPoOAxHR2N8o=\",\"crv\":\"P-
   256\",\"alg\":\"ES256\",\"x\":\"wjC9kMzwIeXNn6lsjdqplcq9aCWp
   AOZ0af1_yruCcJ4\",\"y\":\"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq
   04lBmfEHw\"}]}"
   ```

```
}' \
"http://am.localtest.me:8080/openam/json/realms/root/users/${ID}"
{
  "_id": "manual-thing",
  "_rev": "-1",
  "realm": "/",
  "username": "manual-thing",
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "fr-iot",
    "sunFMSAML2NameIdentifier",
    "inetorgperson",
    "devicePrintProfilesContainer",
    "pushDeviceProfilesContainer",
    "iPlanetPreferences",
    "iplanet-am-user-service",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "oathDeviceProfilesContainer",
    "person",
    "webauthnDeviceProfilesContainer",
    "sunAMAuthAccountLockout",
    "deviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn": [
    "uid=manual-
thing,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "manual-thing"
  ],
  "thingKeys": [
    "{\"keys\":
[{\"use\":\"sig\",\"kty\":\"EC\",\"kid\":\"cbnztC8J_l2feNf0a
TFBDDQJuvrd2JbLPoOAxHR2N8o=\",\"crv\":\"P-
256\",\"alg\":\"ES256\",\"x\":\"wjC9kMzwIeXNn6lsjdqplcq9aCWp
AOZ0af1_yruCcJ4\",\"y\":\"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq
04lBmfEHw\"}]}"
  ],
  "createTimestamp": [
```

```
    "20220629131020Z"
  ],
  "uid": [
    "manual-thing"
  ],
  "universalid": [
    "id=manual-thing,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "sn": [
    "manual-thing"
  ],
  "thingType": [
    "device"
  ]
}
```

Log in to the AM admin UI and select **Identities** in the Top Level Realm. You should see the `manual-thing` in the list.

▼ *Register a gateway*

```
curl \
--request PUT \
--header 'Content-Type: application/json' \
--header 'Accept-Api-Version: resource=4.0, protocol=2.1' \
--cookie "iPlanetDirectoryPro=${tokenId}" \
--data '{
 "userPassword": "5tr0ngG3n3r@ted",
 "thingType": "gateway",
    "thingKeys": "{\"keys\":
[{\"use\":\"sig\",\"kty\":\"EC\",\"kid\":\"cbnztC8J_l2feNf0a
TFBDDQJuvrd2JbLPoOAxHR2N8o=\",\"crv\":\"P-
256\",\"alg\":\"ES256\",\"x\":\"wjC9kMzwIeXNn6lsjdqplcq9aCWp
AOZ0af1_yruCcJ4\",\"y\":\"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq
04lBmfEHw\"}]}"
}' \
"http://am.localtest.me:8080/openam/json/realms/root/users/${ID}"
{
  "_id": "manual-gateway",
  "_rev": "-1",
  "realm": "/",
```

```
  "username": "manual-gateway",
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "fr-iot",
    "sunFMSAML2NameIdentifier",
    "inetorgperson",
    "devicePrintProfilesContainer",
    "iplanet-am-user-service",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "person",
    "sunAMAuthAccountLockout",
    "oathDeviceProfilesContainer",
    "webauthnDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service",
    "deviceProfilesContainer"
  ],
  "dn": [
    "uid=manual-
gateway,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "manual-gateway"
  ],
  "thingKeys": [
    "{\"keys\":
[{\"use\":\"sig\",\"kty\":\"EC\",\"kid\":\"cbnztC8J_l2feNf0a
TFBDDQJuvrd2JbLPoOAxHR2N8o=\",\"crv\":\"P-
256\",\"alg\":\"ES256\",\"x\":\"wjC9kMzwIeXNn6lsjdqplcq9aCWp
AOZ0af1_yruCcJ4\",\"y\":\"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq
04lBmfEHw\"}]}"
  ],
  "createTimestamp": [
    "20200826104156Z"
  ],
  "uid": [
    "manual-gateway"
  ],
  "universalid": [
    "id=manual-
```

```
gateway,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "sn": [
    "manual-gateway"
  ],
  "thingType": [
    "gateway"
  ]
}
```

Log in to the AM admin UI and select **Identities** in the Top Level Realm. You should see the `manual-gateway` in the list.

IoT SDK examples

The IoT SDK examples demonstrate how to:

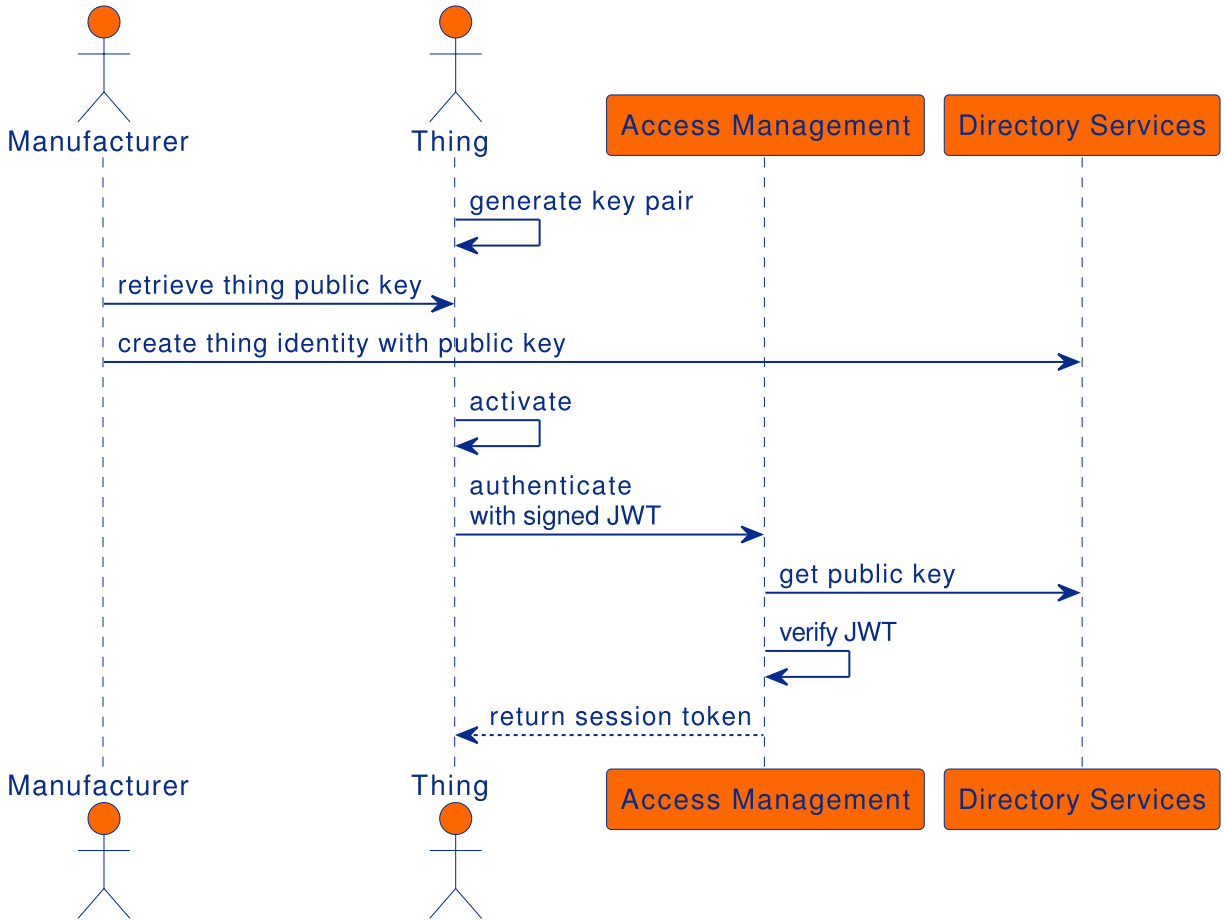
- Authenticate a thing after manual registration
- Authenticate a thing with dynamic registration
- Request a user token for an authenticated thing

These examples assume that you have downloaded the example repository and that the `iot-edge` directory is your current directory.

Authenticate a thing after manual registration

This example authenticates a thing and requests an access token for the thing. The thing must have an asymmetric key pair for signing. This is provided in the `/path/to/iot-edge/examples/resources` directory. The source code for this example is in `/path/to/iot-edge/examples/thing/simple/main.go`.

This sequence diagram shows how the thing is authenticated for the session:



Before you run the example, register the thing manually (using `manual-thing` as the thing's ID).

Then, run the `thing/manual-registration` example:

```
cd /path/to/iot-edge
./run.sh example "thing/manual-registration" \
-name "manual-thing" \
-url "http://am.localtest.me:8080/openam" \
-tree "auth-tree"
Creating Thing manual-thing… Done
Requesting access token… Done
Access token: uk6gfxEwO4Qq8pDaijcD9ssmyqk
Expires in: 3599
Scope(s): [publish]
```

The thing is now authenticated to AM and has received an access token.

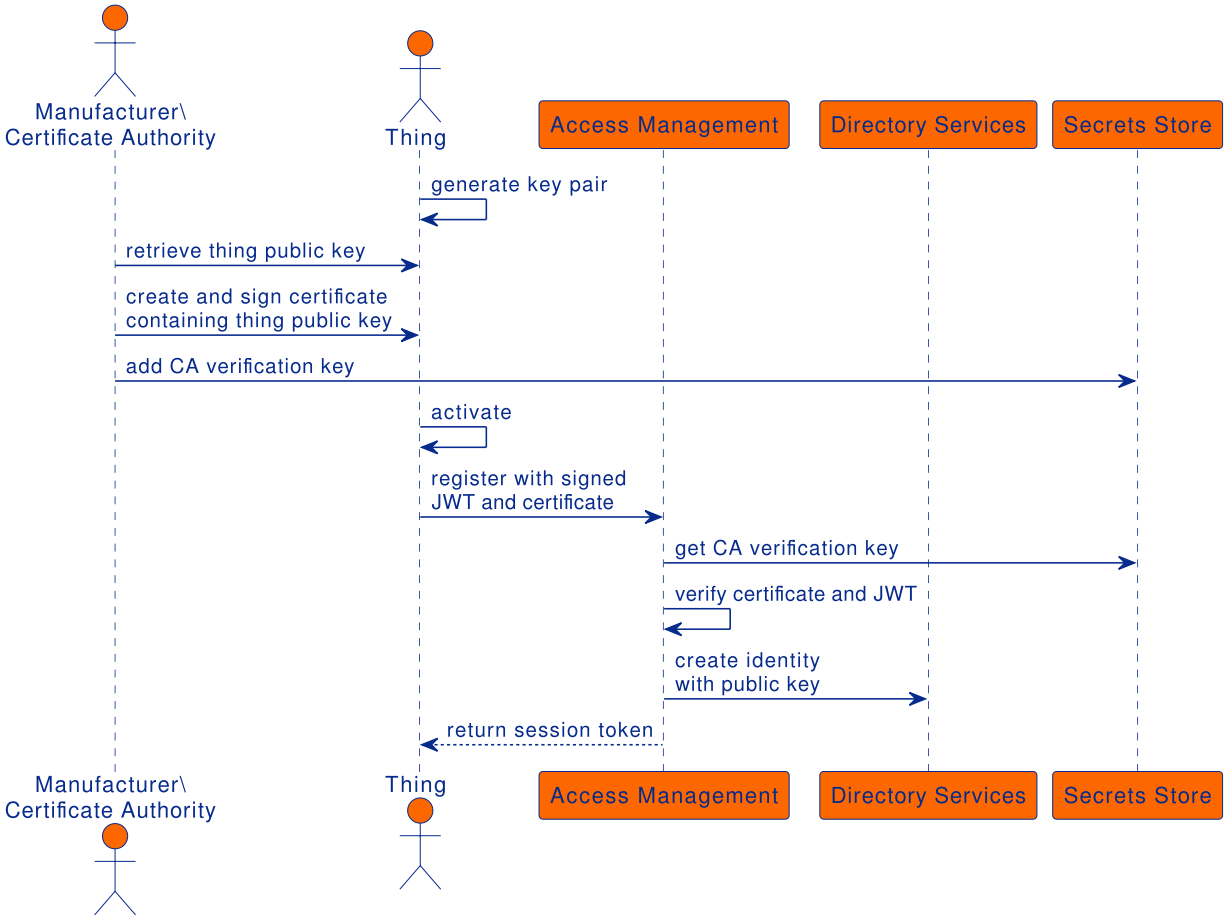
Authenticate a thing with dynamic registration

The Register Thing node supports multiple registration methods, specified in the node's **JWT Registration Method** property. These examples show the different methods.

Proof-of-possession and certificate

This example registers a new identity, authenticates the thing, and requests an access token for the thing. The thing must have an asymmetric key pair for signing, and a CA-signed X.509 certificate that contains the key pair's public key. These are provided in the `/path/to/iot-edge/examples/resources` directory. The source code for this example is in `/path/to/iot-edge/examples/thing/dynamic-registration/pop-cert/main.go`.

This sequence diagram shows how the thing is registered and authenticated for the session:



The example assumes the following node configuration in the `auth-reg-tree`:

- Authenticate Thing node
  - **JWT Authentication Method**: `Proof of Possession`
  - **Issue Restricted Token** enabled
- Register Thing node
  - **JWT Registration Method**: `Proof of Possession & Certificate`
  - **Create Identity** enabled

From the `iot-edge` directory, run the `thing/dynamic-registration/pop-cert` example:

```
cd /path/to/iot-edge
./run.sh example "thing/dynamic-registration/pop-cert" \
-name "dynamic-thing" \
-url "http://am.localtest.me:8080/openam" \
-tree "auth-reg-tree"
Creating Thing dynamic-thing… Done
Requesting access token… Done
Access token: 9szFZrb006z1L7KF_dJCLNMsVPw
Expires in: 3599
Scope(s): [publish]
```

The thing is now registered with the ID `dynamic-thing`. It's authenticated to AM and has received an access token.

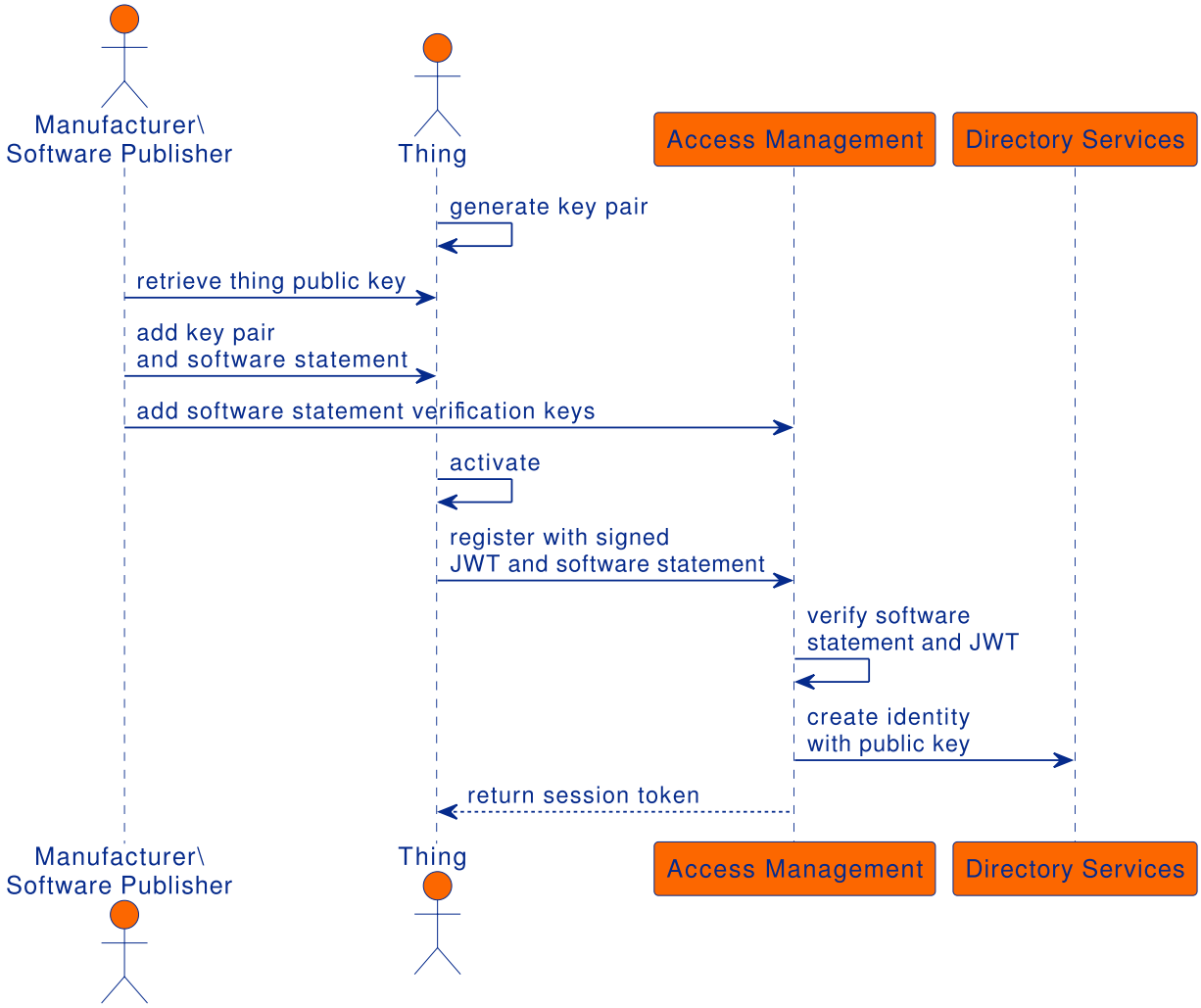
Log in to the AM admin UI and select **Identities** in the Top Level Realm to locate the `dynamic-thing` in the list.

Proof-of-possession and software statement

This example registers a new identity, authenticates the thing, and requests an access token for the thing. The thing must have an asymmetric key pair for signing, and a software statement that contains the key pair's public key in the `jwks` claim.

The source code for this example is in `/path/to/iot-edge/examples/thing/dynamic-registration/pop-sw-stmt/main.go`.

This sequence diagram shows how the thing is registered and authenticated for the session:



Change the registration method in the Register Thing node. The example assumes the following node configuration in the `auth-reg-tree`:

- Authenticate Thing node
  - **JWT Authentication Method**: `Proof of Possession`
  - **Issue Restricted Token** enabled
- Register Thing node
  - **JWT Registration Method**: `Proof of Possession & Software Statement`
  - **Create Identity** enabled

If you have already run the Proof-of-possession and certificate example, delete the `dynamic-thing` identity in the AM admin UI before you run this example.

From the `iot-edge` directory, run the `thing/dynamic-registration/pop-sw-stmt` example:

```
cd /path/to/iot-edge
./run.sh example "thing/dynamic-registration/pop-sw-stmt" \
-name "dynamic-thing" \
-url "http://am.localtest.me:8080/openam" \
```

```
-tree "auth-reg-tree"
Creating Thing dynamic-thing… Done
Requesting access token… Done
Access token: RXDEVQBY6YgZNnX07FEtJRKd_Sg
Expires in: 3599
Scope(s): [publish]
```

The thing is now registered with the ID `dynamic-thing`. It's authenticated to AM and has received an access token.

Log in to the AM admin UI and select **Identities** in the Top Level Realm to locate the `dynamic-thing` in the list.

Software statement

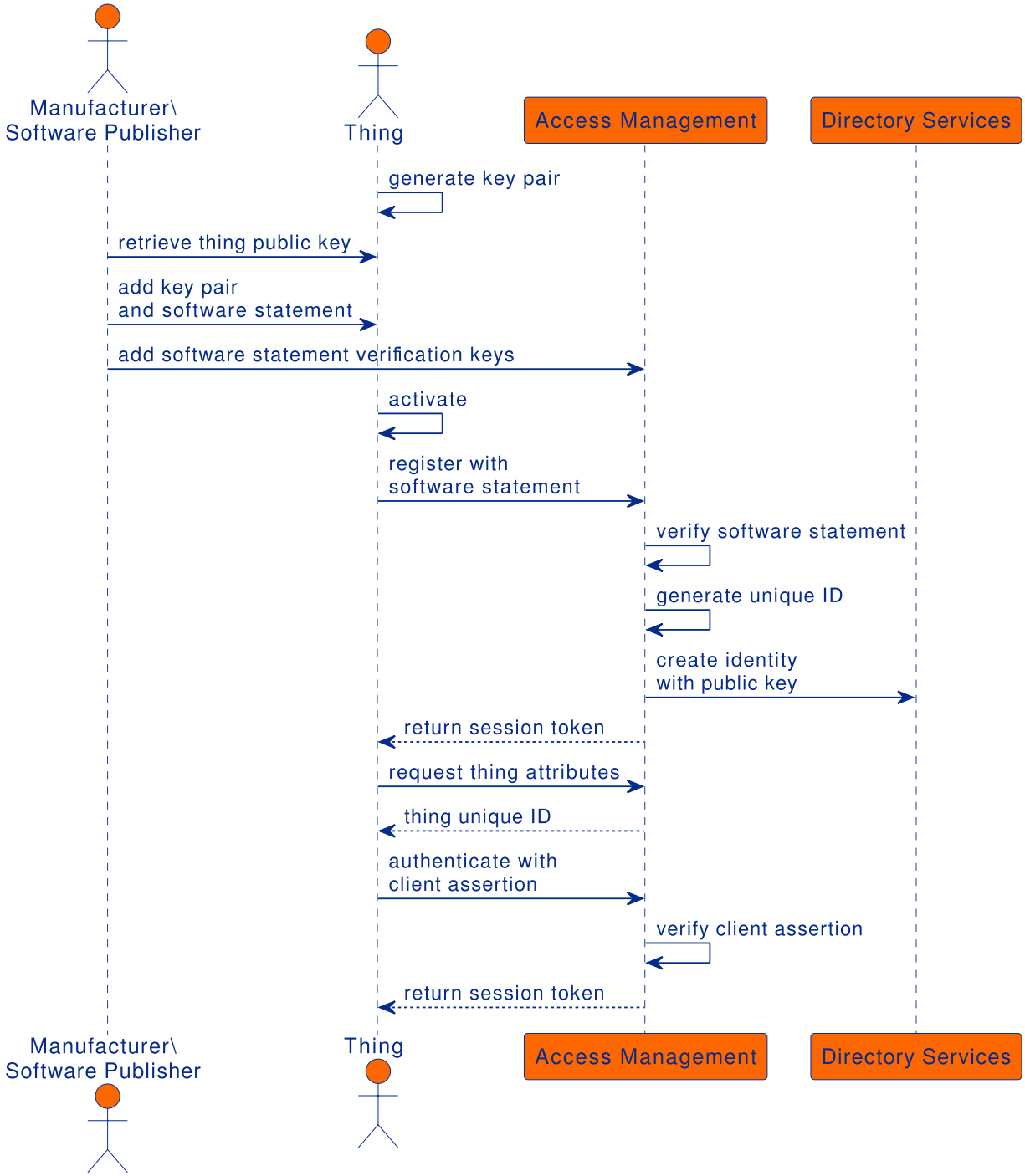
This example registers a thing with a *unique ID*, rather than a specified name. After registration, the ID is retrieved and used to authenticate the thing. When the thing is authenticated, the flow requests an access token for the thing.

The flow mimics OAuth 2.0 Dynamic Registration and OAuth 2.0 JWT Bearer Authentication to request an access token, using a standard API.

The thing must have an asymmetric key pair for signing, and a software statement that contains the key pair's public key in the `jwks` claim.

The source code for this example is in `/path/to/iot-edge/examples/thing/dynamic-registration/sw-stmt/main.go`.

This sequence diagram shows how the thing is registered and authenticated for the session:



Change the configuration of the `auth-tree` and the `reg-tree` as follows:

- In the `auth-tree` set these values on the Authenticate Thing node:
  - **JWT Authentication Method** : `Client Assertion`
  - **Issue Restricted Token** disabled
- In the `reg-tree` set these values on the Register Thing node:
  - **JWT Registration Method** : `Software Statement`
  - **Create Identity** enabled
  - Under **Default Attribute Values**, click **Add**, then set `thingType` as the **KEY** and `device` as the **VALUE**

From the `iot-edge` directory, run the `thing/dynamic-registration/sw-stmt` example:

```
cd /path/to/iot-edge
./run.sh example "thing/dynamic-registration/sw-stmt" \
-url "http://am.localtest.me:8080/openam" \
-audience
"http://am.localtest.me:8080/openam/oauth2/access_token" \
-reg-tree "reg-tree" \
-auth-tree "auth-tree"
Register thing using Software Statement… Done
Requesting attributes… Done
Attributes:  {map[_id:862b8345-d9cd-4931-911a-e8743c3d28cb
thingConfig:[] thingKeys:[{"keys":
[{"kty":"EC","kid":"veziPUQYgKIj0GTML2e2A4epDK_hfFqBZvAJhNzOjYs="
,"use":"sig","alg":"ES256","x":"n3DAs6v4YF3t0SzlV4wtRambjLBR4hige
hgBuMpSf00","y":"sP7JHsDIlF3W334wgSHl9rxSbN1TMg_tYOU9lUC2l1A","cr
v":"P-256"}]}]]}
Authenticate thing using Client Assertion… Done
Requesting access token… Done
Access token: UOVU_dN04HiEWiqd-1P0Fn4QGmY
Expires in: 3599
Scope(s): [publish]
```

The thing is now registered with the ID `dynamic-thing`. It's authenticated to AM and has received an access token.

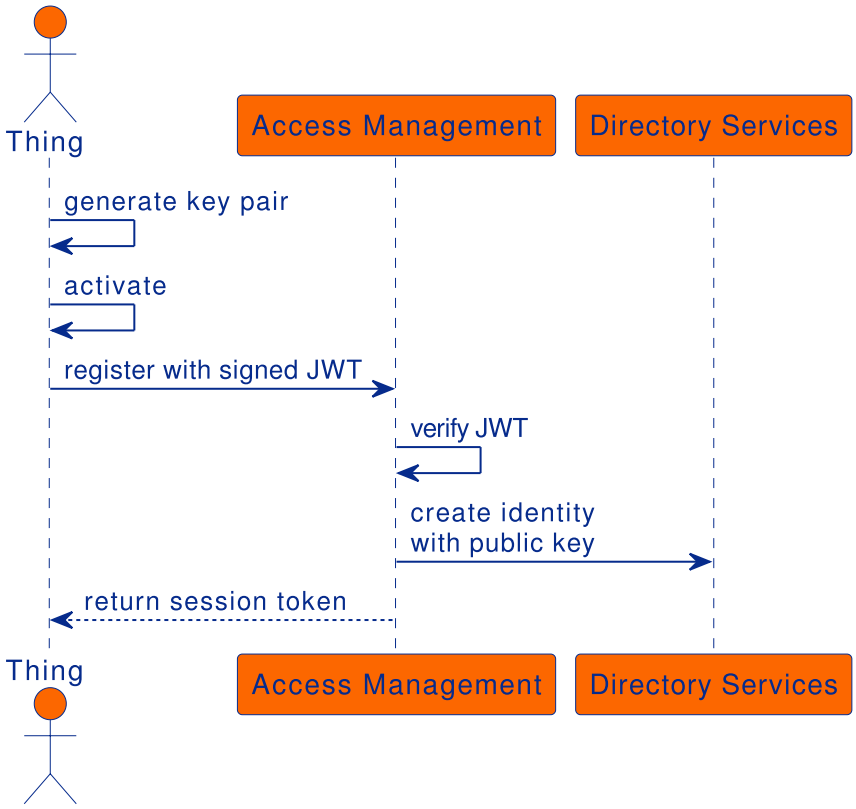
Log in to the AM admin UI and select **Identities** in the Top Level Realm to locate the new identity in the list. The identity will have a unique ID, such as `aa3373b1-93ef-49d0-bbbf-b95dcab6691b`.

Proof-of-possession

This example creates a new identity with a specified name, authenticates the thing, and requests an access token for the thing. The thing must have an asymmetric key pair for signing. No trusted third party is used in this flow, so you should use this registration method only if the thing that's registering is already trusted.

The source code for this example is in `/path/to/iot-edge/examples/thing/dynamic-registration/pop/main.go`.

This sequence diagram shows how the thing is registered and authenticated for the session:



In the `auth-reg-tree`, change the configuration as follows:

- Authenticate Thing node
  - **JWT Authentication Method**: `Proof of Possession`
  - **Issue Restricted Token** enabled
- Register Thing node
  - **JWT Registration Method**: `Proof of Possession`
  - **Create Identity** enabled

If you have already run a previous registration example that created a named identity, delete the `dynamic-thing` identity in the AM admin UI before you run this example.

From the `iot-edge` directory, run the `thing/dynamic-registration/pop` example:

```
cd /path/to/iot-edge
./run.sh example "thing/dynamic-registration/pop" \
-name "dynamic-thing" \
-url "http://am.localtest.me:8080/openam" \
-tree "auth-reg-tree"
Creating Thing dynamic-thing… Done
Requesting access token… Done
Access token: n3cYnvOSq15s2wzSwKyTuQbkoD8
Expires in: 3599
Scope(s): [publish]
```

The thing is now registered with the ID `dynamic-thing`. It's authenticated to AM and has received an access token.

Log in to the AM admin UI and select **Identities** in the Top Level Realm to locate the `dynamic-thing` in the list.

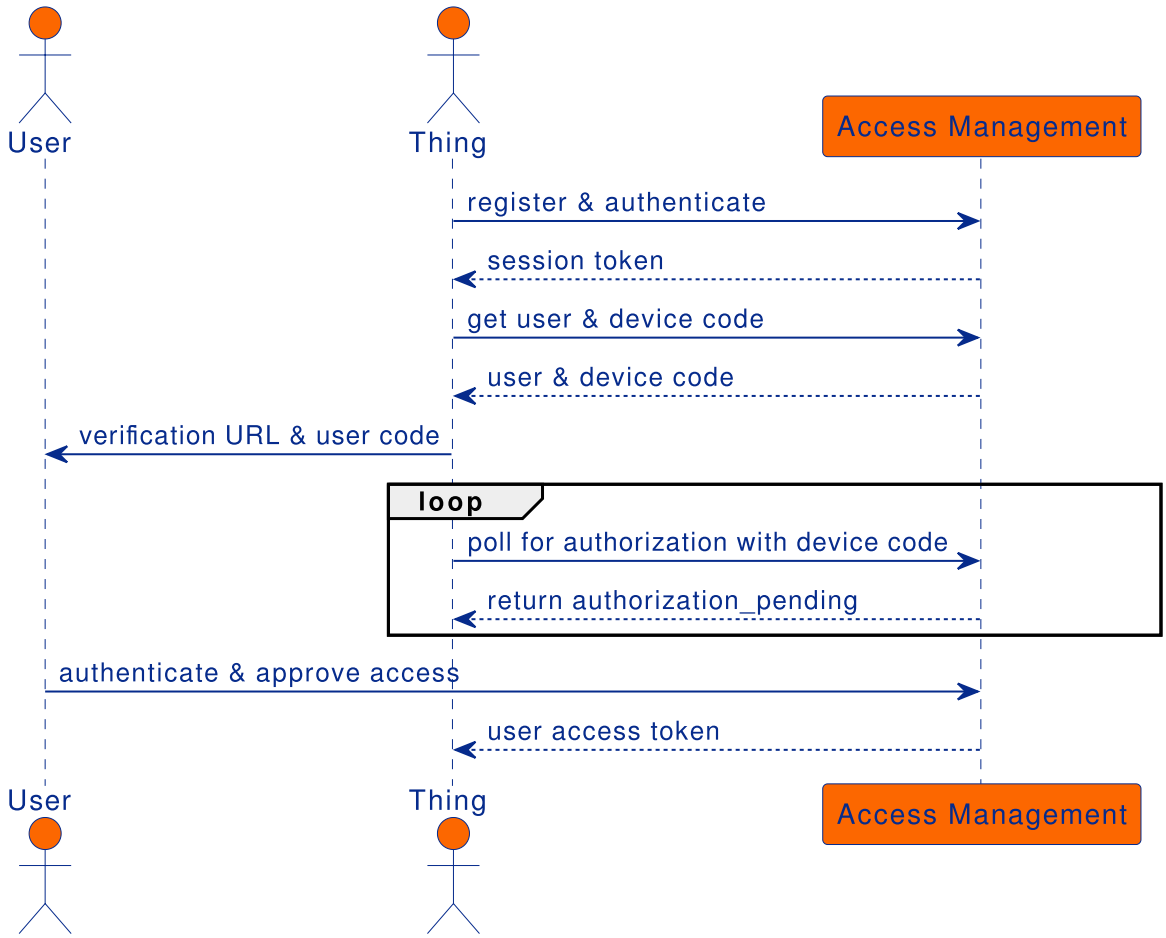
Request a user token for an authenticated thing

This example creates a new identity for a thing, using dynamic registration, and then authenticates it. When the thing is authenticated, it requests a user access token using the OAuth 2.0 Device Authorization Grant. The access token authorizes the thing to access a user's resources, or act on behalf of the user, as specified by the scope granted by the user.

The example demonstrates how the thing can manage the access token's lifecycle by introspecting and refreshing the token.

To request a user token, a user must be registered and authenticated before approving the request. When you run the example, the user is directed to a URL to perform the authorization. This example assumes that user bjensen is authenticated and accepts the request.

This sequence diagram shows how the thing is authorized for the session:



In the `auth-reg-tree`, change the configuration as follows:

- Authenticate Thing node
  - **JWT Authentication Method**: `Proof of Possession`
  - **Issue Restricted Token** enabled
- Register Thing node
  - **JWT Registration Method**: `Proof of Possession & Certificate`
  - **Create Identity** enabled

---

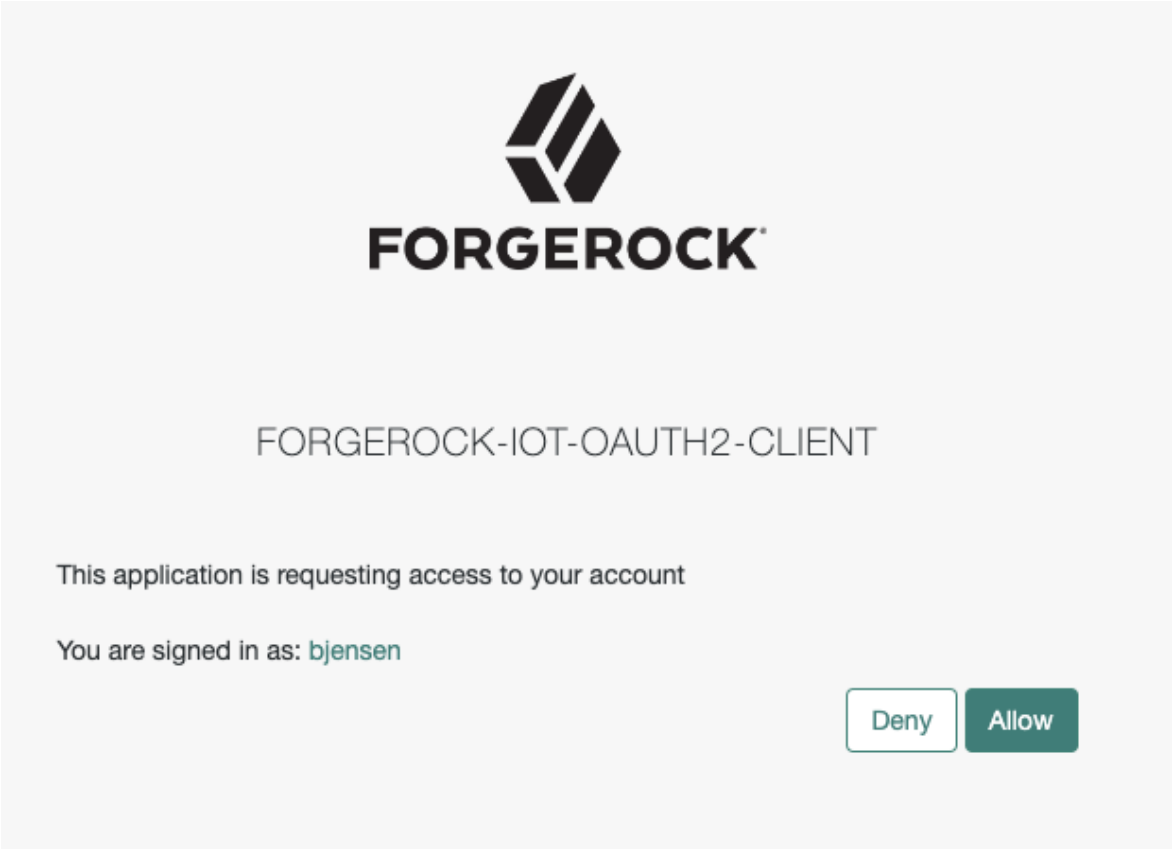
1. Run the `thing/user-token` example:

```
cd /path/to/iot-edge
./run.sh example "thing/user-token" \
-name "user-authorized-thing" \
-url "http://am.localtest.me:8080/openam" \
-tree "auth-reg-tree"

Creating Thing user-authorized-thing… Done

Requesting user code… Done
User code response: {
 "device_code":"code",
 "user_code":"code",

"verification_uri":"http://am.localtest.me:8080/openam/oauth2/
device/user",

"verification_uri_complete":"http://am.localtest.me:8080/opena
m/oauth2/device/user?user_code=code",
 "expires_in":300,
 "interval":5
}
Creating Thing user-authorized-thing… Done

Requesting user code… Done
 User code response: {
   "device_code":"eyJ0eXAiOiJKV1QiLC…p4GQHM",
   "user_code":"Y6GPhHpt",

"verification_uri":"http://am.localtest.me:8080/openam/oauth2/
device/user",

"verification_uri_complete":"http://am.localtest.me:8080/opena
```

```
m/oauth2/device/user?user_code=Y6GPhHpt",
   "expires_in":300,
   "interval":5}
Requesting user access token… To authorise the request, go to

        http://am.localtest.me:8080/openam/oauth2/device/user?
user_code=Y6GPhHpt
```

2. Go to `http://am.localtest.me:8080/openam/oauth2/device/user?user_code=Y6GPhHpt` and click **Confirm**.

   You are redirected to a screen that lets you confirm the authentication request:



3. Click **Allow**.

   The thing is now authenticated to AM and has received an access token. The output shows the complete flow.

```
Done
Access token response: {
"access_token": "I46b6E7k8xHyQPRcJ_Izcas1GVc",
"expires_in": 3599,
"refresh_token": "W9Yjr_7SEsnccw3x50f7o8PbWLs",
"scope": "subscribe publish",
"token_type": "Bearer"
}
```

```
Introspecting access token to get more information… Done
Introspection response: {
"active": true,
"auditTrackingId": "403dafaf-e367-4ef5-a485-e20fa44fbf0c-
998197",
"authGrantId": "tQKArSNFxEJKX_PLFVcHGTKqby8",
"auth_level": 0,
"client_id": "forgerock-iot-oauth2-client",
"exp": 1656586652,
"iss": "http://am.localtest.me:8080/openam/oauth2",
"realm": "/",
"scope": "subscribe publish",
"sub": "(usr!bjensen)",
"subname": "bjensen",
"token_type": "Bearer",
"user_id": "bjensen",
"username": "bjensen"
}

Refreshing access token with reduced scope… Done
Access token response: {
"access_token": "fI4KL7Ui2CjkvtULGMDfw0CDXSk",
"expires_in": 3599,
"refresh_token": "FJ5mfvvB8eMtj7MOj2t4MSKOUsQ",
"scope": "publish",
"token_type": "Bearer"
}

Introspecting access token to get more information… Done
Introspection response: {
"active": true,
"auditTrackingId": "403dafaf-e367-4ef5-a485-e20fa44fbf0c-
998250",
"authGrantId": "tQKArSNFxEJKX_PLFVcHGTKqby8",
"auth_level": 0,
"client_id": "forgerock-iot-oauth2-client",
"exp": 1656586652,
"iss": "http://am.localtest.me:8080/openam/oauth2",
"realm": "/",
"scope": "publish",
"sub": "(usr!bjensen)",
"subname": "bjensen",
"token_type": "Bearer",
"user_id": "bjensen",
```

```
"username": "bjensen"
}
```

IoT Gateway examples

The gateway examples demonstrate how to:

- Authenticate the gateway after manual registration
- Authenticate the gateway with dynamic registration
- Connect a thing to the gateway

These examples assume that you have downloaded the example repository and that the `iot-edge` directory is your current directory.

Authenticate the gateway after manual registration

This example starts the gateway, and authenticates it. The gateway must have an asymmetric key pair for signing. This is provided in the `/path/to/iot-edge/examples/resources` directory. The source code for this example is in `/path/to/iot-edge/cmd/gateway/main.go`.

Before you run the example:

- Register the gateway manually (using `manual-gateway` as the ID)
- In the `auth-tree`, make sure that the Authenticate Thing node has the following configuration:
  - **JWT Authentication Method**: `Proof of Possession`
  - **Issue Restricted Token** enabled

---

1. Start the gateway:

```
cd /path/to/iot-edge
./run.sh gateway \
--name "manual-gateway" \
--url "http://am.localtest.me:8080/openam" \
--audience "/" \
--realm "/" \
--tree "auth-tree" \
--kid "cbnztC8J_l2feNf0aTFBDDQJuvrd2JbLPoOAxHR2N8o=" \
--key "$(pwd)/examples/resources/eckey1.key.pem" \
--address ":5683" \
```

```
--debug
commandline options
  url: http://am.localtest.me:8080/openam
  realm: /
  tree: auth-tree
  name: manual-gateway
  address: :5683
  key: /path/to/iot-edge/examples/resources/eckey1.key.pem
  kid: cbnztC8J_l2feNf0aTFBDDQJuvrd2JbLPoOAxHR2N8o=
  certificate:
  timeout 5s
  debug: true
IoT Gateway server started.
```

The gateway is now started and has authenticated itself to AM.

2. In a separate terminal window, connect a thing to the gateway.
3. To stop the gateway process, press Ctrl+C in the terminal window where the process is running.

Authenticate the gateway with dynamic registration

This example registers an identity for the gateway, then starts the gateway, and authenticates it. The gateway must have an asymmetric key pair for signing, and a CA-signed X.509 certificate that contains the key pair's public key. This is provided in the `/path/to/iot-edge/examples/resources` directory. The source code for this example is in `/path/to/iot-edge/cmd/gateway/main.go`:

1. Start the gateway:

```
cd /path/to/iot-edge
./run.sh gateway \
--name "dynamic-gateway" \
--url "http://am.localtest.me:8080/openam" \
--audience "/" \
--realm "/" \
--tree "auth-reg-tree" \
--key "$(pwd)/examples/resources/eckey1.key.pem" \
--cert "$(pwd)/examples/resources/dynamic-gateway.cert.pem" \
--address ":5683" \
--debug
commandline options
        url: http://am.localtest.me:8080/openam
        realm: /
        tree: reg-tree
```

```
        name: dynamic-gateway
        address: :5683
        key: /path/to/iot-
edge/examples/resources/eckey1.key.pem
        kid:
        certificate: /path/to/iot-
edge/examples/resources/dynamic-gateway.cert.pem
        timeout 5s
        debug: true
IoT Gateway server started.
```

The gateway is now registered, with the ID `dynamic-gateway`, and has started and authenticated itself to AM.

2. In a separate terminal window, connect a thing to the gateway.
3. To stop the gateway process, press Ctrl+C in the terminal window where the process is running.

Connect a thing to the gateway

This example connects a thing to the gateway. When the thing has connected, it can authenticate to AM and request an access token. The source code for this example is in `/path/to/iot-edge/examples/thing/manual-registration/main.go`.

Before you run the example, register the thing manually (using `manual-thing` as the thing's ID). Then, run the `thing/manual-registration` example to connect the thing to the gateway:

```
cd /path/to/iot-edge
./run.sh example "thing/manual-registration" \
-name "manual-thing" \
-url "coap://:5683"
Creating Thing manual-thing… Done
Requesting access token… RequestAccessToken response:  {
    "access_token":"vHJDYCBkOjih90PWGAw0KcsCzpU",
    "scope":"publish",
    "token_type":"Bearer",
    "expires_in":3599
}
Done
Access token: vHJDYCBkOjih90PWGAw0KcsCzpU
Expires in: 3599
Scope(s): [publish]
```