



# Evaluation Guide

/ ForgeRock® IoT 7.0

Latest update: 7.0.0

ForgeRock AS.  
201 Mission St., Suite 2900  
San Francisco, CA 94105, USA  
+1 415-599-1100 (US)  
[www.forgerock.com](http://www.forgerock.com)

---

Copyright © 2020-2021 ForgeRock AS.

## Abstract

### Guide to getting started with ForgeRock IoT.



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

© Copyright 2010-2020 ForgeRock, Inc. All rights reserved. ForgeRock is a registered trademark of ForgeRock, Inc. Other marks appearing herein may be trademarks of their respective owners.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, and distribution. No part of this product or document may be reproduced in any form by any means without prior written authorization of ForgeRock and its licensors, if any.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESSED OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

#### DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: [fonts@gnome.org](mailto:fonts@gnome.org).

#### Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: [tavmjong@free.fr](mailto:tavmjong@free.fr).

#### FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

---






# Table of Contents

Overview .....	iv
1. About IoT .....	1
About ForgeRock IoT .....	1
2. Prerequisites .....	3
Install the Required Software .....	3
Get the Examples .....	3
Install and Configure AM .....	3
3. Register Identities .....	9
4. IoT SDK Examples .....	13
Authenticate a Thing After Manual Registration .....	13
Authenticate a Thing With Dynamic Registration .....	15
5. IoT Gateway Examples .....	18
Authenticate the Gateway After Manual Registration .....	18
Authenticate the Gateway With Dynamic Registration .....	19
Connect a Thing to the Gateway .....	20

# Overview

This guide covers the tasks you need to quickly get a test or demo environment running. It shows you how to configure ForgeRock® Access Management and run the IoT SDK and IoT Gateway examples.

## Quick Start

 <b>About IoT</b> Learn how ForgeRock IoT can help you register, authenticate, and authorize your IoT ecosystem.	 <b>Before you start</b> Install the prerequisite software and get the examples.	 <b>Register identities</b> Register an identity in AM manually, for a Thing or the IoT Gateway.
 <b>SDK Examples</b> Use the SDK examples to authenticate and request an access token for a Thing.	 <b>Gateway Examples</b> 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>.

## Chapter 1

# 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

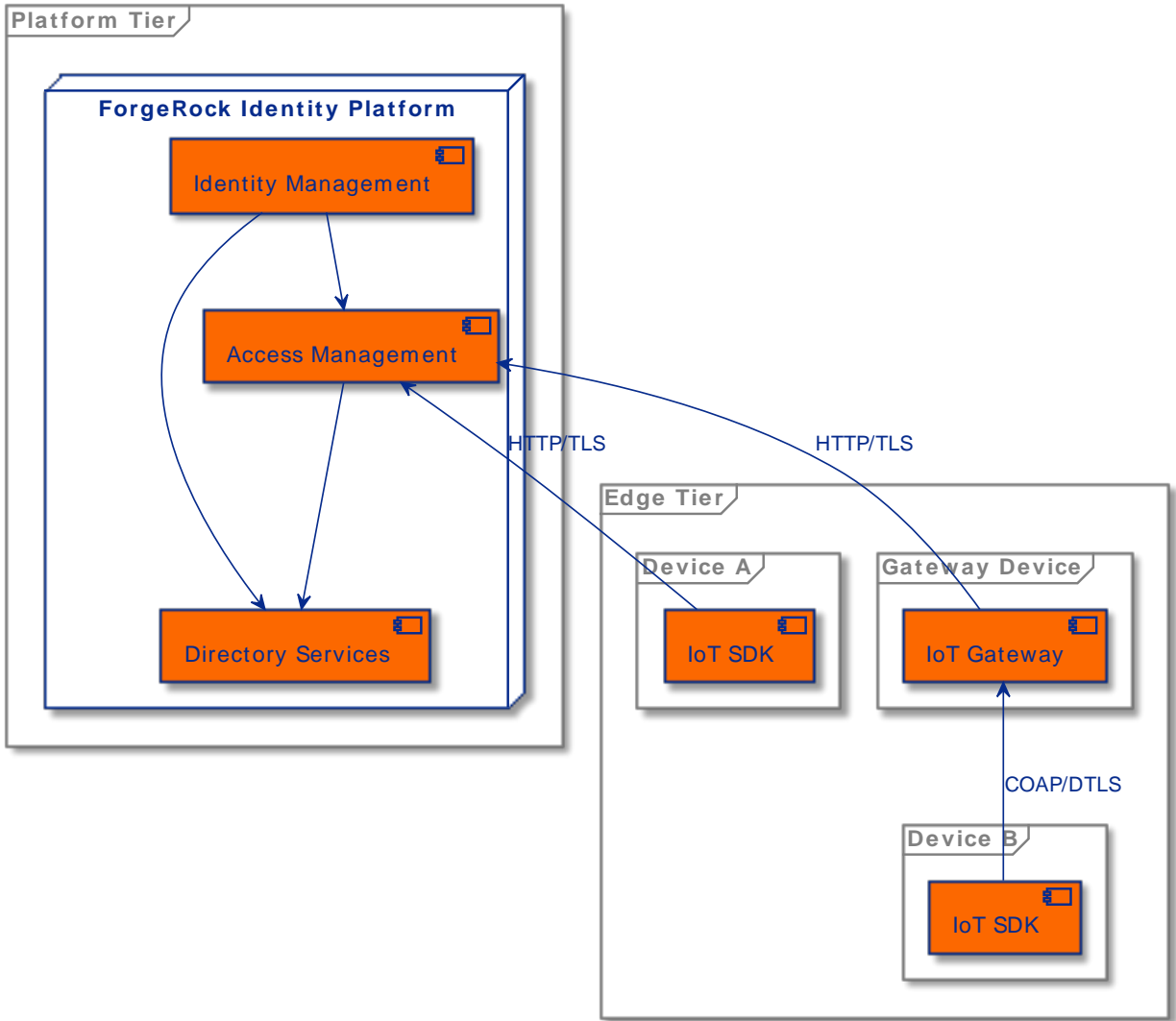
The IoT SDK lets a Thing (either a physical device or a software service) register and authenticate without human interaction. When the thing is registered, it is represented by a digital identity in the ForgeRock Identity Platform. It can then authenticate itself to interact with the platform tier.

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:



## Chapter 2

# Prerequisites

- "Install the Required Software"
- "Get the Examples"
- "Install and Configure AM"

## Install the Required Software

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

- Go, version 1.13 or later.
- Git (to download the source code and run the examples).

## Get the Examples

To download the examples, clone the `iot-edge` Git repository:

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

This command creates a directory named `iot-edge`. Change to that directory:

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

The examples assume that this is your current working directory.

## Install and Configure AM

1. Read the [ForgeRock® Access Management \(AM\) Evaluation Guide](#) 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, navigate to <http://am.localtest.me:8080/openam/>.

- AM is configured with the Default Configuration, with user `amAdmin` and password `changeit`.
2. Log in to AM as user `amAdmin` with password `changeit`.
  3. 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, see *IoT Service* in the *AM Reference*:

- a. In the Top Level Realm, select Services.
- b. Click Add a Service, select IoT Service, and click Create.
- c. 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.

- d. 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.iam.jwt.issuer.signing`. It must use the HS256 algorithm.

- e. Click Save Changes.



4. Add an OAuth2 Provider service:
  - a. Select Services.
  - b. Click Add a Service, select 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-Based Access & Refresh Tokens.
- On the Advanced tab, select an asymmetric key for the OAuth2 Token Signing Algorithm.

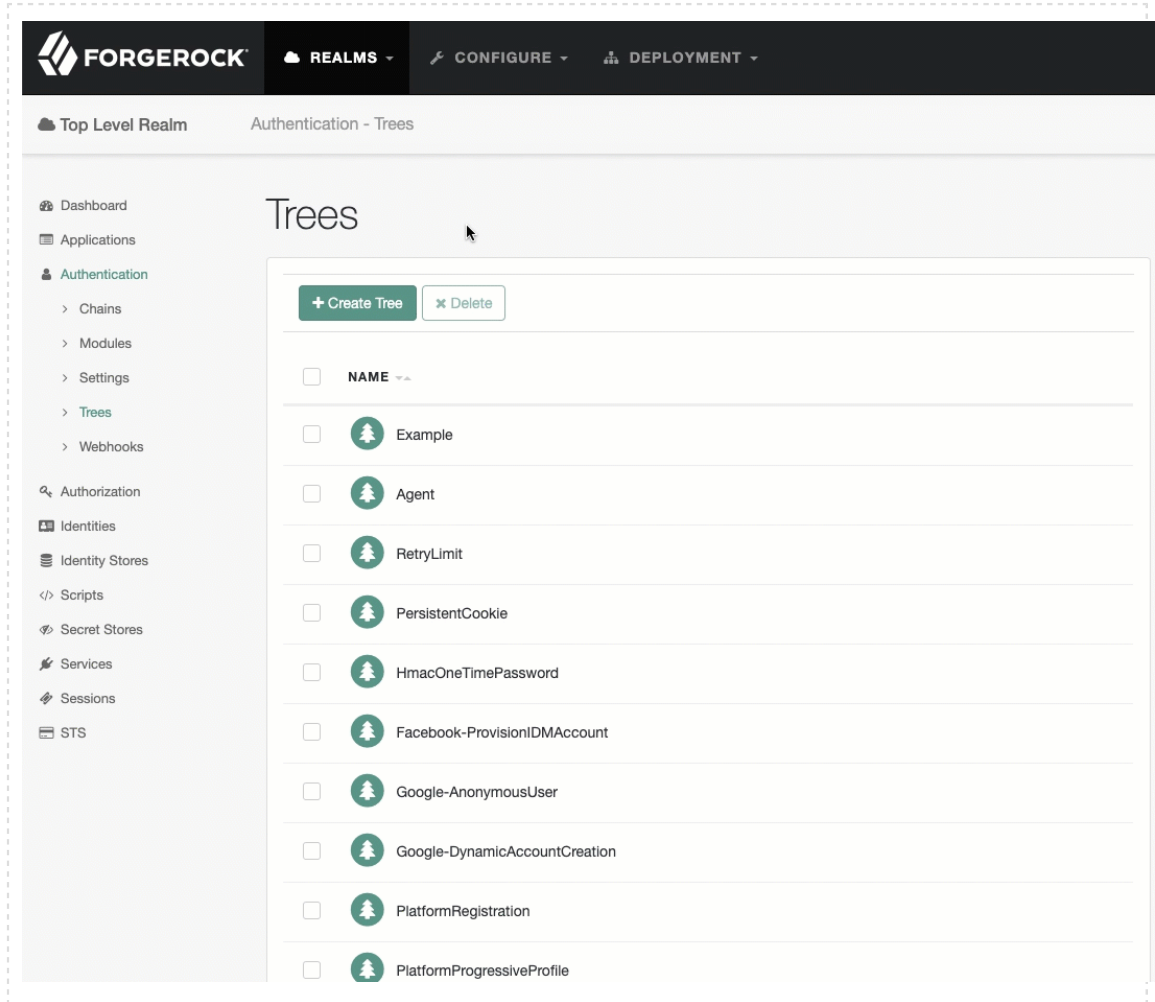
5. Configure the IoT OAuth 2.0 client:
  - a. Select Applications > OAuth 2.0 > Clients.
  - b. Click on forgerock-iot-oauth2-client.
  - c. In the Scope(s) field, type **publish**.
  - d. Click Save Changes.

+ *Advanced Use*

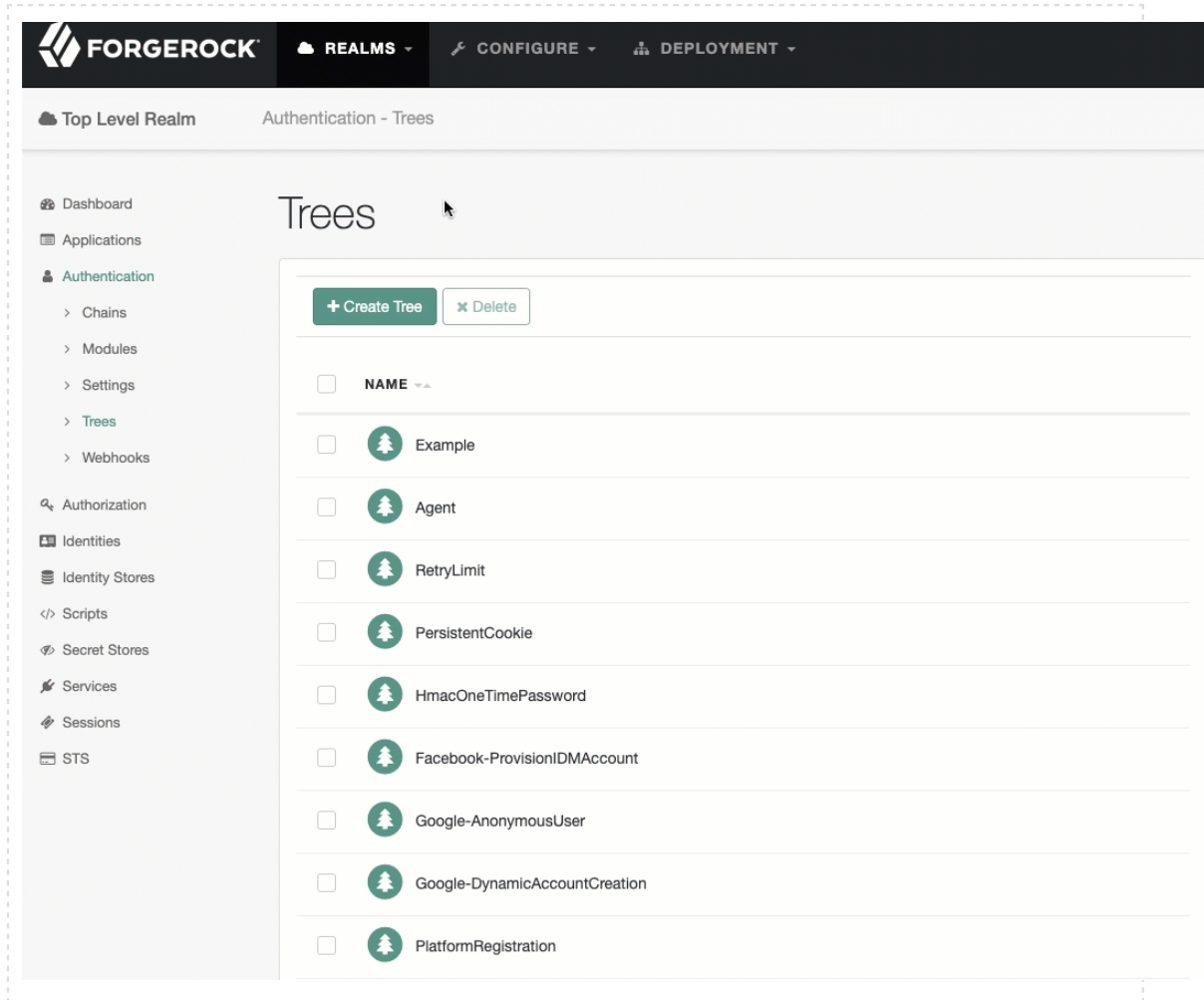
If you create your own OAuth2 client here, make sure that the client contains the **JWT Bearer** grant type and has a strong generated password.

6. Create two authentication trees:
  - a.
    - Select Authentication > Trees > Create Tree.
    - Type **auth-tree** in the Name field, and click Create.
    - Add an **Authenticate Thing** node and click Save.

+ *Show Me*



- b.
- Select Authentication > Trees > Create Tree.
  - Type `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, then click Save.
- + *Show Me*



7. Add a secret ID mapping.

- a. Select Configure > Secret Stores and click on the `default-keystore`.
- b. On the Mappings tab, click + Add Mapping.
- c. In the Secret ID list, select `am.services.iot.cert.verification`, and in the Alias field, type `es256test` then 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.

d. Click Create to add the mapping.

For more information about mapping secret IDs, see *Mapping and Rotating Secrets in the AM Security Guide*.

## Chapter 3

# 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, obtain 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 few sample fields (`thingKeys`) for the Thing or gateway you are registering:

## + Register a Thing

```

curl \
--header 'Content-Type: application/json' \
--header 'Accept-Api-Version: resource=4.0, protocol=2.1' \
--cookie 'iPlanetDirectoryPro=${tokenId}' \
--data '{
  "userPassword": "5tr0ngG3n3r@ted",
  "thingType": "device",
  "thingKeys": "{\"keys\": [{\"use\": \"sig\", \"kty\": \"EC\", \"kid\": \"pop.cnf\", \"crv\": \"P-256\", \"alg\": \"ES256\", \"x\": \"wjC9kMzwIeXNn6lsjdqplcq9aCwpA0Z0af1_yruCcJ4\", \"y\": \"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq04lBmfEHw\"}]}"
}' \
--request PUT \
"http://am.localtest.me:8080/openam/json/realms/root/users/${ID}"
{
  "_id": "manual-thing",
  "_rev": "-1",
  "realm": "/",
  "username": "manual-thing",
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "fr-iot",
    "sunFMSAML2NameIdentifier",
    "inetorgperson",
    "devicePrintProfilesContainer",
    "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-thing,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "manual-thing"
  ],
  "thingKeys": [
    "{\"keys\": [{\"use\": \"sig\", \"kty\": \"EC\", \"kid\": \"pop.cnf\", \"crv\": \"P-256\", \"alg\": \"ES256\", \"x\": \"wjC9kMzwIeXNn6lsjdqplcq9aCwpA0Z0af1_yruCcJ4\", \"y\": \"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq04lBmfEHw\"}]}"
  ],
  "createTimestamp": [
    "20200825154443Z"
  ],
  "uid": [
    "manual-thing"
  ]
}
    
```

```

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

```

If you sign in to the AM Admin UI and select Identities in the Top Level Realm, you will see the **manual-thing** in the list.

### + Register a Gateway

```

curl \
--header 'Content-Type: application/json' \
--header 'Accept-Api-Version: resource=4.0, protocol=2.1' \
--cookie "iPlanetDirectoryPro=${tokenId}" \
--data '{
  "userPassword": "5tr0ngG3n3r@ted",
  "thingType": "gateway",
  "thingKeys": "{\"keys\": [{\"use\": \"sig\", \"kty\": \"EC\", \"kid\": \"pop.cnf\", \"crv\": \"P-256\", \"alg\": \"ES256\", \"x\": \"wjC9kMzwIeXNn6lsjdqplcq9aCwpA0Z0af1_yruCcJ4\", \"y\": \"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq04lBmfEHw\"}]}"
}' \
--request PUT \
"http://am.localtest.me:8080/openam/json/realms/root/users/${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\": \"pop.cnf\", \"crv\": \"P-256\",
  \"alg\": \"ES256\", \"x\": \"wjC9kMzwIeXNn6lsjdqplcq9aCWpA0Z0af1_yruCcJ4\", \"y\":
  \"ihIziCymBnU8W8m5zx69DsQr0sWDiXsDMq04lBmfEHw\"}]}"
],
"createTimestamp": [
  "20200826104156Z"
],
"uid": [
  "manual-gateway"
],
"universalid": [
  "id=manual-gateway,ou=user,dc=openam,dc=forgerock,dc=org"
],
"inetUserStatus": [
  "Active"
],
"sn": [
  "manual-gateway"
],
"thingType": [
  "gateway"
]
}
```

If you sign in to the AM Admin UI and select Identities in the Top Level Realm, you will see the **manual-gateway** in the list.



## Chapter 4

# IoT SDK Examples

The IoT SDK examples demonstrate how to:

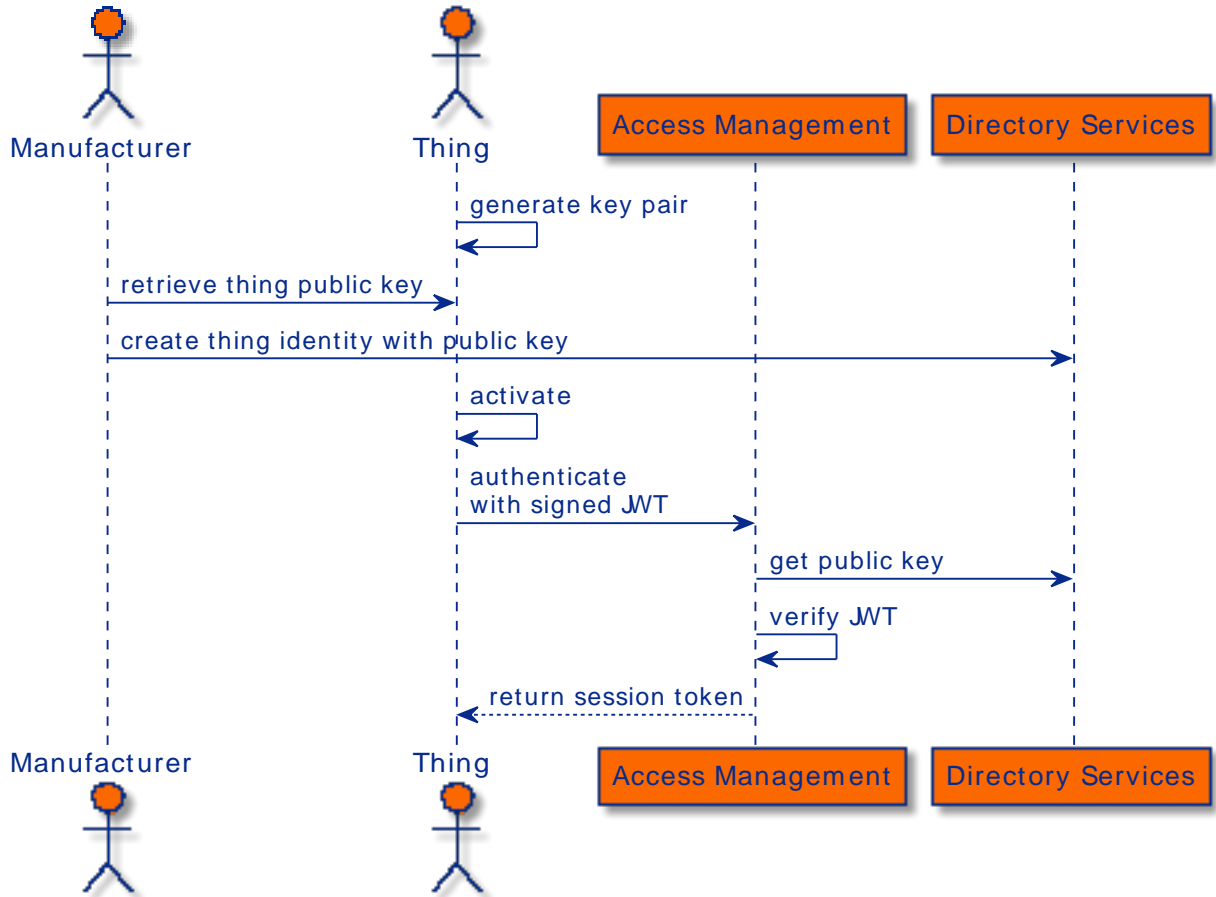
- "Authenticate a Thing After Manual Registration"
- "Authenticate a Thing With Dynamic Registration"

This section assumes 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/simple` example:

```
cd /path/to/iot-edge
./run.sh example "thing/simple" \
-name "manual-thing" \
-url "http://am.localtest.me:8080/openam" \
-audience "/" \
-realm "/" \
-tree "auth-tree" \
-keyfile "./examples/resources/eckey1.key.pem"
Creating Thing manual-thing... Done
Requesting access token... RequestAccessToken response: {
  "access_token":"iaZqWRyVBhGMLWwA0zDr0tKarf",
  "scope":"publish",
  "token_type":"Bearer",
  "expires_in":3599}
Done
Access token: iaZqWRyVBhGMLWwA0zDr0tKarf
Expires in: 3599
Scope(s): [publish]

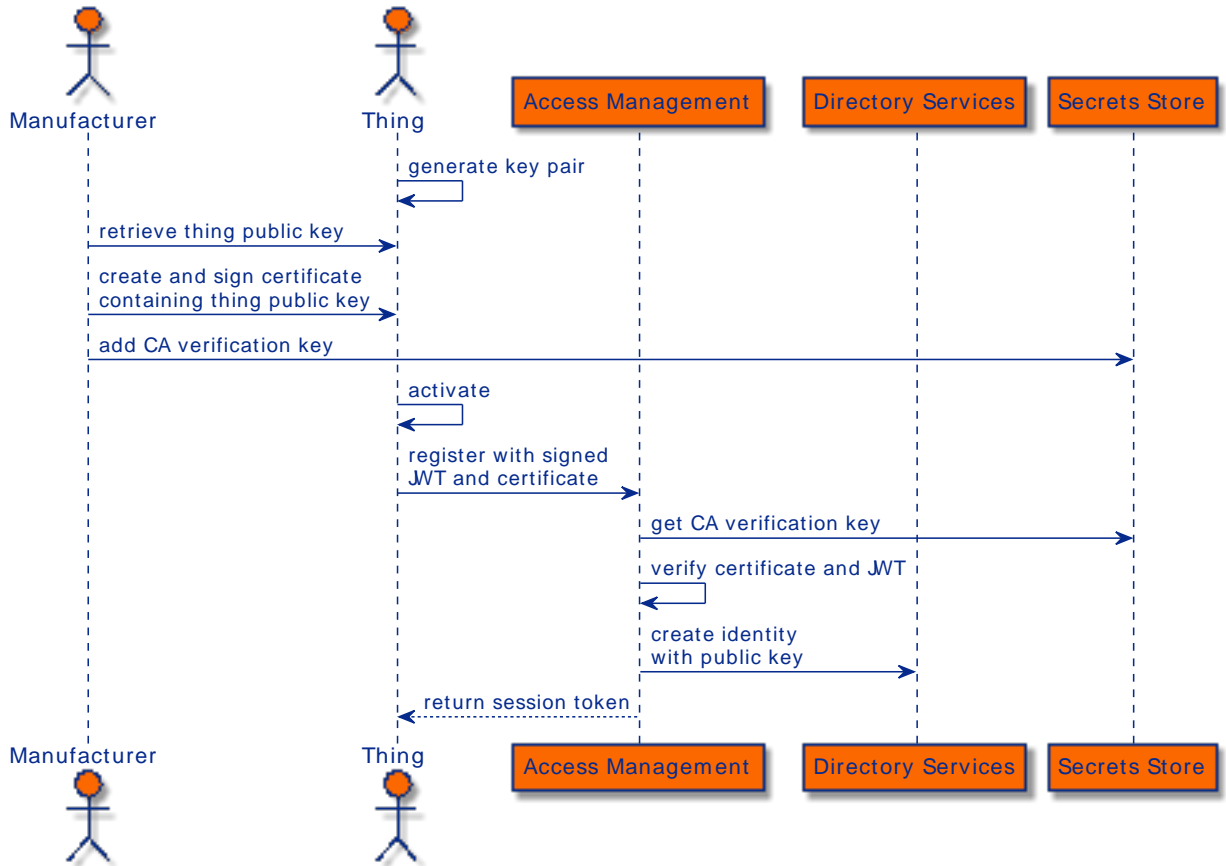
  / \  / \  / \  / \
 / \ / \ / \ / \ / \
/ \ / \ / \ / \ / \
/ \ / \ / \ / \ / \
/ \ / \ / \ / \ / \
```

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

## Authenticate a Thing With Dynamic Registration

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/cert-registration/main.go`.

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



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

```

cd /path/to/iot-edge
./run.sh example "thing/cert-registration" \
-name "dynamic-thing" \
-url "http://am.localtest.me:8080/openam" \
-audience "/" \
-realm "/" \
-tree "reg-tree" \
-keyfile "./examples/resources/eckey1.key.pem" \
-certfile "./examples/resources/dynamic-thing.cert.pem"
Creating Thing dynamic-thing... Done
Requesting access token... RequestAccessToken response: {
  "access_token":"84T-llAwUImk9NTP60bKKWZouW8",
  "scope":"publish",
  "token_type":"Bearer",
  "expires_in":3599
}
Done
Access token: 84T-llAwUImk9NTP60bKKWZouW8
Expires in: 3599
Scope(s): [publish]

  / \  \  / \  / \
 / \  \  / \  / \
 / \  \  / \  / \
 / \  \  / \  / \

```

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

If you sign in to the AM Admin UI and select Identities in the Top Level Realm, you will see the `dynamic-thing` in the list.

## Chapter 5

# 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"

This section assumes 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):

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 "pop.cnf" \
--key "./examples/resources/ekey1.key.pem" \
--address ":5683" \
--debug
commandline options
url: http://am.localtest.me:8080/openam
realm: /
tree: auth-tree
name: manual-gateway
address: :5683
key: ./examples/resources/ekey1.key.pem
kid: pop.cnf
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 "reg-tree" \
--key "./examples/resources/eckey1.key.pem" \
--cert "./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: ./examples/resources/eckey1.key.pem
kid:
certificate: ./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/simple/main.go>.

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

```
cd /path/to/iot-edge
./run.sh example "thing/simple" \
-name "gateway-thing" \
-url "coap://:5683" \
-keyfile "./examples/resources/eckey1.key.pem" \
-audience "/"
Creating Thing gateway-thing... Done
Requesting access token... RequestAccessToken response: {
  "access_token": "vHJDYCBk0jih90PWGAw0KcsCzpU",
  "scope": "publish",
  "token_type": "Bearer",
  "expires_in": 3599
}
Done
Access token: vHJDYCBk0jih90PWGAw0KcsCzpU
Expires in: 3599
Scope(s): [publish]
```

