



Developers 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

Copyright © 2020 ForgeRock AS.

Abstract

Guide to building applications with the IoT SDK.



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.

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.

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. Develop a Client Application With the IoT SDK	1
2. Build the ForgeRock IoT Gateway	5

Overview

This guide shows you how to use the IoT SDK to develop client applications and to register them with AM. It also shows you how to build the IoT Gateway.

Quick Start

 <p>Develop a client</p> <p>Develop a client application with the IoT SDK.</p>	 <p>Build the Gateway</p> <p>Build the IoT Gateway for your specific target system.</p>
---	---

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

Develop a Client Application With the IoT SDK

This section shows you how to create a client application for a Thing, named Gopher. The Thing is manually registered in AM and authenticated with a username/password authentication flow. For more information about the IoT SDK API, see the [Go package documentation](#).

- "Develop a ForgeRock IoT Application"
- "Run the Client Application"

Develop a ForgeRock IoT Application

These steps assume that you have installed the required software and cloned the Things GitHub repository:

1. Create a directory structure for your Go project:

```
mkdir -p things/cmd/gopher
```

2. Create an empty project file (`main.go`):

```
cd things  
touch cmd/gopher/main.go
```

3. Open `main.go` in a text editor, and add the following code:

```

package main

import (
    "github.com/ForgeRock/iot-edge/v7/pkg/builder"
    "github.com/ForgeRock/iot-edge/v7/pkg/callback"
    "log"
    "net/url"
)

func main() {
    amURL, err := url.Parse("http://am.localtest.me:8080/openam")
    if err != nil {
        log.Fatal(err)
    }
    _, err = builder.Thing().
        ConnectTo(amURL).
        InRealm("/").
        WithTree("Example").
        HandleCallbacksWith(
            callback.NameHandler{Name: "Gopher"},
            callback.PasswordHandler{Password: "5tr0ngG3n3r@ted"}).
        Create()
    if err != nil {
        log.Fatal(err)
    }
    log.Println("Gopher successfully authenticated.")
}
    
```

4. Create a Go module:

```

go mod init example.com/things
go: creating new go.mod: module example.com/things
    
```

This step creates a `go.mod` file that specifies your project dependencies and versions.

Run the Client Application

1. Before you can run the application, you need to register an identity for Gopher in AM:

a. Obtain an admin SSO token from AM:

```

curl \
  --header 'X-OpenAM-Username: amAdmin' \
  --header 'X-OpenAM-Password: changeit' \
  --header 'Content-Type: application/json' \
  --header 'Accept-API-Version: resource=2.0, protocol=1.0' \
  --request POST \
  'http://am.localtest.me:8080/openam/json/authenticate'
  {
    "tokenId": "qGAzvBw20z5...AAA.*",
    "successUrl": "/openam/console",
    "realm": "/"
  }
    
```

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

```
export tokenId=qGAzvBw20z5...AAA.*
echo $tokenId
qGAzvBw20z5...AAA.*
```

- c. Register the Gopher application, with the ID **Gopher**:

```
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"
}' \
--request PUT \
"http://am.localtest.me:8080/openam/json/realms/root/users/Gopher"
{
  "_id": "Gopher",
  "_rev": "-1",
  "realm": "/",
  "username": "Gopher",
  "uid": [
    "Gopher"
  ],
  "universalid": [
    "id=Gopher,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "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=Gopher,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "cn": [
    "Gopher"
  ],
  "sn": [
```

```
"Gopher"  
],  
"thingType": [  
  "device"  
],  
"createTimestamp": [  
  "20200831103235Z"  
]  
}
```

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

2. Build an executable for your client application:

```
go build example.com/things/cmd/gopher  
go: finding module for package github.com/ForgeRock/iot-edge/v7/pkg/callback  
go: finding module for package github.com/ForgeRock/iot-edge/v7/pkg/builder  
go: downloading github.com/ForgeRock/iot-edge/v7 v7.0.0  
go: downloading github.com/ForgeRock/iot-edge v0.0.0-20200812141306-ee64981fa05f  
go: found github.com/ForgeRock/iot-edge/v7/pkg/builder in github.com/ForgeRock/iot-edge/v7 v7.0.0  
go: found github.com/ForgeRock/iot-edge/v7/pkg/callback in github.com/ForgeRock/iot-edge/v7 v7.0.0
```

This step builds an executable **gopher** application in the **things** directory.

3. Run the executable to authenticate your application to AM:

```
./gopher  
2020/09/01 11:09:49 Gopher successfully authenticated.
```


Chapter 2

Build the ForgeRock IoT Gateway

ForgeRock does not deliver binaries for the IoT Gateway. There are simply too many operating system and architecture combinations to support. The IoT Gateway and the IoT SDK are developed in the Go programming language primarily because it has uncomplicated build tooling and good support for cross-compilation to target systems.

- "Build the IoT Gateway on a Target System"
- "Cross-Compile the IoT Gateway for a Target System"

Build the IoT Gateway on a Target System

These steps assume that you have installed the required software and cloned the Things GitHub repository:

1. On your target system, navigate to the `iot-edge` directory:

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

2. Build the IoT Gateway binary:

```
go build -o ./bin/gateway ./cmd/gateway
```

The IoT Gateway binary is now available at `bin/gateway`.

3. (Optional) Run the IoT Gateway with the `--help` flag for available command-line options:

```
./bin/gateway --help
Usage:
gateway [OPTIONS]

Application Options:
--url=      AM URL
--realm=    AM Realm
--audience= JWT Audience
--tree=     Authentication tree
--name=     Gateway name
--address=  CoAP Address of Gateway
--key=      The file containing the Gateway's signing key
--kid=      The Gateway's signing key ID
--cert=     The file containing the Gateway's certificate
--timeout=  Timeout for AM communications (default: 5s)
-d, --debug Switch on debug

Help Options:
```

```
-h, --help      Show this help message

2020/08/28 10:40:33 Usage:
gateway [OPTIONS]

Application Options:
--url=         AM URL
--realm=       AM Realm
--audience=   JWT Audience
--tree=        Authentication tree
--name=        Gateway name
--address=     CoAP Address of Gateway
--key=         The file containing the Gateway's signing key
--kid=         The Gateway's signing key ID
--cert=        The file containing the Gateway's certificate
--timeout=     Timeout for AM communications (default: 5s)
-d, --debug    Switch on debug

Help Options:
-h, --help      Show this help message
```

Cross-Compile the IoT Gateway for a Target System

You can specify a target system with a combination of the `$GOOS` and `$GOARCH` environment variables. This lets you build the IoT Gateway for a variety of operating system and architecture combinations.

For example, to run the IoT Gateway on an `arm` 32-bit processor (for example, a Raspberry Pi 3 running in 32-bit mode), build the IoT Gateway for `linux/arm`, as follows:

1.

```
GOOS=linux GOARCH=arm go build -o ./bin/gateway ./cmd/gateway
```
2. (Optional) For a complete list of environment and cross-compilation targets, see the [go Documentation](#).

For more build options, see the [go command environment variables](#)