



# Getting Started Guide

/ ForgeRock Identity Gateway 6.5

Latest update: 6.5.4

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 © 2017-2021 ForgeRock AS.

## Abstract

Quick introduction to ForgeRock® Identity Gateway for new users and readers evaluating the product.



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.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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 NONINFRINGEMENT 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 at gnome dot org](mailto:fonts at gnome dot 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 NONINFRINGEMENT 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, <http://fontawesome.io>.

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

Preface .....	iv
1. About This Guide .....	iv
2. Formatting Conventions .....	iv
3. Accessing Documentation Online .....	v
4. Using the ForgeRock.org Site .....	v
5. Getting Support and Contacting ForgeRock .....	v
1. IG At a Glance .....	1
2. First Steps .....	3
2.1. Software Requirements .....	3
2.2. Configuring the Network .....	3
2.3. Installing and Starting IG .....	4
2.4. Installing the Sample Application .....	6
2.5. Trying IG With a Simple Configuration .....	8
2.6. Adding a Route to the IG Configuration .....	10
3. Configuring Routes With Studio .....	13
3.1. Accessing Studio .....	13
3.2. Summary of Studio Tasks, Route Status, and Icons .....	14
3.3. Creating Simple Routes .....	16
3.4. Changing the Basic Settings of a Route .....	16
3.5. Deploying Routes to Your Configuration .....	17
3.6. Adding Filters to a Route .....	17
3.7. Editing Routes In Editor Mode .....	18
3.8. Importing Routes Into Studio .....	19
3.9. Viewing and Searching for Routes in Your Configuration .....	20
3.10. Restricting Access to Studio .....	20
A. Technology Preview of Freeform Studio .....	23

# Preface

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>.

## 1. About This Guide

IG integrates web applications, APIs, and microservices with the ForgeRock Identity Platform, without changing the application or container where they run. Based on reverse proxy architecture, it enforces security and access control in conjunction with AM modules.

This guide can help access management designers and administrators to install IG with a basic configuration, and start using the basic features of IG quickly and easily.

This guide assumes basic familiarity with the following topics:

- HTTP, including how clients and servers exchange messages, and the role that a reverse proxy (gateway) plays
- JSON, the format for IG configuration files
- Managing services on operating systems and application servers
- Configuring network connections on operating systems

## 2. Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server`, even if the text applies to `C:\path\to\server` as well.

Absolute path names usually begin with the placeholder `/path/to/`. This path might translate to `/opt/`, `C:\Program Files\`, or somewhere else on your system.

Command-line, terminal sessions are formatted as follows:

```
$ echo $JAVA_HOME
/path/to/jdk
```

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command.

Program listings are formatted as follows:

```
class Test {
    public static void main(String [] args) {
        System.out.println("This is a program listing.");
    }
}
```

## 3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

## 4. Using the ForgeRock.org Site

The [ForgeRock.org](https://www.forgerock.org) site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

## 5. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see <https://www.forgerock.com>.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit <https://www.forgerock.com/support>.

## Chapter 1

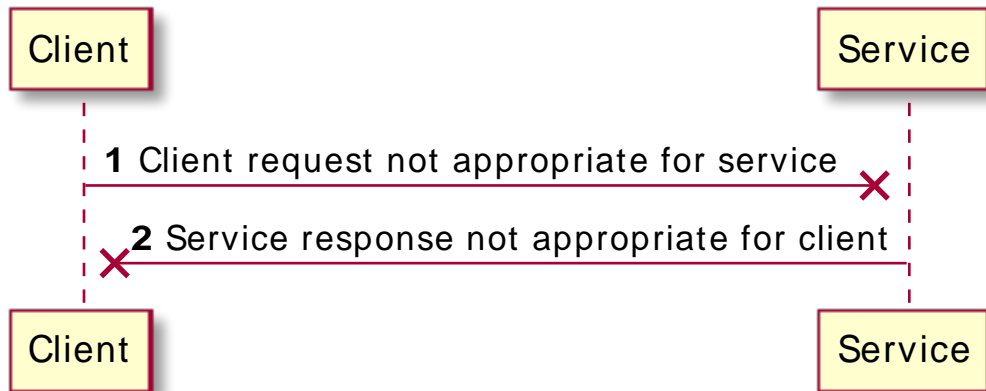
# IG At a Glance

This chapter provides a quick look at what is available in IG. For more information about the features of IG, see "About IG" in the *Gateway Guide*.

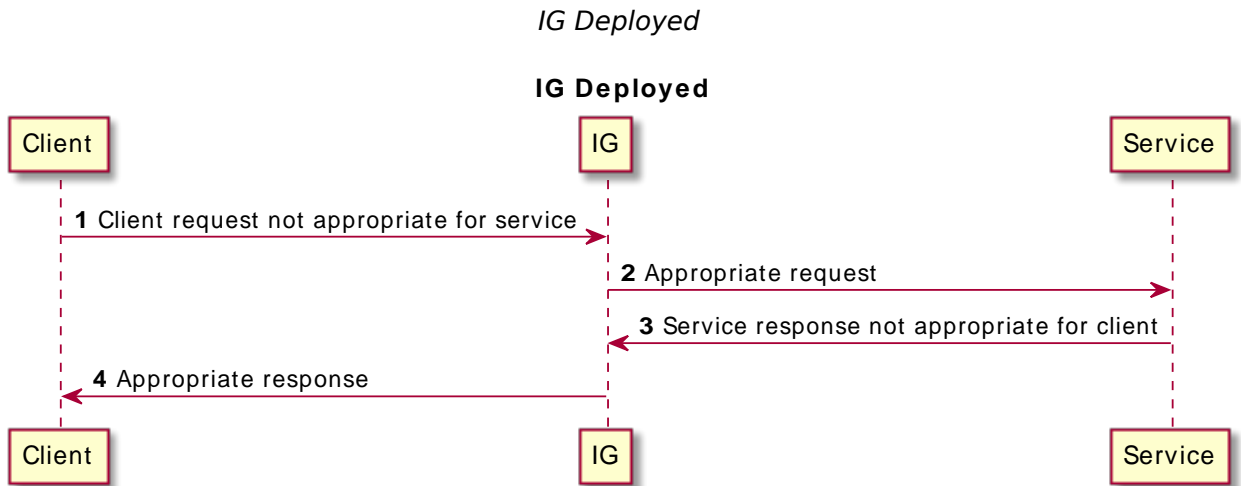
Most organizations have valuable existing services that are not easily integrated into newer architectures. These existing services cannot often be changed. Many client applications cannot communicate as they lack a gateway to bridge the gap. "Missing Gateway" illustrates one example of a missing gateway.

### Missing Gateway

## Missing Gateway



IG works as an HTTP gateway, based on reverse proxy architecture. IG is deployed on a network so it can intercept both client requests and server responses. "IG Deployed" illustrates a IG deployment.



Clients interact with protected servers through IG, which can be configured to add new capabilities to existing services without affecting current clients or servers.

The following list includes features that you can add to your solution by using IG:

- Access management integration
- Application and API security
- Credential replay
- OAuth 2.0 support
- OpenID Connect 1.0 support
- Network traffic control
- Proxy with request and response capture
- Request and response rewriting
- SAML 2.0 federation support
- SSO

## Chapter 2

# First Steps

This chapter describes how to quickly set up an instance of IG and use it as a gateway to access a sample application.

## 2.1. Software Requirements

- **Java Development Kit (JDK):** For information about supported versions of JDK, see "Java Requirements" in the *Release Notes*.
- **Apache Tomcat or Jetty:** For information about supported web application containers, see "Web Application Containers" in the *Release Notes*.

## 2.2. Configuring the Network

Because IG uses reverse proxy architecture, you must configure the network so that traffic from the browser to the sample application goes through IG.

In this guide, IG and a sample application run on the same host as your browser, which is probably your laptop or desktop. Because network configuration is an important deployment step, the configuration in this chapter expects the sample application to run on `app.example.com` rather than `localhost`.

The quickest way to configure the network locally is to add an entry to your `/etc/hosts` file on UNIX systems or `%SystemRoot%\system32\drivers\etc\hosts` on Windows. For more information about host files, see the Wikipedia entry, *Hosts (file)*.

Edit your `/etc/hosts` file as follows:

- To run the browser, IG, and the sample application on the same host, add the following entry:

```
127.0.0.1    localhost openig.example.com app.example.com
```

- To run the browser on one host, and IG and the sample application on another host, add the IP address of the host running IG and the sample application to the hosts file on the system running the browser, where the host name matches that of sample application. For example, if IG and the sample application are running on a host with IP address 192.168.0.15, add the following entry:

```
192.168.0.15    openig.example.com app.example.com
```



- To run the sample application on a different host to IG, also make sure that the host name of the sample application resolves correctly for requests from IG to the application.

### Tip

- Some browsers cache IP address resolutions, even after clearing all browsing data. After changing the IP addresses of named hosts, restart the browser.

To make sure DNS or host settings are configured correctly for remote systems, stop IG and then make sure you cannot reach the target application with the host name and port number of IG. If you can still reach it, double-check your host settings.

- Make sure that name resolution is configured to check host files before DNS. For most UNIX systems, make sure that `files` is listed before `dns` in `/etc/nsswitch.conf`.

## 2.3. Installing and Starting IG

This section describes how to install IG, switch from the default production mode to development mode, and start IG.

### 2.3.1. Installing IG

IG is installed in the root context of a supported web application container listed in "Web Application Containers" in the *Release Notes*.

This guide uses Jetty server as an example. The commands in this guide assume you install Jetty to `/path/to/jetty`, and after installation, you have a directory `/path/to/jetty/webapps` in which you install IG. If you use another directory structure, substitute the commands.

#### *To Install IG*

1. Download a supported version of Jetty server from its [download page](#), and install it to `/path/to/jetty`.
2. Download `IG-6.5.4.war` from the [ForgeRock BackStage download site](#) .
3. Copy the `.war` file:

```
$ cp IG-6.5.4.war /path/to/jetty/webapps/IG-6.5.4.war
```

Jetty automatically deploys IG in the root context on startup.

## 2.3.2. Switching Between Production Mode and Development Mode

IG operates in `production` (immutable mode) and `development` (mutable mode). For information about setting the mode, see the `mode` property of `AdminHttpApplication(5)` in the *Configuration Reference* and `Configuration Tokens(5)` in the *Configuration Reference*.

After installation, IG is by default in production mode. Switch to development mode in one of the following ways, applied in order of precedence:

1. Set the `mode` property of `admin.json` by adding the following file to the IG configuration as `$HOME/.openig/config/admin.json` (on Windows, `%appdata%\OpenIG\config`):

```
{
  "mode": "DEVELOPMENT"
}
```

2. Define an environment variable for the configuration token `ig.run.mode`:

```
$ export IG_RUN_MODE=development
```

If `mode` is not defined in `admin.json`, this token is resolved at startup and its value is set by the environment variable.

3. Define a system property for the configuration token `ig.run.mode` when you start IG:

```
$ java -jar start.jar -Dig.run.mode=development
```

If `mode` is not defined in `admin.json`, and its value is not set by an environment variable, its value is set by the system property.

If a system property is not set, the mode defaults to production.

For information about switching back to production mode, see "To Make the Configuration Immutable" in the *Gateway Guide*.

## 2.3.3. Starting IG

### To Start IG

1. Start Jetty:
  - To start Jetty in the background, enter:

```
$ /path/to/jetty/bin/jetty.sh start
```

- To start Jetty in the foreground, enter:

```
$ cd /path/to/jetty/
$ java -jar start.jar
```

## 2. Make sure that IG is running:

- Make sure you can see the IG welcome page at <http://localhost:8080>.
- Display the product version and build information at <http://localhost:8080/openig/api/info>.
- If you switched to development mode, as described in "Switching Between Production Mode and Development Mode", make sure that you can see Studio at <http://localhost:8080/openig/studio>.

For more information about Studio, see "[Configuring Routes With Studio](#)".

## 2.4. Installing the Sample Application

ForgeRock provides a mockup web application for testing IG configurations. The sample application is used in many of the tutorials in this guide and the *Gateway Guide*. This section describes how to download and run the sample application.

### *Installing the Sample Application*

1. Download the sample application, [IG-sample-application-6.5.4.jar](#), from the ForgeRock BackStage download site .
2. Run the sample application:

```
$ java -jar IG-sample-application-6.5.4.jar
Preparing to listen for HTTP on port 8081.
Preparing to listen for HTTPS on port 8444.
The server will use a self-signed certificate not known to browsers.
When using HTTPS with curl for example, try --insecure.
Using OpenAM URL: http://openam.example.com:8088/openam/oauth2.
Starting server...
Sep 09, 2016 9:52:56 AM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [0.0.0.0:8444]
Sep 09, 2016 9:52:56 AM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [0.0.0.0:8081]
Sep 09, 2016 9:52:56 AM org.glassfish.grizzly.http.server.HttpServer start
INFO: [HttpServer] Started.
Press Ctrl+C to stop the server.
```

By default, this server listens for HTTP on port 8081, and for HTTPS on port 8444. If one or both of those ports are not free, specify other ports:

```
$ java -jar IG-sample-application-6.5.4.jar 8888 8889
Preparing to listen for HTTP on port 8888.
Preparing to listen for HTTPS on port 8889.
The server will use a self-signed certificate not known to browsers.
When using HTTPS with curl for example, try --insecure.
Using OpenAM URL: http://openam.example.com:8088/openam/oauth2.
Starting server...
Sep 09, 2016 9:55:57 AM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [0.0.0.0:8889]
Sep 09, 2016 9:55:57 AM org.glassfish.grizzly.http.server.NetworkListener start
INFO: Started listener bound to [0.0.0.0:8888]
Sep 09, 2016 9:55:57 AM org.glassfish.grizzly.http.server.HttpServer start
INFO: [HttpServer] Started.
Press Ctrl+C to stop the server.
```

If you change the port numbers when starting the server, also account for the differences when using the examples.

3. Browse to <http://localhost:8081/home> to access the home page of the sample application.

Some information about the browser request is displayed.

4. Browse to <http://localhost:8081/login> to access the login page of the sample application, and then log in with username `demo` and password `Ch4ng31t`.

The username and some information about the browser request is displayed.

## Stopping and Restarting IG

1. Stop Jetty:

- If Jetty is running in the background, enter:

```
$ /path/to/jetty/bin/jetty.sh stop
```

- If Jetty is running in the foreground, enter Ctrl+c in the terminal where Jetty is running.

2. Start Jetty:

- To start Jetty in the background, enter:

```
$ /path/to/jetty/bin/jetty.sh start
```

- To start Jetty in the foreground, enter:

```
$ cd /path/to/jetty/
$ java -jar start.jar
```

## 2.5. Trying IG With a Simple Configuration

### 2.5.1. Adding a Base Configuration File

The entry point for requests coming in to IG is a JSON-encoded configuration file, expected by default at `$HOME/.openig/config/config.json` (on Windows, `%appdata%\OpenIG\config\config.json`).

The base configuration file initializes a heap of objects and provides the main handler to receive incoming requests. Configuration in the file is inherited by all applicable objects in the configuration.

At startup, if IG doesn't find a base configuration file it provides a default version, given in the examples section of `GatewayHttpApplication(5)` in the *Configuration Reference*. The default looks for routes in `$HOME/.openig/config/routes` (on Windows, `%appdata%\OpenIG\config\routes`).

Consider adding a custom `config.json` for these reasons:

- To prevent using the default `config.json`, whose configuration might not be appropriate in your deployment.
- To define an object once in `config.json`, and then use it multiple times in your configuration.

After adding or editing `config.json`, stop and restart IG to take the changes into effect.

For more information, see `GatewayHttpApplication(5)` in the *Configuration Reference*, `Heap Objects(5)` in the *Configuration Reference*, and `Router(5)` in the *Configuration Reference*.

#### *To Configure the Base Configuration of IG*

1. Add the following file to the IG configuration as `$HOME/.openig/config/config.json`, on Windows add the file as `%appdata%\OpenIG\config\config.json`:

```
{
  "handler": {
    "type": "Router",
    "name": "_router",
    "baseURI": "http://app.example.com:8081",
    "capture": "all"
  },
  "heap": [
    {
      "name": "JwtSession",
      "type": "JwtSession"
    },
    {
      "name": "capture",
      "type": "CaptureDecorator",
      "config": {
        "captureEntity": true,
        "_captureContext": true
      }
    }
  ]
}
```

```
}  
]  
}
```

Notice the following features of the file:

- The handler contains a main router named `_router`. When IG receives an incoming request, `_router` routes the request to the first route in the configuration whose condition is satisfied.
- The `baseURI` changes the request URI to point the request to the sample application.
- The `capture` captures the body of the HTTP request and response.
- The `JwtSession` object in the heap can be used in routes to store the session information as JSON Web Tokens (JWT) in a cookie. For more information, see `JwtSession(5)` in the *Configuration Reference*.

## 2. Stop and restart IG, as described in "Stopping and Restarting IG".

The Jetty log includes a message that the config is loaded from the new file:

```
INFO o.f.openig.web.Initializer -  
Reading the configuration from $HOME/.openig/config/config.json
```

To locate the `%appdata%` folder for your version of Windows, open Windows Explorer, enter `%appdata%` as the file path, and press Enter. You must create the `%appdata%\OpenIG\config` folder, and then copy the configuration files.

Before you use this base configuration in production, adjust the log level, and deactivate the `CaptureDecorator` that generates several log message lines for each request and response. Also consider editing the router based on recommendations described in "Preventing the Reload of Routes" in the *Gateway Guide*.

### Important

If you plan to create routes through Studio, make sure that `config.json` contains a main router named `_router`. For information about Studio, see "Configuring Routes With Studio".

## 2.5.2. Adding a Default Route

### To Configure a Default Route

- Add the following file to the IG configuration as `$HOME/.openig/config/routes/zz-default.json`, on Windows add the file as `%appdata%\OpenIG\config\routes\zz-default.json`:

```
{  
  "handler": "ReverseProxyHandler"
```

```
}
```

The Jetty log includes a message that the new file is loaded into the config:

```
INFO o.f.o.handler.router.RouterHandler -  
Loaded the route with id 'zz-default' registered with the name 'zz-default'
```

Notice the following features of the route:

- The route calls a `ReverseProxyHandler` with the default configuration. The `ReverseProxyHandler` simply proxies the request to the sample application and returns the response, without changing either the request or the response.
- Routes are ordered lexicographically in the IG configuration by route name. If a route is not named, then the route ID is used instead. Naming the default route as `zz-default` almost guarantees that it is the last route in the configuration, and therefore the last route to which requests are routed.

### 2.5.3. Testing the Setup

To test your configuration, make sure that IG and the sample application are running, and then browse to `http://openig.example.com:8080/home`. You should be directed to the home page of the sample application.

What just happened:

- When you browsed to `http://openig.example.com:8080/home`, you connected to IG deployed in Jetty.
- The `baseURI` in `config.json` changed the request URI to point the request to the sample application, and the capture captured the body of the HTTP request and response.

The main router in `config.json` routed the request to the IG configuration.

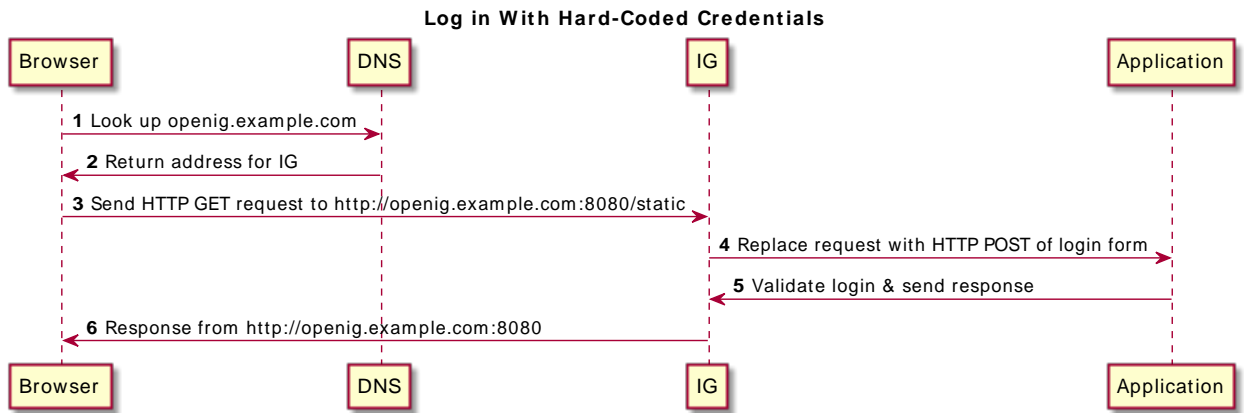
- Because there was no other route in the IG configuration, the request was routed to `zz-default.json`. This default route called a `ReverseProxyHandler`, which submitted the request to the sample application and returned the response without changing either the request or the response.
- The browser request was sent unchanged to the sample application, and the response from the sample application was returned unchanged to the browser.

## 2.6. Adding a Route to the IG Configuration

In the previous section you set up IG to proxy requests to the home page of the sample application. In this section, you add a route to log you in to the sample application automatically.

"Log in With Hard-Coded Credentials" shows the steps to log in to the sample application by using hard-coded credentials.

## Log in With Hard-Coded Credentials



1. The browser host makes a DNS request for the IP address of the HTTP server host, `openig.example.com`.
2. DNS responds with the address for IG.
3. Browser sends a request to the HTTP server.
4. IG replaces the browser's original HTTP GET request with an HTTP POST login request containing credentials to authenticate. As a result, instead of returning the login page with a login form, IG logs you in directly.
5. The sample application responds with the page you see after logging in.
6. IG returns this response to your browser.

### 2.6.1. Configuring IG to Log You In With Credentials

#### *To Configure IG to Log You In With Credentials*

- Add the following file to the IG configuration as `$HOME/.openig/config/routes/01-static.json`, on Windows add the file as `%appdata%\OpenIG\config\routes\01-static.json`:

```

{
  "handler": {
    "type": "Chain",
    "config": {
      "filters": [
        {
          "type": "StaticRequestFilter",

```



```
    "config": {
      "method": "POST",
      "uri": "http://app.example.com:8081/login",
      "form": {
        "username": [
          "demo"
        ],
        "password": [
          "Ch4ng31t"
        ]
      }
    },
    "handler": "ReverseProxyHandler"
  },
  "condition": "${matches(request.uri.path, '^/static')}"
}
```

The Jetty log includes a message that the new file is loaded into the config:

```
INFO o.f.o.handler.router.RouterHandler -
Loaded the route with id '01-static' registered with the name '01-static'
```

By default, routes in the `$HOME/.openig/config/routes` directory are loaded and updated without restarting IG.

Because routes are ordered in the IG configuration lexicographically by route name, a request is routed to `01-static.json` before `zz-default.json`.

## 2.6.2. Testing the Setup

To test your configuration, make sure that IG and the sample application are running, and then browse to `http://openig.example.com:8080/static`. You should be directed to the sample application and logged in automatically as `demo`.

What just happened:

- When you browsed to `http://openig.example.com:8080/static`, you connected to IG deployed in Jetty, and the request was routed to `01-static.json`.
- The `StaticRequestFilter` replaced the request with an HTTP POST request, including the login form with hard-coded credentials.
- The sample application validated the credentials, and responded with the profile page.
- The `ReverseProxyHandler` proxied the request to the sample application and returned the response.
- IG passed the response back to the browser.

## Chapter 3

# Configuring Routes With Studio

IG Studio is a user interface to configure and deploy routes in IG. It provides an easy way to evaluate or demo IG, or to create routes to authenticate users, authorize access to APIs, throttle requests to protected applications, capture messages, and collect statistics.

Freeform Studio is a new user interface available in Technology Preview to develop complex routes of filters and handlers. For more information, see "*Technology Preview of Freeform Studio*".

## 3.1. Accessing Studio

### Important

Before you access Studio, make sure that:

- IG is running development mode. After installation, IG is by default in production mode. For information about how to switch to development mode, see "Switching Between Production Mode and Development Mode".
- A custom `config.json` contains a main router named `_router`. IG deploys and undeploys routes through a main router named `_router`, which is the name of the main router in the default configuration.

When IG is installed and running in development mode, as described in "*First Steps*", access Studio on `http://openig.example.com:8080/openig/studio`. The welcome screen is displayed:

## Welcome to ForgeRock Identity Gateway

Running in development mode - by default, all endpoints are open and accessible. Do not use this mode in a production environment.

ForgeRock Identity Gateway bridges your applications to the digital identity world. Use Identity Gateway Studio to develop and prototype protection for your applications and APIs.

- Authenticate Users
- Enforce Authorization Policies
- Throttle Requests
- Analyze Access

**Protect an Application**

**View Saved Routes**

**View the Documentation**

For information about how to restrict access to Studio, see "Restricting Access to Studio in Development Mode" in the *Gateway Guide*.

## 3.2. Summary of Studio Tasks, Route Status, and Icons

The following tables summarize the basic tasks in Studio, and the route status:

### Task Reference

To do this	Do this
Create a new route.	Select  Protect an Application, and then select  Structured.  Select  ROUTES,  Create a route, and then select  Structured.
Select a route	Select  ROUTES, and then select a route to view.
Display the config of a selected route.	Select a route, and then select  and  Display.
Deploy a selected route.	Select a route, and then select  Deploy.

To do this	Do this
Undeploy a selected route.	Select a deployed route, and then select  and  Undeploy.
Change the basic config of a selected route.	Select a route, and then select  Route settings. Edit the route and save the changes.

### Route Status

Status	Description	Action
Undeployed	The route is saved in Studio but is not deployed to the backend.	Deploy the route. The status changes to  Deployed.
Deployed	The route is saved in Studio and deployed to the backend.	None. The route has the same configuration in Studio and the backend.
Changes pending	The route has been deployed and then subsequently changed in Studio.	Deploy the route. The status changes to  Deployed.
Out of sync	The route has been deployed and then subsequently changed in the backend, or in both Studio and the backend.	<p>Select  Deploy. A message informs you that a different version of the route is deployed in the backend. Select  Deploy or  Import a route.</p> <p> <b>Deploy:</b> The version in Studio overwrites the backend.</p> <p> <b>Import a route:</b> The version in the backend overwrites Studio.</p> <p>When you import a route into Studio you go into <b>editor mode</b>. You can use the JSON editor to manually edit the route, but can no longer use the full Studio interface to add or edit filters.</p>
Compatibility update required	The route was created in Studio in an earlier version of IG. Some information is needed to complete the upgrade.	Enter the information as prompted, and then select  Deploy to deploy the route.

### Icons





Icon	Mode	Description
	Structured Studio	The route was created and edited using the menus and options in Studio.

Icon	Mode	Description
{ }	Editor Mode	The route was imported into Studio, or was created in Studio and then edited in editor mode.
☰	Freeform Studio	The route was created in Freeform Studio, available in Technology Preview. For more information, see " <i>Technology Preview of Freeform Studio</i> "

### 3.3. Creating Simple Routes

This section describes procedures to configure the basic settings for a route in Studio.

#### *To Create a Simple Route*

1. Browse to `http://openig.example.com:8080/openig/studio`, and select  Protect an Application.
2. Choose  Structured, and enter a URL for the application you want to protect, followed by a path condition to access the route. For example, enter `http://app.example.com:8081/my-basic-route`.
3. On the top-right of the screen, select  and  Display to review the route.





A route similar to this is displayed, where the path condition is used for the route name:

```
{
  "name": "my-basic-route",
  "baseURI": "http://app.example.com:8081",
  "condition": "${matches(request.uri.path, '^/my-basic-route')}",
  "handler": "ReverseProxyHandler"
}
```

### 3.4. Changing the Basic Settings of a Route

After creating a route, you can always change its basic settings.





#### *To Change the Basic Settings of a Route*

1. In Studio, select  ROUTES and then select a route.
2. On the top-right of the screen select  Route settings.
3. Using the on-screen hints for guidance, change the name, condition, or other features of the route, and save the changes.
4. On the top-right of the screen, select  and  Display to review the route.

## 3.5. Deploying Routes to Your Configuration



After creating or importing a route in Studio, deploy it to your IG configuration for testing.

### To Deploy a Route to Your Configuration

1. In Studio, select  ROUTES and then select a route.
2. On the top-right of the screen, select  and  Display to review the route.
3. If the route is okay, select  Deploy to push the route to the IG configuration.

#### Important

If the route configuration is not valid, or if a service that the route relies on, such as an AM service, is not available, the route fails to deploy.




The route status  Deployed is displayed, and the  Deploy button is grey and disabled.

4. Check the `$HOME/.openig/config/routes` folder in your IG configuration to see that the route is there.

By default, routes are loaded automatically into the IG configuration. You don't need to stop and restart IG. For information about reloading routes, see "Preventing the Reload of Routes" in the *Gateway Guide*.

5. Check the system log messages to confirm that the route was loaded successfully into the configuration. For information about logs, see "Logging Events" in the *Gateway Guide*.

### To Undeploy a Route

1. In Studio, select  ROUTES and then select a route.
2. On the top-right of the screen, select  and  Undeploy, and then confirm your request.

The route is removed from the IG configuration. On the Studio screen, the route status  Deployed is no longer displayed, and the  Deploy option is active again.

## 3.6. Adding Filters to a Route


After creating a route in Studio, you can add filters to the route. For examples, see the following sections of the *Gateway Guide*:

- For authentication and authorization, see "Enforcing Policy Decisions From AM" in the *Gateway Guide* and "Acting as an OAuth 2.0 Resource Server" in the *Gateway Guide*.
- For throttling, see "Throttling the Rate of Requests to Protected Applications" in the *Gateway Guide*.




- For scriptable filters, reference scripts, and scriptable throttling rates, see "Scripting in Studio" in the *Gateway Guide* and "Configuring a Scriptable Throttling Filter" in the *Gateway Guide*.

Filters are added to a chain that ends with a ReverseProxyHandler. To view the chain, select  Chain on the left of the screen. For information about chains, see Chain(5) in the *Configuration Reference*.

Note the following ways to manage filters in a chain:


- To move a filter to another position in the chain, simply drag it.
- To edit a filter in the chain, select  for the filter.
- To remove a filter from the chain, deselect ENABLED. The filter is disabled and removed from the chain. If you enable the filter again, the configuration is restored, and you don't have to enter the data again.
- To add a disabled filter back in the chain, select ENABLED.

### To Add Other Filters to a Route

1. In Studio, select  ROUTES and then select a route.
2. Select  Other filters,  New filter, and then Other filter.
3. In Type, select a filter type from the list, and then optionally enter a name and configuration for the filter.

#### Note



Studio checks that the JSON is valid, but doesn't check that the configuration of the filter is valid. If the filter configuration isn't valid, the route fails to load when deployed.

When you save, the filter is added to the chain. Select  Chain on the left of the screen to view the chain.

4. Deploy the route as described in "Deploying Routes to Your Configuration".

## 3.7. Editing Routes In Editor Mode




After creating a route in Studio, you can edit it by using the options offered in Studio, or by switching to editor mode and using the JSON editor.

Routes created only in the menus have the icon , and routes edited in editor mode have the icon .

**Important**

When you go into editor mode, you can use the JSON editor to manually edit the route, but can no longer use the full Studio interface to add or edit filters.

### *To Edit and Redeploy a Route*

1. In Studio, select  ROUTES and then select a route.
2. Edit the route in Studio or manually:
  - To edit in Studio, select options offered in Studio.
  - To edit manually, select  and  Editor mode, and use the JSON editor to edit the route.



The route status is  Changes pending.

3. Deploy the route as described in "To Deploy a Route to Your Configuration".

## 3.8. Importing Routes Into Studio


When you import a route into Studio, it is imported in editor mode. You can use the JSON editor to manually edit the route, but can't use the full Studio interface to add or edit filters.

### *To Import a Route Into Studio*

1. In Studio, select  ROUTES and then  Import a route.
2. Click in the window to import a route, or drag a route from your filesystem.

If the route has a `name` property, the name is automatically used for the `Name` and `ID` fields in Studio.


3. If necessary, make the following changes, and then select Import:
  - If the `Name` and `ID` fields are empty, enter a unique name and ID for the route.
  - If the `Name` and `ID` fields are outlined in red, the route name or ID already exists in Studio. Change the name and ID to be unique.
  - If an error message is displayed, the route is not valid JSON. Fix the route and then try again to import it.




The route is added to the list of routes on the  ROUTES page.


4. Deploy the route as described in "Deploying Routes to Your Configuration".



## 3.9. Viewing and Searching for Routes in Your Configuration

All of the routes that exist in your backend configuration are displayed on the  ROUTES page, including imported routes and routes created outside of Studio.

On the  ROUTES page, routes created only in the menus have the icon , and routes edited in editor mode have the icon .

To search for a route, select  ROUTES, and type part of the name or URL of the route in the search box (Q). Routes that match are displayed as you enter the search criteria.

## 3.10. Restricting Access to Studio

When IG is running in development mode, by default the Studio endpoint is open and accessible. To allow only specific users to access Studio, configure a `StudioProtectionFilter` with a `SingleSignOnFilter` or `CrossDomainSingleSignOnFilter`.

The following example uses a `SingleSignOnFilter` to require users to authenticate with AM before they can access Studio, and protects the request from Cross Site Request Forgery (CSRF) attacks.

### *To Require Authentication to Access Studio in Development Mode*

1. Select Applications > Agents > Identity Gateway, add an agent with the following values:
  - Agent ID: `ig_agent`
  - Password: `password`

Leave all other values as default.

2. Set an environment variable for the IG agent password, and then restart IG:

```
$ export AGENT_SECRET_ID='cGFzc3dvcmQ='
```

The password is retrieved by a `SystemAndEnvSecretStore`, and must be base64-encoded.


3. Add the following file to the IG configuration as `$HOME/.openig/config/admin.json` (on Windows, `%appdata%\OpenIG\config\admin.json`):

```
{
  "prefix": "openig",
  "mode": "DEVELOPMENT",
  "properties": {
    "SsoTokenCookieOrHeader": "iPlanetDirectoryPro"
  },
  "heap": [
    {
      "name": "SystemAndEnvSecretStore-1",
      "type": "SystemAndEnvSecretStore"
    },
    {
```

```
"name": "AmService-1",
"type": "AmService",
"config": {
  "agent": {
    "username": "ig_agent",
    "passwordSecretId": "agent.secret.id"
  },
  "secretsProvider": "SystemAndEnvSecretStore-1",
  "url": "http://openam.example.com:8088/openam/",
  "ssoTokenHeader": "&{SsoTokenCookieOrHeader}",
  "version": "${platform.version}"
}
},
{
  "name": "StudioProtectionFilter",
  "type": "ChainOfFilters",
  "config": {
    "filters": [
      {
        "type": "SingleSignOnFilter",
        "config": {
          "amService": "AmService-1"
        }
      },
      {
        "type": "CsrfFilter",
        "config": {
          "cookieName": "&{SsoTokenCookieOrHeader}",
          "failureHandler": {
            "type": "StaticResponseHandler",
            "config": {
              "status": 403,
              "headers": {
                "Content-Type": [
                  "text/plain"
                ]
              }
            }
          },
          "entity": "Request forbidden"
        }
      }
    ]
  }
}
]
```

Notice the following features of the file:

- The `prefix` sets the base of the administrative route to the default value `/openig`. The Studio endpoint is therefore `/openig/studio`.
- The `mode` is `development`, so by default the Studio endpoint is open and unfiltered.
- The `properties` object sets a configuration parameter for the value of the SSO token cookie or header, which is used in `AmService` and `CsrfFilter`.

- The AmService uses the IG agent in AM for authentication.
  - The agent password for AmService is provided by a SystemAndEnvSecretStore in the heap.
  - The StudioProtectionFilter calls the SingleSignOnFilter to redirect unauthenticated requests to AM, and uses the CsrfFilter to protect requests from CSRF attacks. For more information, see `SingleSignOnFilter(5)` in the *Configuration Reference* and `CsrfFilter(5)` in the *Configuration Reference*.
4. Restart IG to take into account the changes to `admin.json`.
  5. Access Studio on `http://openig.example.com:8080/openig/studio`. The `SingleSignOnFilter` redirects the request to AM for authentication.
  6. Log in to AM with username `demo`, password `changeit`. The Studio  ROUTES screen is displayed.

# Appendix A. Technology Preview of Freeform Studio



Freeform Studio is a new user interface to develop complex routes of filters and handlers. As you design a route, Freeform Studio helps you to visualize the chain of filters and handlers, identify break points, and track the path of requests, responses, and contexts.

Freeform Studio is offered as Technology Preview, as defined in "*Release Levels and Interface Stability*" in the *Release Notes*, and is not fully documented. This appendix gives some pointers about how to access Freeform Studio and get started.

Freeform Studio adheres to the same requirements as Studio. For more information, see "Accessing Studio".

Freeform Studio offers a new interface to develop routes, but deploys them to the backend in the same way as Studio, and uses the same set of route statuses. For more information, see "Route Status".

## To Create a Route

1. Browse to <http://openig.example.com:8080/openig/studio>, and select  Protect an Application.
2. Select  Freeform.
3. In Application URL, enter a URL for the application you want to protect followed by a path condition, and then select Create route.
4. Drag handlers and filters from the components bar onto the canvas, to begin designing routes.