Getting Started Guide

/ Identity Gateway 7

Latest update: 7.0.2

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

Abstract

Guide to using ForgeRock® Identity Gateway for new users and readers evaluating the product.

Table of Contents

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.

About This Guide

This guide introduces IG, and describes how to set up and run the product. It 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

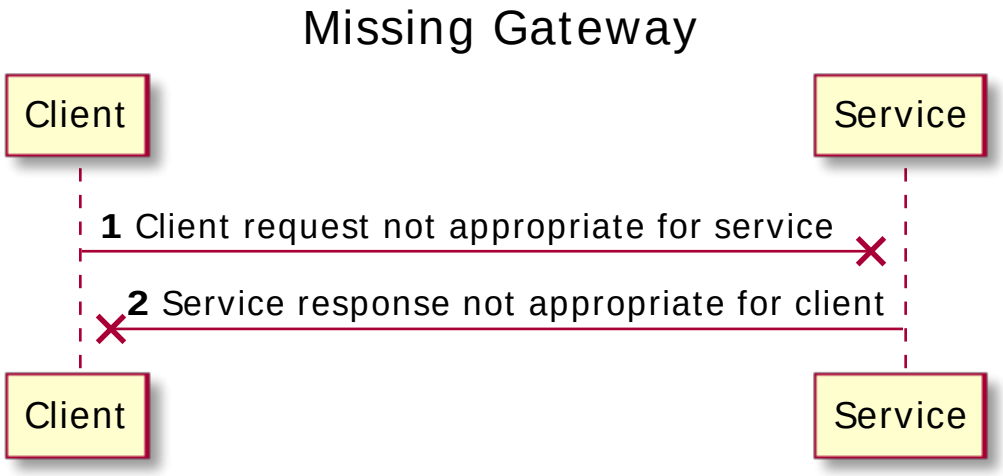
Chapter 1

About IG

IG integrates web applications, APIs, and microservices with the ForgeRock Identity Platform. For more information about IG, see "*About IG*" in the *Gateway Guide*.

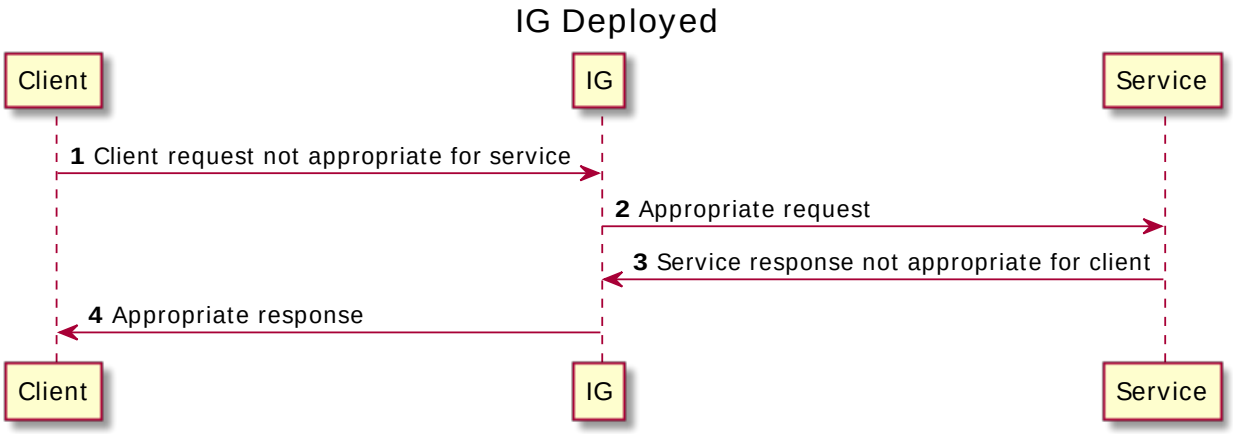
Services That Cannot Be Integrated Into Newer Architectures

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. The following image illustrates an example of a missing gateway.



IG As an HTTP Gateway

IG works as an HTTP gateway, based on reverse proxy architecture. It is deployed on a network so it can intercept client requests and server responses. The following image illustrates a 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.

IG Features

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

IG Studio

IG provides a user interface to help you design and develop routes to protect applications. For information, see the Studio User Guide.

Chapter 2

Preparing to Install

Requirements

For detailed information about the requirements for running IG, see "*Requirements*" in the *Release Notes*. The following software is required:

- Oracle JDK 11 or later versions, or OpenJDK 11 or later versions.

Configuring the Network

Configure the network to route network traffic to IG and the sample application. The examples in the guide assume that:

- IG is reachable on `http://openig.example.com:8080`.
- The sample application is reachable on `http://app.example.com:8081/home`.
- AM is reachable on `http://openam.example.com:8088/openam`.

Before you try out the examples, configure the network to include the hosts.

Configure the Network

- Add the following additional entry to your `/etc/hosts` file on UNIX systems or `%SystemRoot%\system32\drivers\etc\hosts` on Windows:

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

  For more information about host files, see the Wikipedia entry, *Hosts (file)*.

Chapter 3

Downloading and Starting IG

The following sections describe options for downloading and starting IG. For information about other installation options, such as setting the default location of the configuration folders, and configuring for HTTPS, see "*Installation in Detail*" in the *Gateway Guide*.

- "Downloading and Starting IG in Standalone Mode"
- "Downloading and Starting IG in Tomcat"
- "Downloading and Starting IG in Jetty"
- "Downloading and Starting IG in JBoss"

Downloading and Starting IG in Standalone Mode

This section describes how to install and start IG in standalone mode, from a .zip file.

Downloading the IG .zip File

Download IG

1. Create a local installation directory for IG. The examples in this section use `/path/to`.
2. Download `IG-7.0.2.zip` from the ForgeRock BackStage download site, and copy the .zip file to the installation directory:
   ```
   $ cp IG-7.0.2.zip /path/to/IG-7.0.2.zip
   ```
3. Unzip the file:
   ```
   $ unzip IG-7.0.2.zip
   ```
   The directory `/path/to/identity-gateway` is created.

Starting IG With Default Settings

Use the following step to start the instance of IG, specifying the configuration directory where IG looks for configuration files. IG starts up by default on port `8080`, and `route-system.log` is created in the `logs` directory.

Start IG With Default Settings

1. Start IG:

```
$ /path/to/identity-gateway/bin/start.sh
...
... started in 1234ms on ports : [8080]
```

   By default, the base location for IG configuration files is in `$HOME/.openig`, where `$HOME/.openig` is the instance directory. To read the configuration from a different location, specify the base location as an argument. The following example reads the configuration from the `config` directory under `/path/to/instance-dir`:

```
$ /path/to/identity-gateway/bin/start.sh /path/to/instance-dir
```

2. Check that IG is running in one of the following ways:

   - Ping IG at http://openig.example.com:8080/openig/ping, and make sure an `HTTP 200` is returned.
   - Access the IG welcome page at http://openig.example.com:8080.
   - When IG is running in development mode, display the product version and build information at http://openig.example.com:8080/openig/api/info.

Starting IG With Custom Settings

By default IG runs on HTTP, on port `8080`, from the instance directory `$HOME/.openig`.

The `vertx` property of `admin.json` finely-tune Vert.x instances. The `connectors` property of `admin.json` customizes server ports, TLS, and Vert.x-specific configurations. Each `connectors` object represents the configuration of an individual port.

To run IG on a different port, with custom settings, add the configuration file `$HOME/.openig/config/admin.json`, and restart IG. The following example starts IG on non-default ports, and configures Vert.x-specific options for the connection on port 9091:

```
{
  "connectors": [{
    "port": 9090
  },
  {
    "port": 9091,
    "vertx": {
      "maxWebSocketFrameSize": 128000,
      "maxWebSocketMessageSize": 256000,
      "compressionLevel": 4
    }
  }]
}
```

For information about the configuration of `connectors`, see "AdminHttpApplication (`admin.json`)" in the *Configuration Reference*.

Configure Environment Variables and System Properties

Configure environment variables and system properties for IG, as follows:

- By adding environment variables on the command line when you start IG.
- By adding environment variables in `$HOME/.openig/bin/env.sh`, where `$HOME/.openig` is the instance directory.

Starting IG With a Customized Router Scan Interval

By default, IG scans every 10 seconds for changes to the route configuration files. Any changes to the files are automatically loaded into the configuration without restarting IG. For more information about the router scan interval, see "Router" in the *Configuration Reference*.

The following example overwrites the default value of the Router scan interval to two seconds when you start up IG:

```
$ IG_ROUTER_SCAN_INTERVAL='2 seconds' /path/to/identity-gateway/bin/start.sh
```

Specifying Environment Variables for Key and JVM Options

The following example specifies environment variables in the IG `env.sh` file to customize JVM options and keys:

```
# Specify JVM options
JVM_OPTS="-Xms256m -Xmx2048m"

# Specify the DH key size for stronger ephemeral DH keys, and to protect against weak keys
JSSE_OPTS="-Djdk.tls.ephemeralDHKeySize=2048"

# Wrap them up into the JAVA_OPTS environment variable
export JAVA_OPTS="${JAVA_OPTS} ${JVM_OPTS} ${JSSE_OPTS}"
```

Stopping IG

Stop IG

- In the terminal where IG is running, select CTRL+C to stop the service.

Downloading and Starting IG in Tomcat

The commands in this guide assume that you install Tomcat to `/path/to/tomcat`, and after installation, you have a directory `/path/to/tomcat/webapps` in which you install IG. If you use another directory structure, substitute the commands.

Download and Start IG in Tomcat

1. Download a supported version of Tomcat server from its download page, and install it to `/path/to/tomcat`.
2. Remove the `ROOT` directory in Tomcat:

   ```
   $ rm -rf /path/to/tomcat/webapps/ROOT
   ```
3. Download `IG-7.0.2.war` from the ForgeRock BackStage download site.
4. Copy the `IG-7.0.2.war` to the Tomcat `webapps` directory, as `ROOT.war`:

   ```
   $ cp IG-7.0.2.war /path/to/tomcat/webapps/ROOT.war
   ```

   Tomcat automatically deploys IG in the root context on startup.
5. Start Tomcat:

   ```
   $ /path/to/tomcat/bin/startup.sh
   ```

   If necessary, make the startup scripts executable.
6. Check that IG is running in one of the following ways:
   - Ping IG at http://openig.example.com:8080/openig/ping, and make sure an `HTTP 200` is returned.
   - Access the IG welcome page at http://openig.example.com:8080.
   - When IG is running in development mode, display the product version and build information at http://openig.example.com:8080/openig/api/info.

Downloading and Starting IG in Jetty

The commands in this guide assume that 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.

Download and Start IG in Jetty

1. Download a supported version of Jetty server from its download page, and install it to `/path/to/jetty`.

2. Download `IG-7.0.2.war` from the ForgeRock BackStage download site.

3. Copy the .war file:

```
$ cp IG-7.0.2.war /path/to/jetty/webapps/IG-7.0.2.war
```

   Jetty automatically deploys IG in the root context on startup.

4. 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
```

5. Check that IG is running in one of the following ways:

   - Ping IG at http://openig.example.com:8080/openig/ping, and make sure an `HTTP 200` is returned.
   - Access the IG welcome page at http://openig.example.com:8080.
   - When IG is running in development mode, display the product version and build information at http://openig.example.com:8080/openig/api/info.

Downloading and Starting IG in JBoss

This section installs JBoss to `/path/to/jboss`. If you use another directory structure, substitute the commands.

Download and Start IG in JBoss

1. Download a supported version of JBoss server from its download page, and install it to `/path/to/jboss`.

2. In the JBoss configuration file `/path/to/jboss/standalone/configuration/standalone.xml`, delete the line for the JBoss welcome-content handler:

```
<server name="default-server">
  <host name="default-host" alias="localhost">
    <location name="/" handler="welcome-content"/> <!-- Delete this line -->
```

3. Download `IG-7.0.2.war` from the ForgeRock BackStage download site.

4. Copy the `IG-7.0.2.war` to the JBoss deployment directory:

```
$ cp IG-7.0.2.war /path/to/jboss/standalone/deployments/IG-7.0.2.war
```

5. Start JBoss as a standalone server:

```
$ /path/to/jboss/bin/standalone.sh
```

JBoss deploys IG in the root context.

6. Check that IG is running in one of the following ways:

- Ping IG at http://openig.example.com:8080/openig/ping, and make sure an `HTTP 200` is returned.
- Access the IG welcome page at http://openig.example.com:8080.
- When IG is running in development mode, display the product version and build information at http://openig.example.com:8080/openig/api/info.

Chapter 4

Downloading and Starting the Sample Application

ForgeRock provides a mockup web application for testing IG configurations. The sample application is used in the examples in this guide and the *Gateway Guide*.

Downloading and Starting the Sample Application

Download and Start the Sample Application

1. Download `IG-sample-application-7.0.2.jar`, from the ForgeRock BackStage download site.

2. Start the sample application:

   ```
   $ java -jar IG-sample-application-7.0.2.jar
   Preparing to listen for HTTP on port 8081.
   Preparing to listen for HTTPS on port 8444.
   ...
   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-7.0.2.jar 8888 8889
   ```

3. Check that the sample application is running in one of the following ways:

   - Go to http://localhost:8081/home to access the home page of the sample application. Information about the browser request is displayed.

   - Go 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.

Stop the Sample Application

- In the terminal where the sample application is running, select CTRL+C to stop the app.

Serving Static Resources for the Sample Application

When the sample application is used with IG in the examples in this guide and the *Gateway Guide*, the sample application must serve static resources, such as the .css. Add the following route to the IG configuration, as:

Linux

```
$HOME/.openig/config/routes/static-resources.json
```

Windows

```
%appdata%\OpenIG\config\routes\static-resources.json
```

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

Configuration Options for the Sample Application

To view the command-line options for the sample application, start it with the `-h` option:

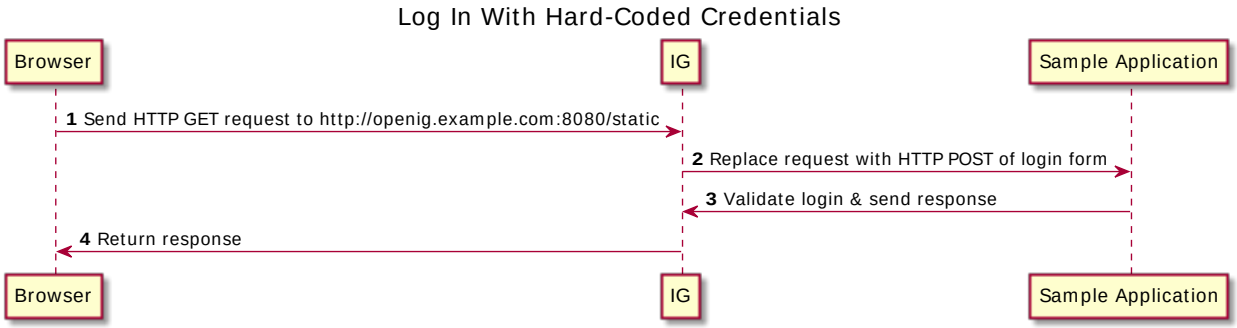
```
$ java -jar IG-sample-application-7.0.2.jar -h
Usage: <main class> [options]
Options:
--http
The HTTP port number.
Default: 8081
--https
The HTTPS port number.
Default: 8444
--session
The session timeout in seconds.
Default: 60
--am-discovery-url
The AM URL base for OpenID Provider Configuration.
Default: http://openam.example.com:8088/openam/oauth2
-h, --help
Default: false
```

Chapter 5

Protecting an Application With IG

This section gives a simple example of how to use IG to protect an application. For many more examples of how to protect applications with IG, see the Gateway Guide.

In the following example, a browser requests access to the sample application, and IG intercepts the request to log the user into the application. The following image shows the flow of data in the example:



1. The browser sends an HTTP GET request to the HTTP server on `openig.example.com`.
2. IG replaces the HTTP GET request with an HTTP POST login request containing credentials to authenticate.
3. The sample application validates the credentials, and returns the page for the user `demo`.

   If IG did not provide the credentials, or if the sample application couldn't validate the credentials, the sample application returns the login page.
4. IG returns this response to the browser.

Configure IG to Log You in to an Application

1. Set up IG as described in "*Downloading and Starting IG*", and the sample application as described in "*Downloading and Starting the Sample Application*".
2. Add the following route to IG, to serve .css and other static resources for the sample application:

   Linux

```
$HOME/.openig/config/routes/static-resources.json
```

Windows

```
%appdata%\OpenIG\config\routes\static-resources.json
```

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

3. Add the following route to IG:

Linux

```
$HOME/.openig/config/routes/01-static.json
```

Windows

```
%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')}"
}
```

Notice the following features of the route:

- The route matches requests to `/static`.
- The StaticRequestFilter replaces the request with an HTTP POST, specifying the resource to post the request to, and a form to include in the request. The form includes credentials for the username `demo`.

- The ReverseProxyHandler replays the request to the sample application.

4. Check that the route system log includes a message that the new files are loaded into the config:

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

5. Go to http://openig.example.com:8080/static.

   You are directed to the sample application, and logged in automatically with the username `demo`.

Chapter 6

Next steps

This section describes some basic options to help you with IG. For information about other installation options, such as setting the default location of the configuration folders, and configuring for HTTPS, see "*Installation in Detail*" in the *Gateway Guide*.

- "Adding a Base Configuration File"
- "Adding a Default Route"
- "Switching from Production Mode to Development Mode"
- "Using IG Studio"

Adding a Base Configuration File

The entry point for requests coming in to IG is a JSON-encoded configuration file, expected by default at:

Linux

```
$HOME/.openig/config/config.json
```

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 "Default Configuration" in the *Configuration Reference*. The default looks for routes in:

Linux

```
$HOME/.openig/config/routes
```

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 (`config.json`)" in the *Configuration Reference*, "Heap Objects" in the *Configuration Reference*, and "Router" in the *Configuration Reference*.

Add a Base Configuration for IG

1. Add the following file to IG:

   Linux

   ```
   $HOME/.openig/config/config.json
   ```

   Windows

   ```
   %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" in the *Configuration Reference*.

2. Stop and restart IG.

3. Check that the route system log includes a message that the file is loaded into the config:

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

Adding a Default Route

When there are multiple routes in the IG configuration, they are ordered lexicographically, by route name. For example, `01-static.json` is ordered before `zz-default.json`.

When IG processes a request, the request traverses the routes in the configuration. If the request matches the condition for `01-static.json` it is processed by that route. Otherwise, it passes to the next route in the configuration. If a route has no condition, it can process any request.

A default route is the last route in a configuration to which a request is routed. If a request matches no other route in the configuration, it is processed by the default route.

Add a Default Route to the Configuration

1. Add the following route to IG:

   Linux

   ```
   $HOME/.openig/config/routes/zz-default.json
   ```

   Windows

   ```
   %appdata%\OpenIG\config\routes\zz-default.json
   ```

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

   Notice the following features of the route:

   - The route name starts with `zz`, so it is the last route that is loaded into the configuration.
   - There is no `condition` property, so the route processes all requests.
   - The route calls a ReverseProxyHandler with the default configuration, which proxies the request to the application and returns the response, without changing either the request or the response.

2. Check that the route system log includes a message that the 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'
   ```

Switching from Production Mode to Development Mode

After installation, to prevent unwanted changes to the configuration, IG is by default in production mode. Access is restricted as follows:

- The `/routes` endpoint is not exposed.
- You cannot manage, list, or even read routes through Common REST.
- Studio is effectively disabled.
- The `/share` and `api/info` endpoints are exposed only to the loopback address.

Switch to development mode in one of the following ways, applied in order of precedence:

1. Add the following route to IG, and restart IG:

   Linux

   ```
   $HOME/.openig/config/admin.json
   ```

   Windows

   ```
   %appdata%\OpenIG\config\admin.json
   ```

   Web container mode

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

   Standalone mode

   ```
   {
     "mode": "DEVELOPMENT",
     "connectors": [
       { "port" : 8080 }
     ]
   }
   ```

   For more information, see "AdminHttpApplication (`admin.json`)" in the *Configuration Reference*

2. Define an environment variable for the configuration token `ig.run.mode`, and then start IG in the same terminal.

   If `mode` is not defined in `admin.json`, the following example starts a standalone instance of IG in development mode:

   ```
   $ IG_RUN_MODE=development /path/to/identity-gateway/bin/start.sh
   ```

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

If `mode` is not defined in `admin.json`, or an `IG_RUN_MODE` environment variable is not set, the following file starts a standalone instance of IG with the system property `ig.run.mode` to force development mode:

Linux

```
$HOME/.openig/env.sh
```

Windows

```
%appdata%\OpenIG\env.sh
```

```
export JAVA_OPTS='-Dig.run.mode=development'
```

For information about restricting access to Studio in development mode, see "*Restricting Access to Studio*" in the *Studio User Guide*. For information about switching back to production mode, see "Switch From Development Mode to Production Mode" in the *Maintenance Guide*.

Using IG Studio

IG Studio is a user interface to help you build and deploy your IG configuration. For information about using Studio, see the Studio User Guide.