


## Getting started

---

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> .

This guide introduces IG, and provides a quick way to set up and run the product. For more installation and set up options, refer to the [Installation guide](#).

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

## Prepare to install

---

### Requirements

Make sure your installation meets the requirements in the [release notes](#).

### Create an IG service account

To limit the impact of a security breach, install and run IG from a dedicated service account. This is optional when you are evaluating IG, but essential in production installations.

A hacker is constrained by the rights granted to the user account where IG runs; therefore, never run IG as root user.

1. In a terminal window, use a command similar to the following to create a service account:

1. Linux

## 2. Windows

```
$ sudo /usr/sbin/useradd \  
--create-home \  
--comment "Account for running IG" \  
--shell /bin/bash IG
```

```
> net user username password /add /comment:"Account for  
running IG"
```

### 2. Apply the principle of least privilege to the account, for example:

- Read/write permissions on the installation directory, `/path/to/identity-gateway`.
- Execute permissions on the scripts in the installation `bin` directory, `/path/to/identity-gateway/bin`.

## Prepare the network

Configure the network to include the hosts.

### 1. Add the following additional entry to your host file:

1. Linux
2. Windows

```
/etc/hosts
```

```
%SystemRoot%\system32\drivers\etc\hosts
```

```
127.0.0.1 localhost ig.example.com app.example.com  
am.example.com
```

For more information about host files, refer to the Wikipedia entry, [Hosts \(file\)](#) 

## Set up Identity Cloud and AM for use with IG

This section contains procedures for setting up items in ForgeRock Identity Cloud and AM that you can use with IG. For more information about setting up Identity Cloud, refer to

the [ForgeRock Identity Cloud docs](#). For more information about setting up AM, refer to the [Access Management docs](#).

## Authenticate an IG agent to Identity Cloud

### IMPORTANT

IG agents are automatically authenticated to Identity Cloud by a non-configurable authentication module. Authentication chains and modules are deprecated in Identity Cloud and replaced by journeys.

You can now authenticate IG agents to Identity Cloud with a journey. The procedure is currently optional, but will be required when authentication chains and modules are removed in a future release of Identity Cloud.

For more information, refer to Identity Cloud's [Journeys](#).

This section describes how to create a journey to authenticate an IG agent to Identity Cloud. The journey has the following requirements:

- It must be called `Agent`
- Its nodes must pass the agent credentials to the Agent Data Store Decision node.

When you define a journey in Identity Cloud, that same journey is used for all instances of IG, Java agent, and Web agent. Consider this point if you change the journey configuration.

1. Log in to the Identity Cloud admin UI as an administrator.
2. Click **Journeys > New Journey**.
3. Add a journey with the following information and click **Create journey**:
  - **Name:** Agent
  - **Identity Object:** The user or device to authenticate.
  - (Optional) **Description:** Authenticate an IG agent to Identity Cloud

The journey designer is displayed, with the `Start` entry point connected to the `Failure` exit point, and a `Success` node.

4. Using the **Q Filter nodes** bar, find and then drag the following nodes from the **Components** panel into the designer area:
  - Zero Page Login Collector node to check whether the agent credentials are provided in the incoming authentication request, and use their values in the following nodes.

This node is required for compatibility with Java agent and Web agent.

- Page node to collect the agent credentials if they are not provided in the incoming authentication request, and use their values in the following nodes.
- Agent Data Store Decision node to verify the agent credentials match the registered IG agent profile.

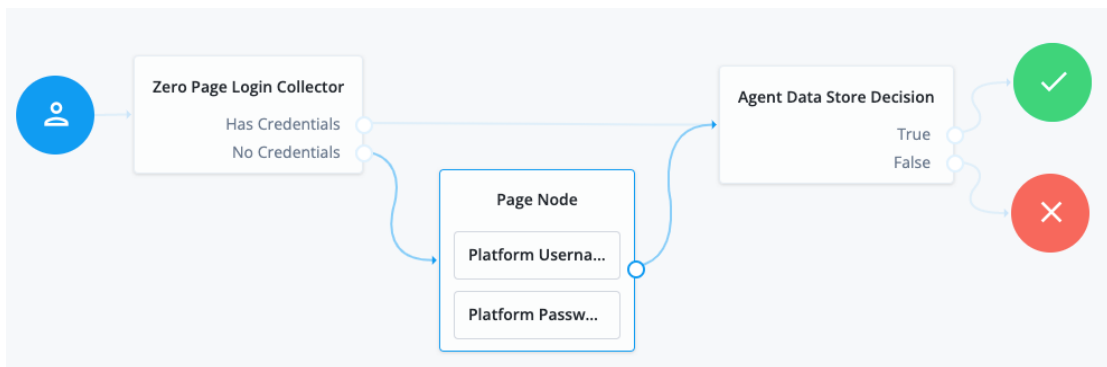
**IMPORTANT**

Many nodes can be configured in the panel on the right side of the page. Unless otherwise stated, do not configure the nodes, and use only the default values.

5. Drag the following nodes from the **Components** panel into the Page node:

- Platform Username node to prompt the user to enter their username.
- Platform Password node to prompt the user to enter their password.

6. Connect the nodes as follows and save the journey:



## Authenticate an IG agent to AM

**IMPORTANT**

IG agents are automatically authenticated to AM by a non-configurable authentication module in AM. Authentication chains and modules were deprecated in AM 7 and are replaced by authentication nodes and trees.

From AM 7.3 you can authenticate IG agents to AM by using authentication nodes and trees. The procedure is currently optional, but will be required when authentication chains and modules are removed in a future release of AM.

For more information, refer to AM's [Authentication Nodes and Trees](#).

This section describes how to create an authentication tree to authenticate an IG agent to AM. The tree has the following requirements:




- It must be called Agent
- Its nodes must pass the agent credentials to the Agent Data Store Decision node.

When you define a tree in AM, that same tree is used for all instances of IG, Java agent, and Web agent. Consider this point if you change the tree configuration.

1. On the **Realms** page of the AM admin UI, choose the realm in which to create the authentication tree.
2. On the **Realm Overview** page, click **Authentication > Trees > + Create tree**.
3. Create a tree named Agent .

The authentication tree designer is displayed, with the **Start** entry point connected to the **Failure** exit point, and a **Success** node.

The authentication tree designer provides the following features on the toolbar:

Button	Usage
	Lay out and align nodes according to the order they are connected.
	Toggle the designer window between normal and full-screen layout.
	Remove the selected node. Note that the <b>Start</b> entry point cannot be deleted.

4. Using the **Filter** bar, find and then drag the following nodes from the **Components** panel into the designer area:
  - Zero Page Login Collector node to check whether the agent credentials are provided in the incoming authentication request, and use their values in the following nodes.

This node is required for compatibility with Java agent and Web agent.
  - Page node to collect the agent credentials if they are not provided in the incoming authentication request, and use their values in the following nodes.
  - Agent Data Store Decision node to verify the agent credentials match the registered IG agent profile.

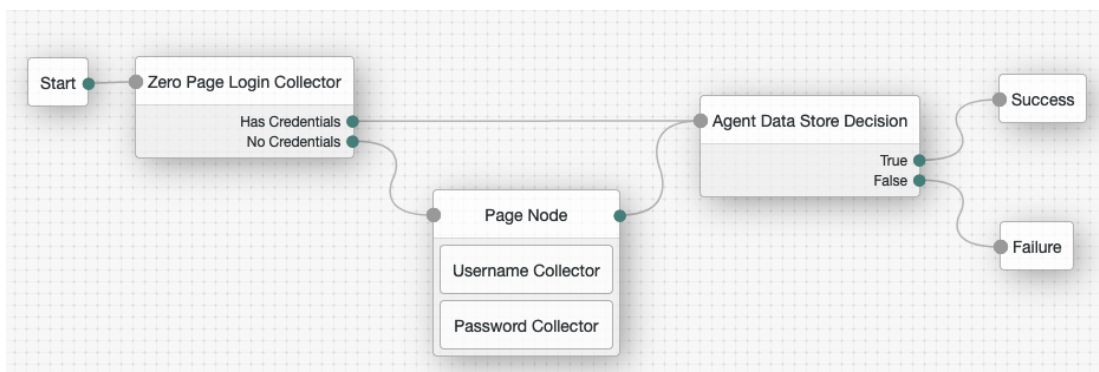
**IMPORTANT**

Many nodes can be configured in the panel on the right side of the page. Unless otherwise stated, do not configure the nodes and use only the default values.

5. Drag the following nodes from the **Components** panel into the Page node:

- Username Collector node to prompt the user to enter their username.
- Password Collector node to prompt the user to enter their password.

6. Connect the nodes as follows and save the tree:



## Register an IG agent in Identity Cloud

This procedure registers an agent that acts on behalf of IG.

1. Log in to the Identity Cloud admin UI as an administrator.
2. Click **Gateways & Agents** > **+ New Gateway/Agent** > **Identity Gateway** > **Next**, and add an agent profile:
  - **ID:** *agent-name*
  - **Password:** *agent-password*


### IMPORTANT

Use secure passwords in a production environment. Consider using a password manager to generate secure passwords.

3. Click **Save Profile** > **Done**. The agent profile page is displayed.
4. To add a redirect URL for CDSSO, go to the agent profile page and add the URL.
5. To change the introspection scope, click **Native Consoles** > **Access Management**, and update the agent in the AM admin UI. By default, the agent can introspect OAuth 2.0 tokens issued to any client, in the realm and subrealm where it is created.

## Register an IG agent in AM 7 and later

In AM 7 and later versions, follow these steps to register an agent that acts on behalf of IG.

1. In the AM admin UI, select the top-level realm, and then select **Applications > Agents > Identity Gateway**.
2. Add an agent with the following values:
  1. For SSO
  2. For CDSSO
    - **Agent ID** : ig\_agent
    - **Password** : password
    - **Agent ID** : ig\_agent
    - **Password** : password
    - **Redirect URL for CDSSO** :  
<https://ig.ext.com:8443/home/cdsso/redirect> 

### *Register an IG agent in AM 6.5 and earlier*

In AM 6.5 and earlier versions, follow these steps to register an agent that acts on behalf of IG.



1. In the AM admin UI, select the top-level realm, and then select **Applications > Agents > Java (or J2EE)**.
2. Add an agent with the following values:
  1. For SSO
  2. For CDSSO
    - **Agent ID** : ig\_agent
    - **Agent URL** : <http://ig.example.com:8080/agentapp>
    - **Server URL** : <http://am.example.com:8088/openam>
    - **Password** : password
    - **Agent ID** : ig\_agent\_cdsso
    - **Agent URL** : <http://ig.ext.com:8080/agentapp>
    - **Server URL** : <http://am.example.com:8088/openam>
    - **Password** : password
3. On the **Global** tab, deselect **Agent Configuration Change Notification**.

This option stops IG from being notified about agent configuration changes in AM. IG doesn't need these notifications.
4. (For CDSSO) On the **SSO** tab, select the following values:
  - **Cross Domain SSO** : Deselect this option

- **CSSO Redirect URI** : /home/cdssso/redirect
5. (For CSSO and policy enforcement) On the **SSO** tab, select the following values:
- **Cross Domain SSO** : Deselect this option
  - **CSSO Redirect URI** : /home/pep-cdssso/redirect

### *Set up a demo user in Identity Cloud*

This procedure sets up a demo user in the alpha realm.

- a. Log in to the Identity Cloud admin UI as an administrator.
- b. Go to  **Identities > Manage >  Alpha realm - Users**, and add a user with the following values:
  - **Username**: demo
  - **First name**: demo
  - **Last name**: user
  - **Email Address**: demo@example.com
  - **Password**: Ch4ng3!t

### *Set up a demo user in AM*

AM is provided with a demo user in the top-level realm, with the following credentials:

- ID/username: demo
- Last name: user
- Password: Ch4ng31t
- Email address: demo@example.com
- Employee number: 123

For information about how to manage identities in AM, refer to AM's [Identity stores](#).

## Downloading, starting, and stopping 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 the [Installation guide](#).



## Download and start IG

### *Download the IG .zip file*

The .zip file unpacks into a `/path/to/identity-gateway` directory with the following content:

- `bin`: Start and stop executables
- `classes`: Initially empty; used to install patches from ForgeRock support
- `docker/Dockerfile`: Dockerfile and README to build an IG Docker image
- `legal-notices`: Licenses and copyrights
- `lib`: IG and third-party libraries

1. Create a local installation directory for IG. The examples in this section use `/path/to`.
2. Download `IG-2023.4.0.zip` from the [ForgeRock BackStage download site](#)<sup>↗</sup>, and copy the .zip file to the installation directory:

```
$ cp IG-2023.4.0.zip /path/to/IG-2023.4.0.zip
```

3. Unzip the file:

```
$ unzip IG-2023.4.0.zip
```

The directory `/path/to/identity-gateway` is created.

### *Start IG*

#### *Start IG with default settings*

Use the following step to start the instance of IG, specifying the configuration directory where IG looks for configuration files.

1. Start IG:
  1. Linux
  2. Windows

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

```
...
```

```
... started in 1234ms on ports : [8080 8443]
```

```
C:\path\to\identity-gateway\bin\start.bat
```

By default, IG configuration files are located under `$HOME/.openig` (on Windows `%appdata%\OpenIG`). For information about how to use a different location, refer to [Change the base location of the IG configuration](#).

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

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

### *Start IG with custom settings*

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

To start IG with custom settings, add the configuration file `admin.json` with the following properties, and restart IG:

- `vertx`: Finely tune Vert.x instances.
- `connectors`: Customize server port, TLS, and Vert.x-specific configurations. Each `connectors` object represents the configuration of an individual port.
- `prefix`: Set the instance directory, and therefore, the base of the route for administration requests.

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 more information, refer to [AdminHttpApplication \(admin.json\)](#).

## Stop IG

Use the `stop.sh` script to stop an instance of IG, specifying the instance directory as an argument. If the instance directory is not specified, IG uses the default instance directory:

1. Linux
2. Windows

```
$ /path/to/identity-gateway/bin/stop.sh $HOME/.openig
```

```
C:\path\to\identity-gateway\bin\stop.bat %appdata%\OpenIG
```

## Using 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 the sample application

1. Download `IG-sample-application-2023.4.0-SNAPSHOT.jar`, from the [ForgeRock BackStage download site](#).

## Starting the sample application

1. Start the sample application:

```
$ java -jar IG-sample-application-2023.4.0-SNAPSHOT.jar

[...] [INFO    ] All 8 verticles started on ports : [8081,
8444]
[...] [INFO    ] Using session timeout of: 60s
```

```
[...] [INFO ] Using AM base for OpenID Discovery URL:  
http://am.example.com:8088/openam/oauth2  
[...] [INFO ] 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-2023.4.0-SNAPSHOT.jar  
8888 8889
```

2. Check that the sample application is running in one of the following ways:
  - a. Go to <http://localhost:8081/home> to access the home page of the sample application. Information about the browser request is displayed.
  - b. 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.

## Stopping the sample application

1. 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:

1. Linux
2. Windows

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

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

```
{  
  "name" : "sampleapp-resources",  
  "baseURI" : "http://app.example.com:8081",
```

```
"condition": "${find(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:

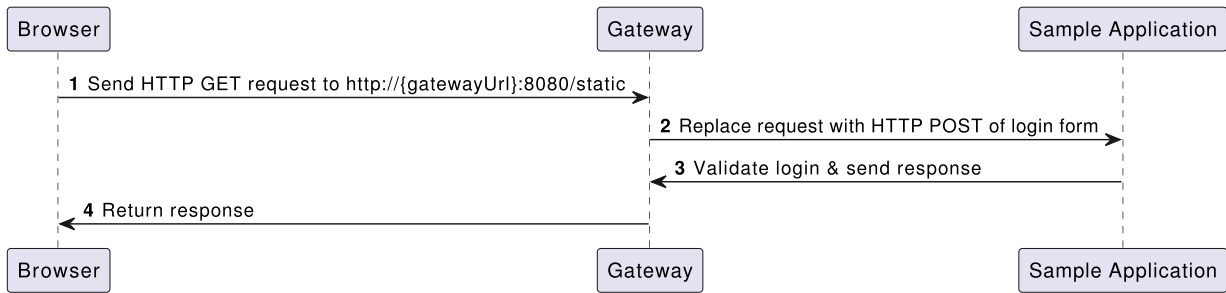
```
$ java -jar IG-sample-application-2023.4.0-SNAPSHOT.jar -h  
  
Usage: <main class> [options]  
Options:  
--http  
The HTTP port number.  
Default: 8081  
--https  
The HTTPS port number.  
Default: 8444  
--session  
The session timeout in seconds.  
Default: 60  
--am-discovery-url  
The AM URL base for OpenID Provider Configuration.  
Default: http://am.example.com:8088/openam/oauth2  
-h, --help  
Default: false
```

## 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, refer to 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:

### Log in with hard-coded credentials



1. The browser sends an HTTP GET request to the HTTP server on `ig.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, starting, and stopping IG](#), and the sample application as described in [Using the sample application](#).
2. Add the following route to serve static resources, such as `.css`, for the sample application:

1. Linux
2. Windows

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

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

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

3. Add the following route to IG:

1. Linux

## 2. Windows

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

```
%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": "${find(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://ig.example.com:8080/static>.

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

## 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, refer to the [Installation guide](#).

## Adding a base configuration file

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

1. Linux
2. Windows

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

```
%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](#). The default looks for routes in:

1. Linux
2. Windows

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



```
%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, refer to [GatewayHttpApplication\(config.json\)](#), [Heap objects](#), and [Router](#).

### *Add a base configuration for IG*

1. Add the following file to IG:

1. Linux
2. Windows

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

```
%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, refer to [JwtSession](#).

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.

1. Add the following route to IG:

1. Linux
2. Windows

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

```
%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 configuration to `admin.json`, and restart IG:

```
{  
  "mode": "DEVELOPMENT",
```

```
"connectors": [  
  { "port" : 8080 }  
]  
}
```

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 an instance of IG in development mode:

1. Linux
2. Windows

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

```
C:\IG_RUN_MODE=development  
C:\path\to\identity-gateway\bin\start.bat %appdata%\OpenIG
```

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 an instance of IG with the system property `ig.run.mode` to force development mode:

1. Linux
2. Windows

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

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

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

For information about restricting access to Studio in development mode, refer to [Restricting access to Studio](#). For information about switching back to production mode, refer to [Switching from development mode to production mode](#).

## Using IG Studio

IG Studio is a user interface to help you build and deploy your IG configuration. For information about using Studio, refer to the [Studio guide](#).

---

Copyright © 2010-2023 ForgeRock, all rights reserved.