


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, see 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 that your installation meets the requirements in [Requirements](#).

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:

Linux	Windows
--------------	----------------

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

Linux

Windows

`/etc/hosts`

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

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

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 in standalone mode

Download the IG .zip file

1. Create a local installation directory for IG. The examples in this section use `/path/to`.
2. Download `IG-7.2.0.zip` from the [ForgeRock BackStage download site](#), and copy the `.zip` file to the installation directory:

```
$ cp IG-7.2.0.zip /path/to/IG-7.2.0.zip
```

3. Unzip the file:

```
$ unzip IG-7.2.0.zip
```

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

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:

Linux

Windows

```
$ /path/to/identity-gateway/bin/start.sh  
  
...  
... started in 1234ms on ports : [8080 8443]
```

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 the instance directory:

Linux

Windows

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

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, see [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:

Linux

Windows

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

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.

1. Download a supported version of Tomcat server from its [download page](#), and install it to `/path/to/tomcat`.

2. Remove the `R00T` directory in Tomcat:

```
$ rm -rf /path/to/tomcat/webapps/R00T
```

3. Download `IG-7.2.0.war` from the [ForgeRock BackStage download site](#).

4. Copy the `IG-7.2.0.war` to the Tomcat `webapps` directory, as `R00T.war`:

```
$ cp IG-7.2.0.war /path/to/tomcat/webapps/R00T.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://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>.

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.

1. Download a supported version of Jetty server from its [download page](#), and install it to `/path/to/jetty`.
2. Download `IG-7.2.0.war` from the [ForgeRock BackStage download site](#).
3. Copy the `.war` file:

```
$ cp IG-7.2.0.war /path/to/jetty/webapps/IG-7.2.0.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://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>.

Downloading and starting IG in JBoss EAP

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

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-7.2.0.war from the [ForgeRock BackStage download site](#).

4. Copy the IG-7.2.0.war to the JBoss deployment directory:

```
$ cp IG-7.2.0.war
/path/to/jboss/standalone/deployments/IG-7.2.0.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://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>.

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-7.2.0.jar, from the [ForgeRock BackStage download site](#).

Starting the sample application

1. Start the sample application:

```
$ java -jar IG-sample-application-7.2.0.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-7.2.0.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:

Linux

Windows

```
$HOME/.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-7.2.0.jar -h

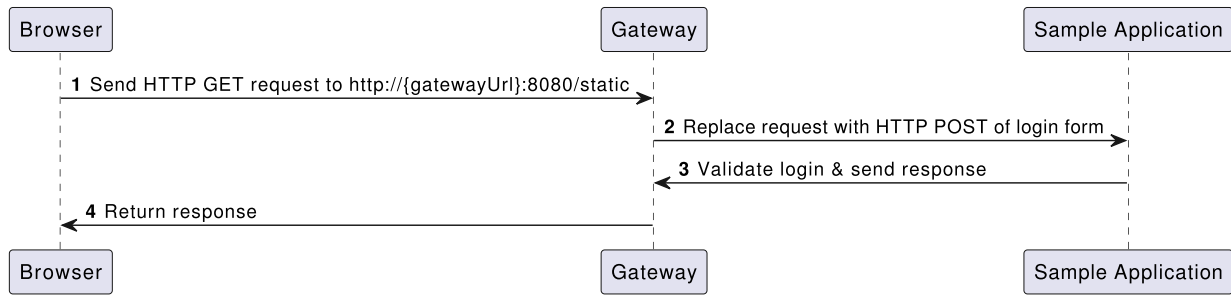
Usage: <main class> [options]
Options:
--http
The HTTP port number.
Default: 8081
--https
The HTTPS port number.
Default: 8444
--session
The session timeout in seconds.
Default: 60
--am-discovery-url
The AM URL base for OpenID Provider Configuration.
Default: http://am.example.com:8088/openam/oauth2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

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:

Linux

Windows

```
$HOME/.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:

Linux

Windows

```
$HOME/.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, see 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:

Linux

Windows

```
$HOME/.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:

Linux

Windows

```
$HOME/.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\(config.json\)](#), [Heap objects](#), and [Router](#).

Add a Base Configuration for IG

1. Add the following file to IG:

Linux

Windows

```
$HOME/.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](#).

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:

Linux

Windows

```
$HOME/.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	Windows
<pre>\$HOME/.openig/config/admin.json</pre>	

Standalone mode	Web container mode
<pre>{ "mode": "DEVELOPMENT", "connectors": [{ "port" : 8080 }] }</pre>	

For more information, see [AdminHttpApplication \(admin.json\)](#).

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:

Linux	Windows
<pre>\$ IG_RUN_MODE=development /path/to/identity-gateway/bin/start.sh</pre>	

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	Windows
<pre>\$HOME/.openig/env.sh</pre>	

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

For information about restricting access to Studio in development mode, see [Restricting access to Studio](#). For information about switching back to production mode, see [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, see the [Studio guide](#).

Was this helpful?  