Release Notes

OpenIG 4

Mark Craig

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

Abstract

Notes covering OpenIG prerequisites, fixes, known issues. OpenIG provides a high-performance reverse proxy server with specialized session management and credential replay functionality.

Table of Contents

Chapter 1
What's New in OpenIG

OpenIG 4 provides many new features and improvements.

1.1. New Features

This release of OpenIG includes the following new capabilities:

Policy Enforcement Point

OpenIG now provides a policy enforcement filter for use with OpenAM as a policy decision point (OPENIG-435).

With this feature, OpenIG can be used instead of an OpenAM agent for both authentication and authorization. This allows you to centralize all your access control policies in OpenAM for applications and APIs.

For more information and a tutorial, see Chapter 6, "*OpenIG As an OpenAM Policy Enforcement Point*" in the *Gateway Guide*.

OpenID Connect Discovery

OpenIG now supports OpenID Connect dynamic client registration and discovery (OPENIG-463, OPENIG-522).

This feature improves the user experience when using OpenID Connect by simplifying the identity provider selection. It also reduces the need for administrators to register OpenIG in advance with all identity providers.

For more information and a tutorial, see Section 9.7, "Using OpenID Connect Discovery and Dynamic Client Registration" in the *Gateway Guide*.

OpenIG now also supports RFC 7591, *OAuth 2.0 Dynamic Client Registration Protocol*.

Better Integration With OpenAM

OpenIG provides the following features for improved integration with OpenAM:

- A token transformation filter to use with the OpenAM REST Security Token Service (STS) to transform an OpenID Connect ID token into a SAML 2.0 assertion (OPENIG-430).

This feature enables and extends SSO and federation for applications, especially mobile apps. For example, a mobile app that has an OpenID Connect token can access resources held by a federated service provider, thanks to the SAML token obtained by OpenIG.

For details, see TokenTransformationFilter(5) in the *Configuration Reference*.

- A password replay filter that simplifies the configuration required to implement replaying credentials automatically by wrapping common use cases in a single filter with few parameters (OPENIG-641).

  For details, see PasswordReplayFilter(5) in the *Configuration Reference*. Several examples in the documentation take advantage of this new feature.

- An OpenAM SSO filter used internally for simplified configuration and interaction with OpenAM (OPENIG-694).

  OpenIG continues to work with any standards-based identity provider, and can now more effectively manage OpenAM SSO tokens.

UMA Resource Server Filter

OpenIG now provides experimental support for building a User-Managed Access (UMA) resource server (OPENIG-433).

This feature makes it possible to enable APIs and applications to use UMA, and to protect resources with OpenAM an UMA Authorization server.

For more information and a tutorial, see Chapter 10, "*OpenIG As an UMA Resource Server*" in the *Gateway Guide*.

Auditing, Monitoring, and Throttling

OpenIG now provides several new features related to auditing, monitoring, and throttling access to APIs and protected applications:

- Integration with ForgeRock's common audit framework, which supports logging to files, databases, and the UNIX system log (Syslog) (OPENIG-495).

  ForgeRock common audit framework allows you to handle audit events in a common way across the ForgeRock platform, to centralize audit logs, and to trace transactions through the platform. Log files can be signed to make tampering evident.

  For more information, see Section 14.3, "Audit Events and Logging" in the *Gateway Guide*.

- Improved monitoring for the server and for access to protected applications and APIs (OPENIG-431).

  This feature allows you to build a better view of how OpenIG and its routes are used, so you can take preemptive administrative action and achieve the required quality of service.

For more information, see Section 14.2, "Monitoring a Route" in the *Gateway Guide*.

- Throttling to limit access to protected applications and APIs (OPENIG-532).

  This feature increases security and fairness in the use of protected APIs and applications. The throttling filter can enforce flexible rate limits for a variety of use cases.

  For more information, see Section 14.1, "Limiting Access With a Throttling Filter" in the *Gateway Guide*.

Non-Blocking HTTP Client Requests

OpenIG now has improved support for asynchronous processing, including asynchronous HTTP client access to external services (OPENIG-513, OPENIG-639).

This feature provides greater scalability with lower resource consumption. OpenIG uses connection pools for connections to protected applications, enabling each server to handle much more traffic than before.

Method Invocation in Expressions

OpenIG now supports Java method invocation in expressions, providing a richer way of building the configuration parameters, allowing string extraction, substrings, and joins (OPENIG-584).

1.2. Product Improvements

This release of OpenIG includes the following enhancements:

Configurable Transport Layer Security

- Configuration of the `SSLContext` algorithm to any algorithm supported by the Java virtual machine (OPENIG-590).
- Options to restrict the list of acceptable TLS and SSL protocols and cipher suites when negotiating an HTTPS connection (OPENIG-749).

For more information, see the settings described in ClientHandler(5) in the *Configuration Reference*.

Convenient Trust Manager for Testing

A `TrustAllManager` for use in testing that blindly trusts all server certificates (OPENIG-516).

For details, see TrustAllManager(5) in the *Configuration Reference*.

Expiration for JWT Session Cookies

Expiration time for JWT session cookies employed by the `JwtSession` implementation (OPENIG-733).

For details, see the `sessionTimeout` property described in JwtSession(5) in the *Configuration Reference*.

Plugins for Expression Evaluation

Plugins to extend configuration expressions (OPENIG-422).

For details, see Section 13.7, "Key Extension Points" in the *Gateway Guide*.

Simplified Base URI Management

A base URI decorator with a default of `baseURI` that replaces the configuration field of the same name in `Route` and `GatewayHttpApplication` definitions (OPENIG-180).

For details, see BaseUriDecorator(5) in the *Configuration Reference*.

Enhancements for Scripting

The capability for `ScriptableFilter` and `ScriptableHandler` arguments (`args`) to reference `heap` objects (OPENIG-332).

For details, see ScriptableFilter(5) in the *Configuration Reference*, and ScriptableHandler(5) in the *Configuration Reference*.

OpenIG now also precompiles Groovy scripts making it possible to identify problems when the script first loads, rather than delaying until the script runs (OPENIG-660).

In addition, OpenIG now supports dependency management with Grape in Groovy scripts (OPENIG-540).

Grape let Groovy scripts use `@Grab` and related annotations to specify dependencies on external .jar files. For details, see the Groovy documentation on *Dependency management with Grape*.

Chapter 2

Before You Install OpenIG Software

This chapter covers requirements for running OpenIG software.

Tip

If you have a special request to support a component or combination not listed here, contact ForgeRock at info@forgerock.com.

2.1. JDK Version

This release of OpenIG requires Java Development Kit 7 or 8. ForgeRock recommends the most recent update to ensure you have the latest security fixes.

If you install an OpenAM policy agent in the same container as OpenIG, then you must use a Java release that is supported with the policy agent as well.

2.2. Web Application Containers

OpenIG runs in the following web application containers:

- Apache Tomcat 7 or 8
- Jetty 8 (8.1.13 or later) or 9

You must deploy OpenIG to the root context of the container. Deployment in other context causes unexpected results, and cannot be supported.

OpenIG requires Servlet 3.0 or later.

For details on setting up your web application container see Section 3.1, "Configuring Deployment Containers" in the *Gateway Guide*.

2.3. OpenAM Features

When using OpenIG with OpenAM, the following features are supported with OpenAM 13 or later:

- OpenAM policy enforcement, as described in Chapter 6, "*OpenIG As an OpenAM Policy Enforcement Point*" in the *Gateway Guide*
- OpenID Connect dynamic registration and discovery, as described in Section 9.7, "Using OpenID Connect Discovery and Dynamic Client Registration" in the *Gateway Guide*
- User Managed Access, as described in Chapter 10, "*OpenIG As an UMA Resource Server*" in the *Gateway Guide*

2.4. OpenAM Policy Agents

When installing an OpenAM policy agent in the same container as OpenIG, use an OpenAM Java EE policy agent version 3.5 or later. Earlier versions of OpenAM policy agents might not shut down properly with the web application container (OPENIG-258).

Make sure that the container version is supported both for OpenIG and for the OpenAM Java EE policy agent that you install alongside OpenIG.

Chapter 3

OpenIG Compatibility

This chapter covers both major changes to existing functionality, and also deprecated and removed functionality.

3.1. Important Changes to Existing Functionality

This release brings the following important changes to OpenIG:

- All paths starting with `/openig` are now reserved for use by OpenIG for administrative use, and can no longer be matched by other route conditions.

  Resources exposed under `/openig` are only accessible to local client applications.

- OpenIG no longer provides an exchange object to model the HTTP exchange. The exchange object model has been replaced by a new model based on requests, responses, and contexts. Table 3.1, "Comparison Between Object Models" summarizes changes that affect configuration expressions and scripts.

Table 3.1. Comparison Between Object Models

Previous Model	Current Model
`exchange`	Removed. Arbitrary properties must move to `attributes`.
`exchange.clientInfo`	`contexts.client`
`exchange.originalUri`	`contexts.router.originalUri`
`exchange.principal`	Use `contexts.client.remoteUser` instead.
`exchange.request`	`request`
`exchange.response`	`response`
`exchange.response.reason`	`response.status.reasonPhrase`
`exchange.response.status`	`response.status` returns a `Status` object. See Status(5) in the *Configuration Reference*.
`exchange.session`	`session`

- As the response status is now represented by a `Status` object, the expression `${response.status}` resolves to a status object rather than a status code. To get the response status code as an integer, use `${response.status.code}`.

  In scripts, add `import org.forgerock.http.protocol.Status` and then use `Status` objects as in the following example:

  ```
  import org.forgerock.http.protocol.Status

  response.status = Status.OK            // 200
  response.status = Status.FORBIDDEN     // 403
  ```

  For details, see Status(5) in the *Configuration Reference*.

- The exchange allowed arbitrary properties at the base level. The new model includes an attributes context instead. Add arbitrary properties to the map named `attributes`.

  For example, if an existing configuration targets an expression that defines a base-level property in the exchange, such as `${exchange.token}`, edit the configuration to make the property one of the attributes instead, as in `${attributes.token}`.

- A change in the way JWTs are represented has the effect that `JwtSession` cookies encrypted by earlier versions of OpenIG cannot be decrypted.

  After upgrade, OpenIG must renew users' `JwtSession` cookies. This is reflected with messages in the log, such as the following:

  - `The JWT Session Cookie 'openig-jwt-session' could not be decrypted.`
  - `Cannot rebuild JWT Session from Cookie 'openig-jwt-session'`

- Previously, when a router or dispatcher could not find a route or handler for a request, it threw a handler exception, resulting in an HTTP 500 Server Error message.

  OpenIG now generates a response instead, resulting in a proper HTTP 404 Not Found message.

- The following changes affect Groovy scripts called from `ScriptableFilter` and `ScriptableHandler` objects:

  - Groovy scripts must now return a `Promise<Response, NeverThrowsException>` or a `Response`. Any other return type, including `null`, yields an HTTP 500 Server Error response.

    Note that in the Groovy language, methods always return a value. If no `return` statement is provided, the value evaluated in the last line is returned.

    Also, to return a `Promise` based on an existing response, use `Response.newResponsePromise(response)`.

  - The global objects passed to Groovy scripts now include `context` and `request`.

Edit your scripts to use `request` instead of `exchange.request`.

- The `http` global object passed to Groovy scripts is now an org.forgerock.http.Client. A `Client` has a `send()` method for sending a request that returns a `Promise` representing the pending HTTP response.

  The script must then use the `Promise` methods to deal with the response, working either with asynchronous callbacks or with the `Promise` `get()` methods for synchronous responses.

- The `Handler` interface has changed, affecting the `next.handle()` method.

  Edit your scripts to use `next.handle(context, request)` instead of `next.handle(exchange)`.

- In Groovy scripts, raw access to header values must now be prefixed with `.values` or `?.values` if the header might not exist. For example, `headers.Username[0]` must replaced with `headers.Username?.values[0]`. Similarly, `headers['Username'][0]` must be replaced with `headers['Username']?.values[0]`.

- Consumers of audit events now access source, tags, and timestamps through a map called `event` rather than through the `exchange`. Request, response, and context are available through `event.data`.

  For example, to access the request URI, use `${event.data.request.uri}`. To access the response headers, use `${event.data.response.headers}`.

- Arguments (`args`) passed to scripts must no longer override other global objects passed to scripts. Attempts to reuse the name of another global object now cause the script to fail and OpenIG to return a response with HTTP status code 500 Internal Server Error.

- The examples in Chapter 13, "*Extending OpenIG's Functionality*" in the *Gateway Guide* reflect these changes.

- The `Route` and `GatewayHttpApplication` configurations now use `baseURI` decorators instead of `baseURI` configuration fields.

- This release introduces independent `Issuer` and `ClientRegistration` configuration objects. An `Issuer` represents an OAuth 2.0 authorization server or OpenID Provider. A `ClientRegistration` represents the client application registration with an `Issuer`. The `OAuth2ClientFilter` configuration has changed to work with the new configuration objects. For details, see Table 3.3, "Deprecated Configuration Settings".

  Previous configurations cause errors and prevent the route from loading when OpenIG reads the configuration:

```
------------------------------
THU SEP 10 17:37:04 CEST 2015 (ERROR) {Router}/handler
The route defined in file '/home/user/.openig/config/routes/07-openid.json'
 cannot be added
------------------------------
THU SEP 10 17:37:04 CEST 2015 (ERROR) {Router}/handler
/handler/config/filters/0/config/loginHandler: Expecting a value
[       JsonValueException] > /handler/config/filters/0/config/loginHandler:
 Expecting a value
```

The new configuration object calls for HTTP Basic authentication by default when connecting to the provider's OAuth 2.0 token endpoint. The previous implementation called for client credentials to be sent as HTTP POST form data. If necessary you can set `tokenEndpointUseBasicAuth` to false in the client registration configuration to send client credentials as HTTP POST form data.

- It is no longer possible to set `openig-base` in the .war file. This was set as a Servlet `<init-param>`:

```
<init-param>
  <param-name>openig-base</param-name>
  <param-value>/path/to/openig</param-value>
</init-param>
```

Set the value as a system property or environment variable instead. For details, see Section 3.3, "Installing OpenIG" in the *Gateway Guide*.

- The classes mentioned in Table 3.2, "Class Changes" have changed names or changed packages.

Table 3.2. Class Changes

Former Name	Current Name
`org.forgerock.openig.jwt.JwtSessionFactory`	`org.forgerock.openig.jwt.JwtSessionManager`
`org.forgerock.openig.http.ClientInfo`	`org.forgerock.services.context.ClientContext`
`org.forgerock.openig.http.Request`	`org.forgerock.http.protocol.Request`
`org.forgerock.openig.http.Response`	`org.forgerock.http.protocol.Response`
`org.forgerock.openig.servlet.GatewayServlet`	`org.forgerock.openig.http.GatewayHttpApplication`
`org.forgerock.openig.util.MutableUri`	`org.forgerock.http.MutableUri`

3.2. Deprecated Functionality

This section lists deprecated functionality. Deprecation is defined in Section A.2, "ForgeRock Product Interface Stability" in the *Configuration Reference*.

Table 3.3. Deprecated Configuration Settings

Configuration Object	Deprecated Settings	Newer Evolving Settings
`AuditDecorator`	Entire object	Use a `monitor` attribute on a route instead. See Section 14.2, "Monitoring a Route" in the *Gateway Guide*.
`CaptureDecorator`	`captureExchange`	New name: `captureContext`
`GatewayHttpApplication`	`handlerObject`	New name: `handler`
	Deprecated format: `"heap": { "objects": [ configuration object, ... ] }`	New format: `"heap": [ configuration object, ... ]`
`MonitorEndpointHandler`	Entire object	Use monitoring on routes instead. See Section 14.2, "Monitoring a Route" in the *Gateway Guide*.
`OAuth2ClientFilter`	`loginHandler`	For multiple registrations, continue to use `loginHandler`. For single registrations, use `registration`.
	`providerHandler`	N/A
	`providers`	Replaced by separate `Issuer` and `ClientRegistration` configuration objects
	`redirect_uris`	List the redirect URIs in the dynamic client registration `metadata`.
	`scopes`	List the scopes in the dynamic client registration `metadata`.
`RedirectFilter`	Entire object	Use `LocationHeaderFilter` instead.
`OAuth2ResourceServerFilter`	`enforceHttps`	New name: `requireHttps`
	`httpHandler`	New name: `providerHandler`
	`requiredScopes`	New name: `scopes`
`RedirectFilter`	Entire object	Use `LocationHeaderFilter` instead.
`Route`	Deprecated format: `"heap": { "objects": [ configuration object, ... ] }`	New format: `"heap": [ configuration object, ... ]`

For details on the new and updated configuration objects, see Client(5) in the *Configuration Reference*, ClientHandler(5) in the *Configuration Reference*, ClientRegistration(5) in the *Configuration Reference*, GatewayHttpApplication(5) in the *Configuration Reference*, Issuer(5) in the *Configuration Reference*, KeyManager(5) in the *Configuration Reference*, LocationHeaderFilter(5) in the *Configuration Reference*, OAuth2ClientFilter(5) in the *Configuration Reference*, OAuth2ResourceServerFilter(5) in the *Configuration Reference*, Route(5) in the *Configuration*

Reference, Status(5) in the *Configuration Reference*, and TrustManager(5) in the *Configuration Reference*.

The following class is likely to be removed in a future release:

- org.forgerock.openig.heap.NestedHeaplet

  The interface to extend instead is org.forgerock.openig.heap.GenericHeaplet.

The following methods for dealing with form strings are deprecated:

`Form.fromString(String s)`

Use `fromFormString(String s)` instead.

`Form.toString(String )`

Use `toFormString()` instead.

3.3. Removed Functionality

This section lists functionality that has been removed:

- Support for Java 6
- The `CaptureFilter` implementation

  Use a `CaptureDecorator` instead, as described in CaptureDecorator(5) in the *Configuration Reference*.

- The `GatewayServlet` implementation

  The implementation has been replaced with `org.forgerock.http.servlet.HttpFrameworkServlet`, which wraps an `HttpApplication` that is provided by `org.forgerock.openig.http.GatewayHttpApplication`. The `GatewayHttpApplication` creates the heap and builds the configuration.

- The `HttpClient` configuration object

  Its configuration is now part of the `ClientHandler` configuration, described in ClientHandler(5) in the *Configuration Reference*.

  The `hostnameVerifier` setting `BROWSER_COMPATIBLE` has been removed. Consider using `STRICT` instead.

  The `keystore` and `truststore` settings, which are deprecated since the release of 3.1, have been removed. Use `keyManager` and `trustManager` instead.

- The client connection information implementation, `org.forgerock.openig.http.ClientInfo`

  The implementation has been replaced with `org.forgerock.services.context.ClientContext`.

- The `StaticRequestFilter` configuration setting `restore`, used to restore the request in the exchange, has been removed.

The following API interfaces and classes have been removed:

- `org.forgerock.openig.filter.Filter`

  The new interface to implement is org.forgerock.http.Filter.

- `org.forgerock.openig.handler.Handler`

  The new interface to implement is org.forgerock.http.Handler.

- `org.forgerock.openig.handler.HandlerException`

Chapter 4

Fixes, Limitations, and Known Issues

OpenIG issues are tracked at https://bugster.forgerock.org/jira/browse/OPENIG. This chapter covers the status of key issues and limitations at release 4.

4.1. Key Fixes

The following important issues were fixed in this release:

- OPENIG-647: SSL and JDK1.6 - handshake failures
- OPENIG-503: Fix resource leak on route loading
- OPENIG-491: Using groovy script embedded in json route doesn't work on windows
- OPENIG-470: Connections are not released after modifying HttpClient connections pool size
- OPENIG-454: Capture decorator impacts the entity returned in GET
- OPENIG-426: Multiple Host header
- OPENIG-315: POST JSON payload not delivered unless CaptureFilter used
- OPENIG-290: Null pointer exception when capturing SAML federation response

4.2. Limitations

The following limitations are present in this release:

- For HTTPS, OpenIG can check server certificates. However, mutual authentication, where OpenIG presents its client certificate, is not supported if the client certificate is not the first certificate in the `ClientHandler` keystore.
- OpenIG scripts are not sandboxed, but instead have access to anything in their environment. You must make sure that the scripts that OpenIG loads are safe.
- The `Issuer` field, `supportedDomains`, only causes OpenIG to use `ClientRegistration` settings if the `ClientRegistration` name is set to the concatenation of the `Issuer` name and the `OAuth2ClientFilter` name. For example, if the `Issuer` name is `openam` and the `OAuth2ClientFilter` name is `OAuth2Client` then the `ClientRegistration` name must be set to `openamOAuth2Client`.

- The `SamlFederationHandler` does not support filtering. Do not use a `SamlFederationHandler` as the handler for a `Chain`.

  More generally, do not use this handler when its use depends on something in the response. The response can be handled independently of OpenIG, and can be `null` when control returns to OpenIG. For example, do not use this handler in a `SequenceHandler` where the `postcondition` depends on the response.

4.3. Known Issues

The following known issues remained open at the time of release:

- OPENIG-816: The UmaResourceServerFilter returns with wrong as_uri
- OPENIG-813: auditService : fileRotation may overwrite existing audit file
- OPENIG-712: Issuer definition : supportedDomains doesn't lead to use of static clientRegistration
- OPENIG-478: assertionMapping doesn't support multi-valued attribute
- OPENIG-466: No way to add realm parameter to tokenInfoEndPoint
- OPENIG-458: CookieFilter is not JwtSession compatible
- OPENIG-322: Cannot access both an OpenAM (self-signed) and a Google HTTPS endpoint
- OPENIG-291: Class cast exception when using SAML federation & policy agent together
- OPENIG-234: Federation doesn't work if we used incomplete user in IDP
- OPENIG-221: Cannot specify which certificate to present to server if server requires mutual authentication in https

Chapter 5

How to Report Problems and Provide Feedback

If you have questions regarding OpenIG that are not answered by the documentation, there is a mailing list which can be found at https://lists.forgerock.org/mailman/listinfo/openig where you are likely to find an answer.

If you have found issues or reproducible bugs within OpenIG, report them in https://bugster.forgerock.org.

When requesting help with a problem, include the following information:

- Description of the problem, including when the problem occurs and its impact on your operation
- Machine type, operating system version, web container and version, Java version, and OpenIG release version, including any patches or other software that might be affecting the problem
- Steps to reproduce the problem
- Any relevant logs or stack traces

Chapter 6
Support

You can purchase OpenIG support, subscriptions and training courses from ForgeRock and from consulting partners around the world and in your area. To contact ForgeRock, send mail to info@forgerock.com. To find a partner in your area, use the ForgeRock website.