

Paul Bryan Mark Craig Jamie Nelson

ForgeRock AS 33 New Montgomery St., Suite 1500 San Francisco, CA 94105, USA +1 415-523-0772 www.forgerock.com

Copyright © 2011-2014 ForgeRock AS

Abstract

Reference documentation for OpenIG. OpenIG provides a high-performance reverse proxy server with specialized session management and credential replay functionality.



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock™ is the trademark of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

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

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

DeiaVu Font

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABLITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDING AND GENERAL, SPECIAL, INDIRECT, INCIDING AND GENERAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

A.... Franks Commission

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABLITY, INCLUDING ANY GENERAL, SPECIAL, INDIRENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tawnjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tawnjong Bah. For further information, contact: tawnjong and give e. fr.

Table of Contents

Preface	
1. Who Should Use this Reference	v
2. Formatting Conventions	v
3. Accessing Documentation Online	
4. Joining the ForgeRock Community	vii
I. Required Configuration	
Gateway Servlet	
Heap Objects	
II. Handlers	
Chain	
ClientHandler	
DispatchHandler	
Route	
Router	
ScriptableHandler	
SequenceHandler	29
StaticResponseHandler	
III. Filters	
AssignmentFilter	
CaptureFilter	
CookieFilter	
EntityExtractFilter	
ExceptionFilter	
FileAttributesFilter	
HeaderFilter	
CryptoHeaderFilter	
HttpBasicAuthFilter	
OAuth2ClientFilter	
OAuth2ResourceServerFilter	
RedirectFilter	
ScriptableFilter	
SqlAttributesFilter	71
StaticRequestFilter	
SwitchFilter	
IV. Miscellaneous Heap Objects	
ConsoleLogSink	
HttpClient	81
TemporaryStorage	
V. Expressions	
Expressions	
Functions	
Patterns	
VI. Exchange Object Model	107

OpenIG Reference

Exchange	109
Request	
Principal	113
Response	115
URÍ	
A. Release Levels & Interface Stability	119
A.1. ForgeRock Product Release Levels	119
A.2. ForgeRock Product Interface Stability	120
Index	121

Preface

This reference covers OpenIG configuration.

1 Who Should Use this Reference

This references is written for access management designers, developers, and administrators using OpenIG. For API specifications, see the appropriate Javadoc.

2 Formatting Conventions

Most examples in the documentation are created on GNU/Linux or Mac OS X. Where it is helpful to make a distinction between operating environments, examples for UNIX, GNU/Linux, Mac OS X, and so forth are labeled (UNIX). Mac OS X specific examples can be labeled (Mac OS X). Examples for Microsoft Windows can be labeled (Windows). To avoid repetition, however, file system directory names are often given only in UNIX format as in /path/to/server, even if the text applies to C:\path\to\server as well.

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

Command line, terminal sessions are formatted as follows.

\$ echo \$JAVA_HOME
/path/to/jdk

V

Accessing Documentation Online

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. In the following example, the query string parameter prettyPrint=true is omitted.

```
$ curl https://bjensen:hifalutin@opendj.example.com:8443/users/newuser
 "_rev" : "000000005b337348",
 "schemas" : [ "urn:scim:schemas:core:1.0" ],
  "contactInformation" : {
   "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  " id" : "newuser",
  "name" : {
   "familyName" : "New",
    "givenName" : "User"
  "userName" : "newuser@example.com",
  "displayName" : "New User",
  "meta" : {
    "created" : "2014-06-03T09:58:27Z"
  "manager" : [ {
    "_id" : "kvaughan",
    "displayName" : "Kirsten Vaughan"
```

Program listings are formatted as follows.

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

3 Accessing Documentation Online

ForgeRock core documentation, such as what you are now reading, aims to be technically accurate and complete with respect to the software documented.

Core documentation therefore follows a three-phase review process designed to eliminate errors.

- Product managers and software architects review project documentation design with respect to the users' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.

Joining the ForgeRock Community

• Quality experts validate implemented documentation changes for technical validity with respect to the software, technical completeness with respect to the scope of the document, and usability for the expected audience.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at http://docs.forgerock.org/. Use this documentation when working with a ForgeRock Enterprise release.

In-progress documentation can be found at each project site under the Developer Community projects page. Use this documentation when trying a nightly build.

The ForgeRock Community Wikis and provide additional, user-created information. We encourage you to join the community, so that you can update the Wikis, too.

4 Joining the ForgeRock Community

After you sign up to join the ForgeRock community, you can edit the Community Wikis, and also log bugs and feature requests in the issue tracker.

If you have a question regarding a project but cannot find an answer in the project documentation or Wiki, browse to the Developer Community page for the project, where you can find details on joining the project mailing lists, and find links to mailing list archives. You can also suggest updates to documentation through the ForgeRock docs mailing list.

The Community Wikis describe how to check out and build source code. Should you want to contribute a patch, test, or feature, or want to author part of the core documentation, first have a look on the ForgeRock site at how to get involved.

Required Configuration

You must specify at least the entry point for incoming requests, the OpenIG Servlet, and the heap objects that configure and initialize objects, with dependency injection.

Table of Contents

Gateway Servlet	11
Heap Objects	13

Gateway Servlet

Gateway Servlet — entry point for all incoming requests

1 Description

The gateway servlet is the entry point for all incoming requests. It is responsible for initializing a heap of objects, and dispatching all requests to a configured handler. The configuration of the gateway servlet is loaded from a JSON-encoded configuration file, expected by default at \$HOME/.openig/config.json.

The gateway servlet also creates a default HttpClient object to handle requests to servers. You can override the default settings by configuring your own HttpClient.

2 Usage

```
{
    "heap": { heap-configuration object },
    "handlerObject": string,
    "baseURI": string,
    "logSink": string,
    "temporaryStorage": string
}
```

3 Properties

"heap": *object, required*

The heap object configuration.

"handlerObject": string, required

The name of the handler heap object to dispatch all requests to.

"baseURI": *string*, *optional*

Overrides the existing request URI, making requests relative to a new base URI.

Only scheme, host and port are used in the supplied URI.

Default: leave the request URI untouched.

"logSink": *string*, *optional*

The name of the LogSink heap object to which to send log messages.

Default: use the heap object named "LogSink". Otherwise use an internally-created ConsoleLogSink object that is named "LogSink" and that uses default settings.

"temporaryStorage": string, optional

The name of the TemporaryStorage heap object where content is cached during processing.

Default: use the heap object named "TemporaryStorage". Otherwise use an internally-created TemporaryStorage object that is named "TemporaryStorage" and that uses default settings.

4 Javadoc

org. forgerock. openig. servlet. Gateway Servlet

Heap Objects

Heap Objects — configure and initialize objects, with dependency injection

1 Description

A heap is a collection of associated objects, initialized from declarative configuration artifacts. All configurable objects in OpenIG are heap objects. Heap objects are created and initialized by associated "heaplets", which retrieve any objects an object depends on from the heap. The heap configuration is included as an object in the gateway servlet configuration, and has the following format.

2 Usage

3 Properties

"name": *string*, *required*

The unique name to give the heap object in the heap. This name is used to resolve the heap object, for example when another heap object names a heap object dependency.

"type": string, required

The class name of the object to be created. To determine the type name, see the object's documentation in this reference.

"config": object, required

The configuration that is specific to the heap object being created.

4 Automatically Created Objects

When a heap is first created, it is automatically populated with some objects, without required configuration. An automatically created object can be overridden by creating a heap object with the same name.

"LogSink"

The default object to use for writing all audit and performance logging. Default: a ConsoleLogSink object with default values.

"TemporaryStorage"

The default object to use for managing temporary buffers. Default: a TemporaryStorage object with default values.

5 Implicit Properties

Every heap object has a set of implicit properties, which can be overridden on an object-by-object basis:

"logSink": string

Specifies the heap object that should be used for audit and performance logging. Default: "LogSink".

"temporaryStorage": string

Specifies the heap object that should be used for temporary buffer storage.

Default: "TemporaryStorage".

Handlers

Handler objects process an HTTP exchange request by producing an associated response.

Table of Contents

Chain	17
ClientHandler	19
DispatchHandler	21
Route	
Router	25
ScriptableHandler	27
SequenceHandler	29
StaticResponseHandler	31

Chain

Chain — dispatch exchange to ordered list of filters

1 Description

A chain is responsible for dispatching an exchange to an ordered list of filters, and finally a handler.

2 Usage

3 Properties

"filters": array of strings, required

The names of the filter heap objects to dispatch the exchange to, in order.

"handler": *string*, *required*

The name of the handler object to dispatch to once the exchange has traversed all of the specified filters.

4 Example

```
{
    "name": "LoginChain",
    "type": "Chain",
    "config": {
        "filters": [ "LoginFilter" ],
        "handler": "ClientHandler"
    }
}
```

5 Javadoc

org.forgerock.openig.filter.Chain

ClientHandler

ClientHandler — submit exchange requests to remote servers

1 Description

Submits exchange requests to remote servers.

2 Usage

```
{
    "name": string,
    "type": "ClientHandler",
    "config": {
        "httpClient": string
    }
}
```

3 Properties

"httpClient": string, optional

The name of the HttpClient configuration object to use when making HTTP requests.

Default: use the heap object named "HttpClient", which matches the name of the HttpClient object created by the gateway servlet by default with default settings

If you overload that name by configuring your own HttpClient object named "HttpClient", then use that heap object.

4 Example

The following object configures a ClientHandler named Client that uses the HTTP client configuration specified in the configuration object with name MyHttpClient.

```
{
  "name": "Client",
  "type": "ClientHandler",
  "config": {
    "httpClient": "MyHttpClient"
  }
}
```

5 Javadoc

org. forgerock. openig. handler. Client Handler

DispatchHandler

DispatchHandler — dispatch to one of a list of handlers

1 Description

Dispatches to one of a list of handlers. When an exchange is handled, each handler's condition is evaluated. If a condition expression yields true, then the exchange is dispatched to the associated handler with no further processing.

2 Usage

3 Properties

"bindings": array of objects, required

A list of bindings of conditions and associated handlers to dispatch to.

"condition": expression, optional

Condition to evaluate to determine if associated handler should be dispatched to. If omitted, then dispatch is unconditional.

"handler": *string*, *required*

The name of the handler heap object to dispatch to if the associated condition yields true.

"baseURI": *string*, *optional*

Overrides the existing request URI, making requests relative to a new base URI. Only scheme, host and port are used in the supplied URI. Default: leave URI untouched.

4 Example

The following sample is from a SAML 2.0 federation configuration. If the incoming URI starts with /saml, then OpenIG dispatches to a

"SamlFederationHandler". If the user name is not set in the exchange session, then the user has not authenticated with the SAML 2.0 Identity Provider, so OpenIG dispatches to a "SPInitiatedSSORedirectHandler" to initiate SAML 2.0 single sign-on from the Service Provider, which is OpenIG. All other requests go through a "LoginChain" handler.

```
"name": "DispatchHandler",
"type": "DispatchHandler",
"config": {
    "condition": "${matches(exchange.request.uri.path, '^/saml')}",
    "handler": "SamlFederationHandler"
    },
    {
        "condition": "${empty exchange.session.username}",
        "handler": "SPInitiatedSSORedirectHandler",
        "baseURI": "http://www.example.com:8081"
    },
    {
        "handler": "LoginChain",
        "baseURI": "http://www.example.com:8081"
    }
}
```

5 Javadoc

org.forgerock.openig.handler.DispatchHandler

Route

Route — Configuration for handling a specified Exchange condition

1 Description

In OpenIG, a route is represented by a separate JSON configuration file and that handles an Exchange when a specified condition is met.

A top-level Router is responsible for reloading the route configuration. Use a Router to call route handlers, rather than calling a route directly as the "handlerObject" of the gateway servlet. By default the Router rereads the configurations periodically, so that configuration changes to routes apply without restarting OpenIG.

Each separate route has its own Heap of configuration objects. The route's Heap inherits from its parent Heap, which is the global heap for top-level routes, so the route configuration can reference configuration objects specified in the top-level Router configuration file.

For examples of route configurations, see the Routing Tutorial.

2 Properties

"handler": string, required

Name of the main handler for this route.

"baseURI": *string*, *optional*

URI on which to rebase the request URL.

If this is set, the request URL is rebased using the value. Rebasing changes the scheme, host, and port of the request URL. Rebasing does not affect the path, query string, or fragment.

Default: the request URL remains unchanged.

"condition": *expression*, optional

Whether the route accepts to handle the Exchange.

Default: If the condition is not set, or is null, then this route accepts any Exchange.

"name": string, optional

Name for the route, used by the Router to order the routes.

Default: Route configuration file name

Router

Router — Route processing to distinct configurations

1 Description

A Router is a handler that routes Exchange processing to separate configuration files. Each separate configuration file then defines a Route. See the *Description* section of the Route reference for details.

The Router reloads configuration files for Routes from the specified directory at the specified scan interval.

2 Usage

```
{
   "name": "Router",
   "type": "Router",
   "config": {
       "directory": expression,
       "scanInterval": integer
   }
}
```

An alternative value for "type" is "RouterHandler".

3 Properties

"directory": expression, optional

Base directory from which to load configuration files for routes.

Default: default base directory for route configuration files. For details see the section, *Installing OpenIG*.

"scanInterval": integer, optional

Interval in seconds after which OpenIG scans the specified directory for changes to configuration files.

Default: 10 (seconds)

To prevent OpenIG from reloading Route configurations after you except at startup, set the scan interval to -1.

4 Javadoc

org.forgerock.openig.handler.router.RouterHandler

ScriptableHandler

ScriptableHandler — handle a request by using a script

1 Description

Handles a request by using a script.

The script has access to the following global objects.

exchange

The exchange provides access to the HTTP request and response.

The request is created and populated before calling the handler. The handler is responsible for creating and for populating the response in the exchange.

globals

This Map holds variables that persist across successive invocations.

http

The http object provides an embedded HTTP client.

Use this client to perform outbound HTTP requests.

ldap

The ldap object provides an embedded LDAP client.

Use this client to perform outbound LDAP requests, such as LDAP authentication.

logger

The logger object provides access to the server log sink.

2 Usage

```
{
   "name": string,
   "type": "ScriptableHandler",
   "config": {
      "type": string,
      "file": string, // Use either "file"
      "source": string // or "source", but not both.
}
}
```

3 Properties

"type": string, required

The Internet media type (formerly MIME type) of the script, either "application/x-groovy" for Groovy or "text/javascript" for JavaScript

"file": string

Path to the file containing the script; mutually exclusive with "source"

Relative paths in the "file" field are relative to the base location for scripts. The base location depends on the configuration. For details see the section, *Installing OpenIG*.

The base location for Groovy scripts is on the classpath when the scripts are executed. If therefore some Groovy scripts are not in the default package, but instead have their own package names, they belong in the directory corresponding to their package name. For example, a script in package com.example.groovy belongs under <code>openig-base/scripts/groovy/com/example/groovy/.</code>

"source": string

The script as a string; mutually exclusive with "file"

4 Javadoc

org.forgerock.openig.handler.ScriptableHandler

SequenceHandler

SequenceHandler — process exchange through sequence of handlers

1 Description

Processes an exchange through a sequence of handlers. This allows multi-request processing such as retrieving a form, extracting form content (for example, nonce) and submitting in a subsequent request. Each handler in the bindings is dispatched to in order; the binding postcondition determines if the sequence should continue.

2 Usage

3 Properties

"bindings": array of objects, required

A list of bindings of handler and postcondition to determine that sequence continues.

"handler": string, required

The name of the handler heap object to dispatch to.

"postcondition": *expression*, optional

Evaluated to determine if the sequence continues. Default: unconditional.

4 Javadoc

org.forgerock.openig.handler.SequenceHandler

StaticResponseHandler

StaticResponseHandler — create static response in HTTP exchange

1 Description

Creates a static response in an HTTP exchange.

2 Usage

3 Properties

"status": number, required

The response status code (for example, 200).

"reason": string, optional

The response status reason (for example, "OK").

"version": string, optional

Protocol version. Default: "HTTP/1.1".

"headers": array of objects, optional

Header fields to set in the response. The name specifies the header name, with an associated array of expressions to evaluate as values.

"entity": *expression*, optional

The message entity expression to be evaluated and included in the response.

Conforms to the Content-Type header and sets Content-Length.

4 Example

```
{
"name": "ErrorHandler",
"type":"StaticResponseHandler",
```

```
"config": {
    "status": 500,
    "reason": "Error",
    "entity": "<html>
    <h2>Epic #FAIL</h2>
    </html>"
}
```

5 Javadoc

org. forgerock. openig. handler. Static Response Handler

Filters

Filter objects perform filtering of the request and response of an HTTP exchange.

Table of Contents

AssignmentFilter	
CaptureFilter	37
CookieFilter	39
EntityExtractFilter	41
ExceptionFilter	45
FileAttributesFilter	47
HeaderFilter	49
CryptoHeaderFilter	51
HttpBasicAuthFilter	53
OAuth2ClientFilter	55
OAuth2ResourceServerFilter	63
ScriptableFilter	69
SqlAttributesFilter	71
StaticRequestFilter	
SwitchFilter	75

AssignmentFilter

AssignmentFilter — conditionally assign values to expressions

1 Description

Conditionally assigns values to expressions before and after the exchange is handled.

2 Usage

```
"name": string,
    "type": "AssignmentFilter",
    "config": {
        "onRequest": [
                 "condition": expression,
                "target": lvalue-expression,
                "value": expression
            }, ...
        "onResponse": [
                 "condition": expression,
                 "target": lvalue-expression,
                "value": expression
            }, ...
        1
    }
}
```

3 Properties

"onRequest": array of objects, optional

Defines a list of assignment bindings to evaluate before the exchange is handled.

"onResponse": array of objects, optional

Defines a list of assignment bindings to evaluate after the exchange is handled.

"condition": expression, optional

Expression to evaluate to determine if an assignment should occur. Omitting the condition makes the assignment unconditional.

"target": *lvalue-expression*, required

Expression that yields the target object whose value is to be set.

"value": expression, optional

Expression that yields the value to be set in the target.

4 Example

This is an example of how you would capture credentials and store them in the OpenIG session during a login request. Notice the credentials are captured on the request, but not marked as valid until the response returns a positive 302. The credentials would then be used to login a user to a different application.

```
"name": "PortalLoginCaptureFilter",
 "type": "AssignmentFilter",
 "config": {
     "onRequest": [
               "target": "${exchange.session.authUsername}",
"value": "${exchange.request.form['username'][0]}",
          },
               "target": "${exchange.session.authPassword}",
               "value": "${exchange.request.form['password'][0]}",
          },
               "comment": "Indicates authentication has not yet been confirmed.",
              "target": "${exchange.session.authConfirmed}",
"value": "${false}",
     "onResponse": [
               "condition": "${exchange.response.status == 302}",
               "target": "${exchange.session.authConfirmed}",
               "value": "${true}",
 }
}
```

5 Javadoc

org.forgerock.openig.filter.AssignmentFilter

CaptureFilter

CaptureFilter — capture request and response messages

1 Description

Captures request and response messages for further analysis.

2 Usage

```
{
    "name": string,
    "type": "CaptureFilter",
    "config": {
        "file": expression,
        "charset": string,
        "condition": expression,
        "captureEntity": boolean
    }
}
```

3 Properties

"file": expression, required

The path of the file where captured output should be written.

"charset": *string*, *optional*

The character set to encode captured output with. Default: "UTF-8".

"condition": expression, optional

The condition to evaluate to determine whether to capture an exchange. Default: unconditional.

"captureEntity": boolean, optional

Whether the message entity should be captured.

The filter omits binary entities, instead writing a [binary entity] marker to the file.

Default: true

4 Examples

Log the entire request and response:

```
{
    "name": "LogToTemporaryFile",
```

```
"type": "CaptureFilter",
"config": {
    "file": "/tmp/gateway.log",
    }
}
```

Log the request and response. Do not log the entity:

```
{
    "name": "LogToTemporaryFile",
    "type": "CaptureFilter",
    "config": {
        "file": "/tmp/gateway.log"
        "captureEntity": false,
    }
}
```

You can use the CaptureFilter to capture the exchange before sending the request and when receiving the response as in the following example.

```
{
   "name": "OutgoingChain",
   "type": "Chain",
   "config": {
        "filters": [ "LogToTemporaryFile" ],
        "handler": "ClientHandler"
   }
},
{
   "name": "LogToTemporaryFile",
   "type": "CaptureFilter",
   "config": {
        "captureEntity": false,
        "file": "/tmp/gateway.log"
}
```

5 Javadoc

org. forgerock. openig. filter. Capture Filter

CookieFilter

CookieFilter — manage, suppress, relay cookies

1 Description

Manages, suppresses and relays cookies. Managed cookies are intercepted by the cookie filter itself and stored in the gateway session; managed cookies are not transmitted to the user agent. Suppressed cookies are removed from both request and response. Relayed cookies are transmitted freely between user agent and remote server and vice-versa.

If a cookie does not appear in one of the three action parameters, then the default action is performed, controlled by setting the defaultAction parameter. If unspecified, the default action is to manage all cookies. In the event a cookie appears in more than one configuration parameter, then it will be selected in the order of precedence: managed, suppressed, relayed.

2 Usage

```
{
    "name": string,
    "type": "CookieFilter",
    "config": {
        "managed": [ string, ... ],
        "suppressed": [ string, ... ],
        "relayed": [ string, ... ],
        "defaultAction": string
    }
}
```

3 Properties

"managed": array of strings, optional
A list of the names of cookies to be managed.

"suppressed": array of strings, optional
A list of the names of cookies to be suppressed.

"relayed": *array of strings, optional*A list of the names of cookies to be relayed.

"defaultAction": string, optional
Action to perform for cookies that do not match an action set. Must be one of:
"MANAGE", "RELAY", "SUPPRESS". Default: "MANAGE".

4 Javadoc

org. forgerock. openig. filter. Cookie Filter

EntityExtractFilter

EntityExtractFilter — extract pattern from message entity

1 Description

Extracts regular expression patterns from a message entity. The extraction results are stored in a target object. For a given matched pattern, the value stored in the object is either the result of applying its associated pattern template (if specified) or the match result itself otherwise.

2 Usage

3 Properties

"messageType": *string*, *required*

The message type in the exchange to extract patterns from. Must be one of: "REQUEST", "RESPONSE".

"charset": *string*, *optional*

Overrides the character set encoding specified in message. Default: the message encoding is used.

"target": lvalue-expression, required

Expression that yields the target object that contains the extraction results.

The "bindings" determine what type of object is stored in the target location. The object stored in the target location is a Map<String, String>. You can then access its content with \$\{target.key\}\ or \$\{target['key']\}.

"key": string, required

Name of element in target object to contain an extraction result.

"pattern": pattern, required

The regular expression pattern to find in the entity.

"template": pattern-template, optional

The template to apply to the pattern and store in the named target element. Default: store the match result itself.

4 Examples

Extracts a nonce from the response, which is typically a login page, and sets its value in the exchange to be used by the downstream filter posting the login form. The nonce value would be accessed using the following expression \${exchange.wikiNonce.wpLoginToken}. The pattern is finding all matches in the HTTP body of the form wpLogintoken value="abc". Setting the template to \$1 assigns the value abc to exchange.wikiNonce.wpLoginToken:

Reads the response looking for the OpenAM login page. When found it sets loginPage.found = true to be used in a SwitchFilter to post the login credentials:

5 Javadoc

org. forgerock. openig. filter. Entity Extract Filter

ExceptionFilter

ExceptionFilter — catch exceptions when handling request

1 Description

Catches any exceptions thrown during handling of a request. This allows friendlier error pages to be displayed than would otherwise be displayed by the container. Caught exceptions are logged with a log level of WARNING and the exchange is diverted to the specified exception handler.

2 Usage

```
{
    "name": string,
    "type": "ExceptionFilter",
    "config": {
        "handler": string,
    }
}
```

3 Properties

"handler": string, required

The name of the handler heap object to dispatch to in the event of caught exceptions.

4 Javadoc

org.forgerock.openig.filter.ExceptionFilter

FileAttributesFilter

FileAttributesFilter — retrieve record from a file

1 Description

Retrieves and exposes a record from a delimiter-separated file. Lookup of the record is performed using a specified key, whose value is derived from an exchange-scoped expression. The resulting record is exposed in an object whose location is specified by the target expression. If a matching record cannot be found, then the resulting object is empty.

The retrieval of the record is performed lazily; it does not occur until the first attempt to access a value in the target. This defers the overhead of file operations and text processing until a value is first required. This also means that the value expression is not evaluated until the object is first accessed.

2 Usage

```
"name": string,
  "type": "FileAttributesFilter",
  "config": {
      "file": expression,
      "charset": string,
      "separator": string,
      "header": boolean,
      "fields": [ string, ... ],
      "target": lvalue-expression,
      "key": string,
      "value": expression
}
```

For an example see the section, Login With Credentials From a File.

3 Properties

```
"file": expression, required

The file containing the record to be read.
```

"charset": string, optional

The character set the file is encoded in. Default: "UTF-8".

"header": boolean, optional
Indicates the first line of the file contains the set of defined field keys.
Default: true.

"fields": array of strings, optional

Explicit field keys in the order they appear in a record, overriding any existing field header. Default: use field header.

"target": lvalue-expression, required

Expression that yields the target object to contain the record.

The target object is a Map<String, String>, where the "fields" are the keys. For example, if the target is \${exchange.credentials} and the record has a "username" field and a "password" field mentioned in the "fields" list, Then you can access the user name as \${exchange.credentials.username} and the password as \${exchange.credentials.password}.

"key": *string*, *required*

The name of the field in the file to perform the lookup on.

"value": expression, required

Expression that yields the value to be looked-up within the file.

4 Javadoc

org.forgerock.openig.filter.FileAttributesFilter

HeaderFilter

HeaderFilter — remove and add headers

1 Description

Removes headers from and adds headers to a message. Headers are added to any existing headers in the message. To replace, remove the header and add it.

2 Usage

```
{
    "name": string,
    "type": "HeaderFilter",
    "config": {
        "messageType": string,
        "remove": [ string, ... ],
        "add": {
            name: [ string, ... ], ...
        }
    }
}
```

3 Properties

 $\verb"messageType": string, required"$

Indicates the type of message in the exchange to filter headers for. Must be one of: "REQUEST", "RESPONSE".

"remove": array of strings, optional

The names of header fields to remove from the message.

"add": object, optional

Header fields to add to the message. The name specifies the header name, with an associated array of string values.

4 Examples

Replace the host header on the incoming request with myhost.com:

```
{
   "name": "ReplaceHostFilter",
   "type": "HeaderFilter",
   "config": {
        "messageType": "REQUEST",
        "remove": [ "host" ],
        "add": {
            "host": [ "myhost.com" ]
        }
}
```

}

Add a Set-Cookie header in the response:

```
{
    "name": "SetCookieFilter",
    "type": "HeaderFilter",
    "config": {
        "messageType": "RESPONSE",
        "add": {
            "Set-Cookie": [ "mysession=12345" ]
        }
    }
}
```

Add headers custom1 and custom2 to the request:

5 Javadoc

org.forgerock.openig.filter.HeaderFilter

CryptoHeaderFilter

CryptoHeaderFilter — encrypt, decrypt headers

1 Description

Encrypts or decrypts headers in a request or response.

2 Usage

```
"name": string,
    "type": "CryptoHeaderFilter",
    "config": {
        "messageType": string,
        "operation": string,
        "algorithm": string,
        "key": expression,
        "keyType": string,
        "headers": [ string, ... ]
}
```

3 Properties

"messageType": string, required

Indicates the type of message in the exchange whose headers to encrypt or decrypt. Must be one of: "REQUEST", "RESPONSE".

"operation": string, required

Indicates whether to encrypt or decrypt. Must be one of: "ENCRYPT", "DECRYPT".

"algorithm": string

Algorithm used for encryption and decryption. Defaults to DES/ECB/NoPadding.

"key": *expression, required*Base64 encoded key value.

"headers": array of strings, optional

The names of header fields to encrypt or decrypt.

4 Example

```
{
    "name": "DecryptReplayPasswordFilter",
    "type": "CryptoHeaderFilter",
    "config": {
```

```
"messageType": "REQUEST",
"operation": "DECRYPT",
"algorithm": "DES/ECB/NoPadding",
"keyType": "DES",
"key": "oqdP3DJdE1Q=",
"headers": [ "replaypassword" ]
}
```

5 Javadoc

org. forgerock. openig. filter. Crypto Header Filter

HttpBasicAuthFilter

HttpBasicAuthFilter — perform HTTP Basic authentication

1 Description

Performs authentication through the HTTP Basic authentication scheme. For more information, see RFC 2617.

If challenged for authentication via a 401 Unauthorized status code by the server, this filter retries the request with credentials attached. Once an HTTP authentication challenge is issued from the remote server, all subsequent requests to that remote server that pass through the filter includes the user credentials.

If authentication fails (including the case of no credentials yielded from expressions), then the exchange is diverted to the specified authentication failure handler.

2 Usage

```
{
    "name": string,
    "type": "HttpBasicAuthFilter",
    "config": {
        "username": expression,
        "password": expression,
        "failureHandler": string,
        "cacheHeader": boolean
    }
}
```

3 Properties

"username": expression, required

Expression that yields the username to supply during authentication.

 $\verb"password": expression, required"$

Expression that yields the password to supply during authentication.

"failureHandler": string, required

The name of the handler heap object to dispatch to if authentication fails.

"cacheHeader": boolean, optional

Whether to cache credentials in the session after the first successful authentication, and then replay those credentials for subsequent authentications in the same session.

With "cacheHeader": false, the filter generates the header for each request. This is useful for example when users change their passwords during a browser session.

Default: true

4 Example

```
{
   "name": "TomcatAuthenticator",
   "type": "HttpBasicAuthFilter",
   "config": {
        "username": "tomcat",
        "password": "tomcat",
        "failureHandler": "TomcatAuthFailureHandler",
        "cacheHeader": false
   }
}
```

5 Javadoc

org. forgerock. openig. filter. Http Basic Auth Filter

OAuth2ClientFilter

OAuth2ClientFilter — Authenticate an end user with OAuth 2.0 delegated authorization

1 Description

An OAuth2ClientFilter is a filter that authenticates an end user using OAuth 2.0 delegated authorization. The filter can act as an OpenID Connect relying party as well as an OAuth 2.0 client.

The filter configuration includes the client credentials that are used to authenticate to identity providers. The client credentials can be included directly in the configuration, or retrieved in some other way using an expression.

In the case where all users share the same identity provider, you can configure the filter as a client of a single provider. You can also configure the filter to work with multiple providers, taking the user to a login handler page—often full of provider logos, and known as a *Nascar page*. The name comes from Nascar race cars, some of which are covered with sponsors' logos.—to choose a provider.

What an OAuth2ClientFilter does depends on the incoming request URI. In the following list <code>clientEndpoint</code> represents the value of the "clientEndpoint" in the filter configuration.

clientEndpoint/login/provider?goto=url

Redirect the end user for authorization with the specified provider.

The provider then authenticates the end user and obtains authorization consent from the end user before redirecting the user-agent back to the callback client endpoint.

Ultimately if the entire process is successful, the filter saves the authorization state in the exchange and redirects the user-agent to the specified URL.

clientEndpoint/logout?goto=url

Remove the authorization state for the end user and redirect to the specified URL.

clientEndpoint/callback

Handle the callback from the OAuth 2.0 authorization server that occurs as part of the authorization process.

If the callback is handled successfully, the filter saves the authorization state in the exchange at the specified "target" location and redirects to the URL during login.

Other request URIs

Restore authorization state in the specified "target" location and call the next filter or handler in the chain.

2 Usage

```
"name": string,
  "type": "OAuth2ClientFilter",
  "config": {
    "clientEndpoint": expression,
    "failureHandler": handler,
    "loginHandler": handler,
    "providerHandler": handler,
    "providers": [ provider configuration, ... ],
    "target": expression,
    "defaultLoginGoto": expression,
    "defaultLogoutGoto": expression.
    "requireHttps": boolean,
    "requireLogin": boolean,
    "scopes": [ expression, ...],
    "useJWTSession": boolean
 }
}
```

3 Properties

"clientEndpoint": expression, required
Base URI for the filter.

For example, if you set "clientEndpoint": "/openid", then the service URIs for this filter on your OpenIG server are /openid/login, /openid/logout, and /openid/callback.

"failureHandler": handler, required

The name of the handler configuration object to invoke when authentication fails.

If this handler is invoked, then the "target" in the exchange is populated with information about the provider, and the error

The failure object in the "target" is a simple map. It has the following layout.

```
{
    "provider": "provider name string",
    "error": {
        "realm": "optional string",
        "scope": [ "optional required scope string", ... ],
        "error": "optional string",
```

In the failure object, the following fields are not always present. Their presence depends on when the failure occurs.

- "access token"
- "id token"
- "token type"
- "expires in"
- "scope"
- "client endpoint"

"loginHandler": handler, required when multiple providers are configured
The name of the handler configuration object to invoke if the user must
choose a provider.

This handler allows the user to choose a provider, as in the following example that allows the user to choose between "openam" and "google".

"providerHandler": handler, required

The name of the handler configuration object to invoke to communicate with the provider.

Usually set this to the name of a ClientHandler configured in the heap, or a chain that ends in a ClientHandler.

"providers": array of provider configuration objects, required

One or more provider configuration objects that indicate how the client communicates with authorization providers.

If the provider has a well-known configuration URL as defined for OpenID Connect 1.0 Discovery that returns JSON with at least authorization and token endpoint URLs, then you can specify that URL in the provider configuration. Otherwise, you must specify at least the provider authorization and token endpoint URLs, and optionally the user info endpoint URL.

Provider configuration objects have the following layout:

```
{
   "name": string,
   "clientId": expression,
   "clientSecret": expression,
   "wellKnownConfiguration": URL string,
   "authorizeEndpoint": URI expression,
   "tokenEndpoint": URI expression,
   "userInfoEndpoint": URI expression,
   "scopes": [ "expression", ... ]
}
```

The provider configuration object properties are as follows.

"name": string, required

A name for the provider configuration.

"clientID": expression, required

The client_id obtained when registering with the provider.

"clientSecret": expression, required

The client_secret obtained when registering with the provider.

"wellKnownConfiguration": $URL\ string,\ required\ unless\ "authorizeEndpoint"$ and "tokenEndpoint" are specified

The URL to the well-known configuration resource as described in OpenID Connect 1.0 Discovery.

"authorizeEndpoint": expression, required unless obtained through "wellKnownConfiguration"

The URL to the provider's OAuth 2.0 authorization endpoint.

"tokenEndpoint": expression, required unless obtained through

"wellKnownConfiguration"

The URL to the provider's OAuth 2.0 token endpoint.

"userInfoEndpoint": expression, optional

The URL to the provider's OpenID Connect UserInfo endpoint.

Default: no UserInfo is obtained from the provider.

"scopes": array of expressions, optional

Overrides the list of scopes specified globally for the filter.

Default: use the list of scopes specified globally for the filter.

"target": expression, required

Expression that yields the target object whose value is to be set.

"defaultLoginGoto": expression, optional

The URI to redirect to after successful authentication and authorization.

Default: return an empty page.

"defaultLogoutGoto": expression, optional

The URI to redirect to after successful logout.

Default: return an empty page.

"requireHttps": boolean, optional

Whether to require that requests use the HTTPS scheme.

Default: true.

"requireLogin": boolean, optional

Whether to require authentication for all incoming requests.

Default: true.

"scopes": array of expressions, optional

Expression that yields the scope strings to request of any providers contacted by this filter.

Instead of or in addition to specifying scopes globally here, you can also specify a list of scopes per provider in each provider's configuration. Perprovider scope lists then override this list.

Default: do not specify scopes.

"useJWTSession": boolean, optional

Whether to store authorization state in a JWT cookie on the user-agent.

At present this setting has no effect. Authorization state is stored only in the session.

Default: false.

4 Example

The following example configures an OAuth 2.0 client filter. The base client endpoint is /openid. The filter uses well-known configuration endpoints to obtain configuration information for OpenAM and for Google as providers. The client credentials are not shown.

When a incoming request is made to /openid/login, this filter takes the user to a "NascarPage" to choose an identity provider. It then handles negotiation for authorization with the provider, requesting the scopes defined in "scopes".

If the authorization process completes successfully, then the filter injects the authorization state data into exchange openid.

At the end of the exchange, the aim of this configuration is simply to dump the data obtained back in the response.

```
"name": "OpenIDConnectClient".
    "type": "OAuth2ClientFilter",
    "config": {
        "target"
                                : "${exchange.openid}",
        "scopes"
                               : ["openid", "profile", "email"],
                               : "/openid",
: "NascarPage",
        "clientEndpoint"
        "loginHandler"
                               : "Dump"
        "failureHandler"
        "providerHandler"
                               : "ClientHandler",
        "defaultLoginGoto"
                               : "/dump".
                               : "/unprotected",
        "defaultLogoutGoto"
        "requireHttps"
                                : false,
        "requireLogin"
                                : true,
        "providers"
            {
                                : "openam",
                "name"
                "wellKnownConfiguration"
                  "http://openam.example.com:8080/openam/.well-known/openid-configuration",
                "clientId" : "******"
                "clientSecret" : "******
            },
                "name"
                                : "google",
                "wellKnownConfiguration"
                  "https://accounts.google.com/.well-known/openid-configuration",
                           : "***********.apps.googleusercontent.com",
                "clientId"
                "clientSecret" : "**********
        1
    }
}
```

Notice that this configuration is for development and testing purposes only, and is not secure ("requireHttps": false). Make sure you do require HTTPS in production environments.

5 Javadoc

org. forgerock. openig. filter. oauth 2. client. OAuth 2 Client Filter

6 See Also

The OAuth 2.0 Authorization Framework

OAuth 2.0 Bearer Token Usage

 $\begin{array}{c} \textbf{OpenID Connect site, in particular the list of standard OpenID Connect 1.0 scope} \\ \textbf{values} \end{array}$

61

OAuth2ResourceServerFilter

 ${\tt OAuth2ResourceServerFilter-Validate\ an\ Exchange\ containing\ an\ OAuth\ 2.0}$ access token

1 Description

An OAuth2ResourceServerFilter is a filter that validates an exchange containing an OAuth 2.0 access token. The filter expects an OAuth 2.0 token from the HTTP Authorization header of the request, such as the following example header, where the OAuth 2.0 access token is 1fc0e143-f248-4e50-9c13-1d710360cec9.

```
Authorization: Bearer 1fc0e143-f248-4e50-9c13-1d710360cec9
```

The filter extracts the access token, and then validates it against the configured "tokenInfoEndpoint" URL.

On successful validation, the filter includes the token info from the authorization server response as JSON in the exchange. Subsequent filters and handlers can access the token info as exchange.oauth2AccessToken.

Regarding errors, if the filter configuration and access token together result in an invalid request to the authorization server, the filter returns an HTTP 400 Bad Request response to the user-agent.

If the access token is missing from the request, the filter returns an HTTP 401 Unauthorized response to the user-agent.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="OpenIG"
```

If the access token is not valid, for example because it has expired, the filter also returns an HTTP 401 Unauthorized response to the user-agent.

If the scopes for the access token do not match the specified required scopes, the filter returns an HTTP 403 Forbidden response to the user-agent.

2 Usage

```
{
    "name": string,
    "type": "0Auth2ResourceServerFilter",
    "config": {
        "httpHandler": string,
        "requiredScopes": [ string, ... ],
```

```
"tokenInfoEndpoint": URL string,
"cacheExpiration": duration string,
"enforceHttps": boolean,
"realm": string
}
}
```

An alternative value for "type" is "OAuth2RSFilter".

3 Properties

"httpHandler": string, required

HTTP client handler to send token info requests.

Usually set this to the name of a ClientHandler configured in the heap.

"requiredScopes": array of strings, required

The list of required OAuth 2.0 scopes for this protected resource.

"tokenInfoEndpoint": URL string, required

The URL to the token info endpoint of the OAuth 2.0 authorization server.

"cacheExpiration": duration string, optional

Duration for which to cache OAuth 2.0 access tokens.

A duration is a lapse of time expressed in English, such as "23 hours 59 minutes and 59 seconds".

Durations are not case sensitive.

The following units can be used in durations.

- "indefinite", "infinity", "undefined", "unlimited": unlimited duration
- "days", "day", "d": days
- "hours", "hour", "h": hours
- "minutes", "minute", "min", "m": minutes
- "seconds", "second", "sec", "s": seconds
- "milliseconds", "millisecond", "millisec", "millis", "milli", "ms": milliseconds
- "microseconds", "microsecond", "microsec", "micros", "micro", "us": microseconds
- "nanoseconds", "nanosecond", "nanosec", "nanos", "nano", "ns": nanoseconds

Default: 1 minute

"enforceHttps": boolean, optional

Whether to require that requests use the HTTPS scheme.

Default: true

"realm": string, optional

HTTP authentication realm to include in the WWW-Authenticate response header field when returning an HTTP 401 Unauthorized status to a useragent that need to authenticate.

Default: OpenIG

4 Example

The following example configures an OAuth 2.0 protected resource filter that expects scopes "email" and "profile" (and returns an HTTP 403 Forbidden status if the scopes are not present), and validates access tokens against the OpenAM token info endpoint. It caches access tokens for up to 2 minutes.

```
{
  "name": "ProtectedResourceFilter",
  "type": "OAuth2ResourceServerFilter",
  "config": {
        "httpHandler": "ClientHandler",
        "requiredScopes": [
        "email",
        "profile"
        ],
        "tokenInfoEndpoint": "https://openam.example.com:8443/openam/oauth2/tokeninfo",
        "cacheExpiration": "2 minutes"
    }
}
```

5 Javadoc

org.forgerock.openig.filter.oauth2.OAuth2ResourceServerFilter

6 See Also

The OAuth 2.0 Authorization Framework

OAuth 2.0 Bearer Token Usage

RedirectFilter

RedirectFilter — rewrites Location headers

1 Description

Rewrites Location headers on responses that generate a redirect that would take the user directly to the application being proxied rather than taking the user through OpenIG.

For example, if OpenIG listens on https://proxy.example.com:443/ and the application it protects listens on http://www.example.com:8080/, then you can configure this filter to rewrite redirects that would take the user to locations under http://www.example.com:8080/ to go instead to locations under https://proxy.example.com:443/.

The current implementation supports only HTTP 302 redirects.

2 Usage

```
{
   "name": string,
   "type": "RedirectFilter",
   "config": {
      "baseURI": expression
   }
}
```

3 Properties

"baseURI": expression, required

The base URI of the OpenIG instance. This is used to rewrite the Location header on the redirect response.

4 Example

```
{
    "name": "LocationRewriter",
    "type": "RedirectFilter",
    "config": {
        "baseURI": "https://proxy.example.com:443/"
     }
}
```

5 Javadoc

org. forgerock. openig. filter. Redirect Filter

ScriptableFilter

ScriptableFilter — process exchange by using a script

1 Description

Processes an exchange by using a script.

The script has access to the following global objects.

exchange

The exchange provides access to the HTTP request and response.

globals

This Map holds variables that persist across successive invocations.

http

The http object provides an embedded HTTP client.

Use this client to perform outbound HTTP requests.

ldap

The ldap object provides an embedded LDAP client.

Use this client to perform outbound LDAP requests, such as LDAP authentication.

logger

The logger object provides access to the server log sink.

next

The next object refers to the next handler in the filter chain.

When finished processing the request, call the next.handle(exchange) method to call the next filter in the current chain. Everything in the script subsequent to this method call deals with the exchange response.

2 Usage

```
{
  "name": string,
  "type": "ScriptableFilter",
  "config": {
      "type": string,
      "file": string, // Use either "file"
      "source": string // or "source", but not both.
}
}
```

3 Properties

"type": string, required

The Internet media type (formerly MIME type) of the script, either "application/x-groovy" for Groovy or "text/javascript" for JavaScript

"file": string

Path to the file containing the script; mutually exclusive with "source"

Relative paths in the "file" field are relative to the base location for scripts. The base location depends on the configuration. For details see the section, *Installing OpenIG*.

The base location for Groovy scripts is on the classpath when the scripts are executed. If therefore some Groovy scripts are not in the default package, but instead have their own package names, they belong in the directory corresponding to their package name. For example, a script in package com.example.groovy belongs under <code>openig-base/scripts/groovy/com/example/groovy/.</code>

"source": string

The script as a string; mutually exclusive with "file"

4 Javadoc

org.forgerock.openig.filter.ScriptableFilter

SqlAttributesFilter

SqlAttributesFilter — execute SQL query

1 Description

Executes a SQL query through a prepared statement and exposes its first result. Parameters in the prepared statement are derived from exchange-scoped expressions. The query result is exposed in an object whose location is specified by the target expression. If the query yields no result, then the resulting object is empty.

The execution of the query is performed lazily; it does not occur until the first attempt to access a value in the target. This defers the overhead of connection pool, network and database query processing until a value is first required. This also means that the parameters expressions is not evaluated until the object is first accessed.

2 Usage

```
{
    "name": string,
    "type": "SqlAttributesFilter",
    "config": {
        "dataSource": string,
        "preparedStatement": string,
        "parameters": [ expression, ... ],
        "target": lvalue-expression
    }
}
```

3 Properties

"dataSource": string, required

The JNDI name of the factory for connections to the physical data source.

"preparedStatement": string, required

The parameterized SQL query to execute, with? parameter placeholders.

"parameters": array of expressions, required

The parameters to evaluate and include in the execution of the prepared statement.

"target": *lvalue-expression*, required

Expression that yields the target object that will contain the query results.

4 Example

Using the users sessionid from a cookie, query the database to find the user logged in and set the profile attributes in the exchange:

```
{
          "name": "SqlAttributesFilter",
          "type": "SqlAttributesFilter",
          "config": {
                 "target": "${exchange.sql}",
                 "dataSource": "java:comp/env/jdbc/mysql",
"preparedStatement": "SELECT f.value AS 'first', l.value AS

'last', u.mail AS 'email', GROUP_CONCAT(CAST(r.rid AS CHAR)) AS

'roles'
                    FROM sessions s
                   INNER JOIN users u
                   ON ( u.uid = s.uid AND u.status = 1 )
                    LEFT OUTER JOIN profile_values f
                   ON ( f.uid = u.uid AND \overline{f}.fid = 1 )
                   LEFT OUTER JOIN profile_values l
                   ON (l.uid = u.uid AND \overline{l}.fid = 2)
                   LEFT OUTER JOIN users roles r
                   ON (r.uid = u.uid)
                   WHERE (s.sid = ? AND s.uid <> 0) GROUP BY s.sid;",
                 "parameters": [ "${exchange.request.cookies
                    [keyMatch(exchange.request.cookies,'JSESSION1234')]
                    [0].value}" ]
           }
 }
```

Lines are folded for readability in this example. In your JSON, keep the values for "preparedStatement" and "parameters" on one line.

5 Javadoc

org. forgerock. openig. filter. Sql Attributes Filter

StaticRequestFilter

StaticRequestFilter — create new request within exchange object

1 Description

Creates a new request within the exchange object. It replaces any request that may already be present in the exchange. The request can include a form, specified in the form parameter, which is included in an entity encoded in application/x-www-form-urlencoded format if request method is POST, or otherwise as (additional) query parameters in the URI.

2 Usage

```
{
    "name": string,
    "type": "StaticRequestFilter",
    "config": {
        "method": string,
        "version": string,
        "restore": boolean,
        "headers": {
            name: [ expression, ... ], ...
        },
        "form": {
            param: [ expression, ... ], ...
        }
    }
}
```

3 Properties

```
Properties
"method": string, required
    The HTTP method to be performed on the resource (for example, "GET").
"uri": string, required
    The fully-qualified URI of the resource to access (for example, "http://www.example.com/resource.txt").
"version": string, optional
    Protocol version. Default: "HTTP/1.1".
"restore": boolean, optional
    Whether to restore the original request after the exchange is handled.
    Default: false
"headers": object, optional
    Header fields to set in the request.
```

The name specifies the header name. Its value is an array of expressions to evaluate as header values.

"form": object, optional
A form to include in the request.

The param specifies the form parameter name. Its value is an array of expressions to evaluate as form field values.

4 Example

```
{
    "name": "LoginRequestFilter",
    "type": "StaticRequestFilter",
    "config": {
        "method": "POST",
        "uri": "http://10.10.0.2:8080/wp-login.php",
        "form": {
            "log": [ "george" ],
            "pwd": [ "bosco" ],
            "rememberme": [ "forever" ],
            "redirect_to": [ "http://portal.example.com:8080/wp-admin/" ],
            "testcookie": [ "I" ]
        }
}
```

5 Javadoc

org.forgerock.openig.filter.StaticRequestFilter

SwitchFilter

SwitchFilter — divert exchange to other handler

1 Description

Conditionally diverts the exchange to another handler. If a condition evaluates to true, then the exchange is dispatched to the associated handler with no further processing by the switch filter.

2 Usage

3 Properties

"onRequest": array of objects, optional

Conditions to test (and handler to dispatch to, if true) before the exchange is handled.

"onResponse": array of objects, optional

Conditions to test (and handler to dispatch to, if true) after the exchange is handled.

"condition": expression, optional

Condition to evaluate to determine if exchange should be dispatched to handler. Default: unconditional dispatch to handler.

"handler": string, required

The name of the handler heap object to dispatch to if condition yields true.

4 Example

This example intercepts the response if it is equal to 200 and executes the LoginRequestHandler. This filter might be used in a login flow where the request for the login page must go through to the target, but the response should be intercepted in order to send the login form to the application. This is typical for scenarios where there is a hidden value or cookie returned in the login page which must be sent in the login form:

5 Javadoc

org. forgerock. openig. filter. Switch Filter

Miscellaneous Heap Objects

Table of Contents

ConsoleLogSink	79
HttpClient	81
TemporaryStorage	87

ConsoleLogSink

ConsoleLogSink — log to standard error

1 Description

A log sink that writes log entries to the standard error stream.

2 Usage

```
{
    "name": string,
    "type": "ConsoleLogSink",
    "config": {
        "level": string
     }
}
```

3 Properties

"level": string, optional

The level of log entries to display in the console. Must be one of: OFF, ERROR, WARNING, INFO, CONFIG, STAT, DEBUG, TRACE, ALL. Default: INFO.

4 Example

```
{
   "name": "LogSink",
   "comment": "Default sink for logging information.",
   "type": "ConsoleLogSink",
   "config": {
        "level": "DEBUG"
   }
}
```

5 Javadoc

org.forgerock.openig.log.ConsoleLogSink

org. forgerock. openig. log. Log Level

HttpClient

HttpClient — group settings and submit requests to remote servers

1 Description

Groups settings and submits requests to remote servers.

You configure HttpClient objects in order to specify HTTP client settings for ClientHandler objects.

2 Usage

```
"name": string,
"type": "HttpClient",
  "config": {
     "connections": number,
     "disableReuseConnection": boolean,
     "disableRetries": boolean,
     "hostnameVerifier": string,
     "soTimeout": duration string,
     "connectionTimeout": duration string,
     "keystore": {
          "file": expression,
"password": expression,
"alg": string,
"type": string
     },
"truststore": {
          "file": expression,
          "password": expression,
          "alg": string,
"type": string
  }
}
```

3 Properties

"connections": number, optional

The maximum number of connections to open, from 1-64 inclusive.

Default: 64

"connectionTimeout": duration string, required

Amount of time to wait to establish a connection, expressed as a duration

A duration is a lapse of time expressed in English, such as "23 hours 59 minutes and 59 seconds".

Durations are not case sensitive.

The following units can be used in durations.

- "indefinite", "infinity", "undefined", "unlimited": unlimited duration
- "days", "day", "d": days
- "hours", "hour", "h": hours
- "minutes", "minute", "min", "m": minutes
- "seconds", "second", "sec", "s": seconds
- "milliseconds", "millisecond", "millisec", "millis", "milli", "ms": milliseconds
- "microseconds", "microsecond", "microsec", "micros", "micro", "us": microseconds
- "nanoseconds", "nanosecond", "nanosec", "nanos", "nano", "ns": nanoseconds

Default: 10 seconds

"disableRetries": boolean, optional

Whether to disable automatic retries for failed requests.

Default: true

"disableReuseConnection": boolean, optional
Whether to disable connection reuse.

Default: true

"hostnameVerifier": string, optional

How to handle hostname verification for outgoing SSL connections.

Set this to one of the following values.

- ALLOW ALL: turn off verification.
- BROWSER_COMPATIBLE: match the hostname either as the value of the the first CN, or any of the subject-alt names.

A wildcard can occur in the CN, and in any of the subject-alt names. Wildcards match all subdomains, so *.example.com matches www.example.com and some.host.example.com.

• STRICT: match the hostname either as the value of the the first CN, or any of the subject-alt names.

A wildcard can occur in the CN, and in any of the subject-alt names. Wildcards match one domain level, so *.example.com matches www.example.com but not some.host.example.com.

Default: ALLOW ALL

"keystore": object, optional

Configuration of the key store holding this client's keys and certificates.

The "keystore" object has the following fields.

"file" expression, required

Absolute path name of the key store file

 $\verb"password" \it expression, required"$

Password for the key store

"alg" string, optional

Certificate algorithm to use

Default: SunX509

"type" *string, optional*Key store format

Default: IKS

"soTimeout": duration string, required

Socket timeout, after which stalled connections are destroyed, expressed as a duration

A duration is a lapse of time expressed in English, such as "23 hours 59 minutes and 59 seconds".

Durations are not case sensitive.

The following units can be used in durations.

- "indefinite", "infinity", "undefined", "unlimited": unlimited duration
- "days", "day", "d": days
- "hours", "hour", "h": hours
- "minutes", "minute", "min", "m": minutes
- "seconds", "second", "sec", "s": seconds
- "milliseconds", "millisecond", "millisec", "millis", "milli", "ms": milliseconds

- "microseconds", "microsecond", "microsec", "micros", "micro", "us": microseconds
- "nanoseconds", "nanosecond", "nanosec", "nanos", "nano", "ns": nanoseconds

Default: 10 seconds

"truststore": object, optional

Configuration of the trust store holding server certificates that this client trusts.

The "truststore" object has the following fields.

"file" expression, required

Absolute path name of the trust store file

"alg" string, optional

Certificate algorithm to use

Default: SunX509

"password" *expression*, *optional*Password for the trust store

"type" *string, optional*Key store format

Default: JKS

4 Example

```
"name": "HttpClient",
  "type": "HttpClient",
  "config": {
     "connections": 64,
     "disableReuseConnection": true,
     "disableRetries": true,
     "hostnameVerifier": "ALLOW_ALL",
     "soTimeout": "10 seconds",
     "connectionTimeout": "10 seconds",
     "keystore": {
         "file": "/path/to/keystore.jks",
         "password": "changeit"
    },
"truststore": {
         "file": "/path/to/keystore.jks",
         "password": "changeit"
}
```

5 Javadoc

org. forgerock. openig. http. Http Client

TemporaryStorage

TemporaryStorage — cache streamed content

1 Description

Allocates temporary buffers for caching streamed content during request processing. Initially uses memory; when the memory limit is exceeded, switches to a temporary file.

2 Usage

```
{
    "name": string,
    "type": "TemporaryStorage",
    "config": {
        "initialLength": number,
        "memoryLimit": number,
        "fileLimit": number,
        "directory": string
    }
}
```

3 Properties

"initialLength": number, optional

The initial length of memory buffer byte array. Default: 8192 (8 KiB).

"memoryLimit": number, optional

The length limit of the memory buffer. Exceeding this limit results in promotion from memory to file. Default: 65536 (64 KiB).

"fileLimit": number, optional

The length limit of the file buffer. Exceeding this limit results in a thrown exception. Default: 1048576 (1 MiB).

"directory": *string*, *optional*

The directory where temporary files are created. If omitted, then the system-dependent default temporary directory is used (typically "/tmp" on Unix systems). Default: use system-dependent default.

4 Javadoc

org.forgerock.openig.io.TemporaryStorage

Expressions

Many configuration parameters support dynamic expressions.

Table of Contents

Expressions	91
Functions	95
Patterns	05

Expressions

Expressions — expression configuration parameter values

1 Description

Expressions are specified as configuration parameter values for a number of built-in objects. Such expressions conform to the Universal Expression Language as specified in JSR-245.

2 General Syntax

All expressions follow standard Universal Expression Language syntax: \${expression}. The expression can be a simple reference to a value, a function call, or arbitrarily complex arithmetic, logical, relational and conditional operations. When supplied within a configuration parameter, an expression is always a string enclosed in quotation marks, for example: "\${exchange.request.method}".

3 Value Expressions

A value expression references a value relative to the scope supplied to the expression. In the current version of OpenIG, the supplied scope is always the HTTP exchange object. For example "\${exchange.request.method}" references the method of an incoming HTTP request in the exchange scope.

An *lvalue-expression* is a specific type of value expression that references a value to be written. For example, "\${exchange.session.gotoURL}" specifies a session attribute named gotoURL to write a value to. Attempts to write values to read-only values are ignored.

4 Indexed Properties

Properties of values are accessed using the . and [] operators, and can be nested arbitrarily.

The value expressions $\$\{exchange.request\}$ and $\$\{exchange['request']\}$ are equivalent.

In the case of arrays, the index of an element in the array is expressed as a number in brackets. For example, "\${exchange.request.headers['Content-Type'][0]}" references the first Content-Type header value in a request. If a property does not exist, then the index reference yields a null (empty) value.

5 Operations

Universal Expression Language supports arbitrarily complex arithmetic, logical, relational and conditional operations. They are, in order of precedence:

- Index property value: [], .
- Change precedence of operation: ()
- Unary negative: -
- Logical operations: not, !, empty
- Arithmetic operations: *, /, div, %, mod
- Binary arithmetic operations: +, -
- Relational operations: <, >, <=, >=, lt, gt, le, ge, ==, !=, eq, ne
- Logical operations: &&, and, ||, or
- Conditional operations: ?, :

6 System Properties & Environment Variables

You can use expressions to retrieve Java system properties, and to retrieve environment variables.

For system properties, \${system['property']} yields the value of property, or null if there is no value for property. For example, \${system['user.home']} yields the home directory of the user running the application server for OpenIG.

For environment variables, \${env['variable']} yields the value of variable, or null if there is no value for variable. For example, \${env['HOME']} yields the home directory of the user running the application server for OpenIG.

7 Functions

A number of built-in functions can be called within an expression.

Syntax is \${function(parameter, ...)}, where zero or more parameters are supplied to the function. For example, "\${toLowerCase(exchange.request. method)}" yields the method of the request, converted to lower case. Functions can be operands for operations, and can yield parameters for other function calls.

8 Examples

In your JSON, keep the values on one line.

```
"${exchange.request.uri.path == '/wordpress/wp-login.php' and exchange.request.form['action'][0] != 'logout'}"
"${exchange.request.uri.host == 'wiki.example.com'}"
"${exchange.request.cookies[keyMatch(exchange.request.cookies,'^SESS.*')][0].value}"
"${toString(exchange.request.uri)}"
"${exchange.request.method == 'POST' and exchange.request.uri.path == '/wordpress/wp-login.php'}"
"${exchange.request.method != 'GET'}"
"${exchange.request.headers['cookie'][0]}"
"${exchange.request.uri.scheme == 'http'}"
"${exchange.request.uri.scheme == 'http'}"
"${exchange.response.status == 302 and not empty exchange.session.gotoURL)}"
"${exchange.response.headers['Set-Cookie'][0]}"
"${exchange.request.headers['host'][0]}"
```

9 See Also

Exchange

Functions

93

Functions

Functions — built-in functions to call within expressions

1 Description

A set of built-in functions that can be called from within expressions.

2 contains

```
contains(object, value)
```

Returns true if the object contains the specified value. If the object is a string, a substring is searched for the value. If the object is a collection or array, its elements are searched for the value.

Parameters

object

the object to be searched for the presence of.

value

the value to be searched for.

Returns

true

if the object contains the specified value.

3 decodeBase64

decodeBase64(string)

Returns the base64-decoded string, or null if the string is not valid Base64.

Parameters

string

The base64-encoded string to decode.

Returns

string

The base64-decoded string.

4 encodeBase64

```
encodeBase64(string)
```

Returns the base64-encoded string, or null if the string is null.

Parameters

string

The string to encode into Base64.

Returns

string

The base64-encoded string.

5 indexOf

```
indexOf(string, substring)
```

Returns the index within a string of the first occurrence of a specified substring.

Parameters

string

the string to be searched.

substring

the value to search for within the string.

Returns

the index of the first instance of substring, or -1 if not found.

6 join

join(strings, separator)

Joins an array of strings into a single string value, with a specified separator.

Parameters

separator

the separator to place between joined elements.

strings

the array of strings to be joined.

Returns

the string containing the joined strings.

7 keyMatch

keyMatch(map, pattern)

Returns the first key found in a map that matches the specified regular expression pattern, or null if no such match is found.

Parameters

map

the map whose keys are to be searched.

pattern

a string containing the regular expression pattern to match.

Returns

the first matching key, or null if no match found.

8 length

length(object)

Returns the number of items in a collection, or the number of characters in a string.

Parameters

object

the object whose length is to be determined.

Returns

the length of the object, or 0 if length could not be determined.

9 matchingGroups

matchingGroups(string, pattern)

Returns an array of matching groups for a regular expression pattern against a string, or null if no such match is found. The first element of the array is the entire match, and each subsequent element correlates to any capture group specified within the regular expression.

Parameters

string

the string to be searched.

pattern

a string containing the regular expression pattern to match.

Returns

an array of matching groups, or null if no such match is found.

10 matches

matches(string, pattern)

Returns true if the string contains the specified regular expression pattern.

Parameters

string

the string to be searched.

pattern

a string containing the regular expression pattern to find.

Returns

true

if the string contains the specified regular expression pattern.

11 read

read(string)

Takes a file name as a string, and returns the content of the file as a plain string, or null on error (due to the file not being found, for example).

Either provide the absolute path to the file, or a path relative to the location of the Java system property user.dir.

Parameters

string

The name of the file to read.

Returns

string

The content of the file or null on error.

12 readProperties

readProperties(string)

Takes a Java Properties file name as a string, and returns the content of the file as a key/value map of properties, or null on error (due to the file not being found, for example).

Either provide the absolute path to the file, or a path relative to the location of the Java system property user.dir.

For example, to get the value of the key property in the properties file /path/to/my.properties, use \${readProperties('/path/to/my.properties')['key']}.

Parameters

string

The name of the Java Properties file to read.

Returns

object

The key/value map of properties or null on error.

13 split

```
split(string, pattern)
```

Splits a string into an array of substrings around matches of the given regular expression pattern.

Parameters

string

the string to be split.

pattern

the regular expression to split substrings around.

Returns

the resulting array of split substrings.

14 toLowerCase

toLowerCase(string)

Converts all of the characters in a string to lower case.

Parameters

string

the string whose characters are to be converted.

Returns

the string with characters converted to lower case.

15 toString

toString(object)

Returns the string value of an arbitrary object.

Parameters

object

the object whose string value is to be returned.

Returns

the string value of the object.

16 toUpperCase

toUpperCase(string)

Converts all of the characters in a string to upper case.

Parameters

string

the string whose characters are to be converted.

Returns

the string with characters converted to upper case.

17 trim

trim(string)

Returns a copy of a string with leading and trailing whitespace omitted.

Parameters

string

the string whose white space is to be omitted.

Returns

the string with leading and trailing white space omitted.

18 trim

urlDecode(string)

Returns the URL decoding of the provided string.

Parameters

string

The string to be URL decoded, which may be null.

Returns

The URL decoding of the provided string, or null if string was null.

19 trim

urlEncode(string)

Returns the URL encoding of the provided string.

Parameters

string

The string to be URL encoded, which may be null.

Returns

The URL encoding of the provided string, or null if string was null.

20 Javadoc

org. forgerock. openig. el. Functions

Patterns

Patterns — regular expression patterns

1 Description

Patterns in configuration parameters and expressions use the standard Java regular expression Pattern class. For more information on regular expressions, see Oracle's tutorial on Regular Expressions.

2 Pattern Templates

A regular expression pattern template expresses a transformation to be applied for a matching regular expression pattern. It may contain references to capturing groups within the match result. Each occurrence of g (where g is an integer value) is substituted by the indexed capturing group in a match result. Capturing group zero "g" denotes the entire pattern match. A dollar sign or numeral literal immediately following a capture group reference can be included as a literal in the template by preceding it with a backslash (\). Backslash itself must be also escaped in this manner.

3 See Also

Java Pattern class

Regular Expressions tutorial

Exchange Object Model

Expressions are evaluated within an exchange object model scope.

Table of Contents

Exchange	109
Request	111
Principal	113
Response	
URI	

Exchange

Exchange — HTTP exchange of request and response

1 Description

The root object for the exchange object model: an HTTP exchange of request and response. The exchange object model parallels the document object model, exposing elements of the exchange. It supports this by exposing a set of fixed properties and allowing arbitrary properties to be added.

2 Properties

"exchange": object

Self-referential property to make this the root object in the exchange object model.

"request": object

The request portion of the HTTP exchange.

"response": object

The response portion of the HTTP exchange.

"principal": object

The principal associated with the request, or null if unknown.

"session": object

Session context associated with the remote client. Exposes session attributes as name-value pairs, where both name and value are strings.

3 Javadoc

org.forgerock.openig.http.Exchange

Request

Request — HTTP exchange request

1 Description

An HTTP request message in an exchange object model.

2 Properties

"method": string

The method to be performed on the resource. Example: "GET".

"uri": object

The fully-qualified URI of the resource being accessed. Example: "http://www.example.com/resource.txt".

"version": string

Protocol version. Example: "HTTP/1.1".

"headers": object

Exposes message header fields as name-value pairs, where name is header name and value is an array of header values.

"cookies": object

Exposes incoming request cookies as name-value pairs, where name is cookie name and value is an array of string cookie values.

"form": object

Exposes query parameters and/or application/x-www-form-urlencoded entity as name-value pairs, where name is the field name and value is an array of string values.

"entity": object

The message entity body (no accessible properties).

3 Javadoc

org.forgerock.openig.http.Request

Principal

Principal — user principal in HTTP exchange

1 Description

Represents a user principal in an exchange object model, containing the name of the current authenticated user.

2 Properties

"name": string

The name of the principal, or null if principal is undefined (user has not been authenticated).

3 Javadoc

java.security.Principal

Response

Response — HTTP exchange response

1 Description

An HTTP response message in an exchange object model.

2 Properties

```
"status": number
```

The response status code. Example: 200.

"reason": *string*

The response status reason. Example: "OK".

"version": string

Protocol version. Example: "HTTP/1.1".

"headers": object

Exposes message header fields as name-value pairs, where name is header name and value is an array of header values.

"entity": object

The message entity body (no accessible properties).

3 Javadoc

org.forgerock.openig.http.Response

URI

URI — Uniform Resource Identifier in HTTP exchange

1 Description

Represents a Uniform Resource Identifier (URI) reference in an exchange object model.

2 Properties

"scheme": string

The scheme component of the URI, or null if the scheme is undefined.

"authority": string

The decoded authority component of the URI, or null if the authority is undefined.

Use "rawAuthority" to access the raw (encoded) component.

"userInfo": string

The decoded user-information component of the URI, or null if the user information is undefined.

Use "rawUserInfo" to access the raw (encoded) component.

"host": string

The host component of the URI, or null if the host is undefined.

"port": number

The port component of the URI, or null if the port is undefined.

"path": string

The decoded path component of the URI, or null if the path is undefined.

Use "rawPath" to access the raw (encoded) component.

"query": *string*

The decoded query component of the URI, or null if the query is undefined.

Use "rawQuery" to access the raw (encoded) component.

"fragment": string

The decoded fragment component of the URI, or null if the fragment is undefined.

Use "rawFragment" to access the raw (encoded) component.

3 Javadoc

org.forgerock.openig.util.MutableUri

Appendix A. Release Levels & Interface Stability

This appendix includes ForgeRock definitions for product release levels and interface stability.

A.1 ForgeRock Product Release Levels

ForgeRock defines Major, Minor, and Maintenance product release levels. The release level is reflected in the version number. The release level tells you what sort of compatibility changes to expect.

Major (version: x.0.0)

Major releases bring big new features. Major releases can include changes even to Stable interfaces. Major releases can remove previously Deprecated functionality, and in rare cases remove Evolving functionality that has not been explicitly Deprecated. Major releases also include the changes present in previous Minor and Maintenance releases.

Minor (version: x.y.0)

Minor releases might include new features, backwards-compatible changes to Stable interfaces in the same Major release, and incompatible changes to Evolving interfaces. Minor releases can remove previously Deprecated functionality. Minor releases also include the changes present in Maintenance releases.

ForgeRock Product Interface Stability

Maintenance (version: x.v.z)

Maintenance releases can include bug fixes. Maintenance releases are intended to be fully compatible with previous versions from the same Minor release.

A.2 ForgeRock Product Interface Stability

ForgeRock products support many protocols, APIs, GUIs, and command-line interfaces. Some of these interfaces are standard and very stable. Others offer new functionality that is continuing to evolve.

We realize that you invest in these interfaces, and therefore must know when and how ForgeRock expects them to change. For that reason, ForgeRock defines interface stability labels and uses these definitions in ForgeRock products.

Stable

This documented interface is expected to undergo only backwards-compatible changes between major releases. Changes are announced at least one minor release before they take effect.

Evolving

This documented interface is continuing to evolve and so is expected to change, potentially in backwards-incompatible ways even in a minor release. Changes are documented at the time of product release.

While new protocols and APIs are still in the process of standardization, they are Evolving. This applies for example to recent Internet-Draft implementations, and also to newly developed functionality.

Deprecated

This interface is deprecated and likely to be removed in a future release. For previously stable interfaces, the change was likely announced in a previous release. Deprecated interfaces will be removed from ForgeRock products.

Removed

This interface was deprecated in a previous release and has now been removed from the product.

Internal/Undocumented

Internal and undocumented interfaces can change without notice. If you depend on one of these interfaces, contact ForgeRock support or email info@forgerock.com to discuss your needs.

Index

Ε

Exchange Object Model
Exchange, 109
Principal, 113
Request, 111
Response, 115
URI, 117
Expressions
Expressions, 91
Functions, 95
Patterns, 105

F

Filters

AssignmentFilter, 35 CaptureFilter, 37 CookieFilter, 39 CryptoHeaderFilter, 51 EntityExtractFilter, 41 ExceptionFilter, 45 FileAttributesFilter, 47 HeaderFilter, 49 HttpBasicAuthFilter, 53 OAuth2ClientFilter, 55 OAuth2ResourceServerFilter, 63 RedirectFilter, 67 ScriptableFilter, 69 SqlAttributesFilter, 71 StaticRequestFilter, 73 SwitchFilter, 75

Н

Handlers
Chain, 17
ClientHandler, 19
DispatchHandler, 21
Route, 23
Router, 25
ScriptableHandler, 27
SequenceHandler, 29
StaticResponseHandler, 31

M

Miscellaneous Heap Objects ConsoleLogSink, 79 HttpClient, 81 TemporaryStorage, 87

R

Required configuration Gateway servlet, 11 Heap objects, 13