

OpenGL 2.1.0 Reference

**Paul Bryan
Mark Craig
Jamie Nelson
May 17, 2012**

Copyright © 2011-2012 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.

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, MERCHANTABILITY, 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.

DejaVu Fonts

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 LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, 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@gnome.org.

Arev Fonts Copyright

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 LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, 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 Tavmjong 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 Tavmjong Bah. For further information, contact: tavmjong@free.fr.

Table of Contents

Preface v

I. Required Configuration 7

II. Servlets 13

III. Handlers 23

IV. Filters 35

V. Miscellaneous Heap Objects 61

VI. Expressions 67

VII. Exchange Object Model 81

Index 93

Preface

This reference covers OpenIG configuration.

1. Who Should Use this Reference

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

2. Formatting Conventions

Some items are formatted differently from other text, like filenames, **commands**, and literal values.

```
$ echo Terminal sessions are formatted with lines folded for easier reading.  
In HTML documents click the [-] image for a flat, copy-paste version. Click  
the [+] image for an expanded, line-wrapped version. > /dev/null
```

In many cases, sections pertaining to UNIX, GNU/Linux, Mac OS X, BSD, and so forth are marked (UNIX). Sections pertaining to Microsoft Windows might be marked (Windows). To avoid repetition, however, file system directory names are often given only in UNIX format as in /path/to/OpenIG, even if the text applies to C:\path\to\OpenIG as well.

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

3. Accessing OpenIG Documentation Online

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. The review process should slow authors down enough that documentation you get with a stable release has had time to bake fully.

Fully baked core documentation is available at docs.forgerock.org.

The OpenAM Wiki offers additional content related to ForgeRock access management software. You are welcome to sign up and then edit the Wiki if you notice an error, or if you have something to share.

4. Joining the OpenIG Community

After you sign up at ForgeRock, you can also login to the Wiki and the issue database to follow what is happening with the project.

If you have questions regarding OpenIG which 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.

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.

Name

Gateway Servlet — entry point for all incoming requests

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 servlet (which is itself a heap object). The configuration of the gateway servlet is loaded from a JSON-encoded configuration file, typically located at `~/.ForgeRock/OpenIG/config.json`.

Usage

```
{  
  "heap": { heap-configuration object },  
  "servletObject": string  
}
```

Properties

"heap": *object, required*

The heap object configuration.

"servletObject": *string, required*

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

Javadoc

`org.forgerock.openig.gateway.GatewayServlet`

Name

Heap Objects — configure and initialize objects, with dependency injection

Description

A heap is a collection of associated objects, initialized from declarative configuration artifacts. All configurable objects in the gateway 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.

Usage

```
{
  "objects": [
    {
      "name": string,
      "type": string,
      "config": { object-specific configuration }
    }, ...
  ]
}
```

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. Example: "HandlerServlet". To determine the type name, see the specific object documentation in this reference.

"config": *object, required*

The configuration that is specific to the heap object being created. For an example, see the HandlerServlet reference.

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.

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".

Servlets

The gateway includes objects that can create, initialize and dispatch to Java Enterprise Edition Servlets and Servlet filters.

Name

DispatchServlet — dispatch request based on extra path information

Description

Dispatches requests to mapped filters and servlets based on request's extra path information. The extra path information is the path that follows the path of the dispatch servlet itself, but precedes the query string. It is guaranteed to be a value that always begins with a "/" character.

All filters that match the pattern are invoked in the order they are expressed in the "bindings" list until a matching servlet is encountered. The first matching servlet object in the bindings list is invoked, and terminates any further processing of the request. If no matching servlet is found, an exception is thrown. To avoid this, a final "catch-all" servlet binding with a pattern of ".*" is recommended.

Usage

```
{
  "name": string,
  "type": "DispatchServlet",
  "config": {
    "bindings": [
      {
        "pattern": pattern,
        "object": string,
      }, ...
    ]
  }
}
```

Properties

"bindings": *array of objects, required*

A list of bindings of patterns and associated servlets/filters to dispatch to.

"pattern": *pattern, required*

The regular expression pattern to match against the incoming request extra path information.

"object": *string, required*

The name of the HTTP servlet or servlet filter heap object to dispatch to if the regular expression pattern matches.

Example

Sample from a federation configuration. Federation is implemented as its own servlet which will be dispatched-to based on the incoming URI starting with "/saml". All other requests will go through the HandlerServlet.

```
{
  "name": "Dispatcher",
  "type": "DispatchServlet",
  "config": {
    "bindings": [
      {
        "pattern": "^/saml",
        "object": "FederationServlet"
      },
      {
        "pattern": ".*",
        "object": "HandlerServlet"
      }
    ]
  }
}
```

Javadoc

org.forgerock.openig.servlet.DispatchServlet

Name

HandlerServlet — translate and marshal request to a handler

Description

Services a servlet request by translating it and marshaling to a handler.

Usage

```
{
  "name": string,
  "type": "HandlerServlet",
  "config": {
    "handler": string,
    "baseURI": string
  }
}
```

Properties

"handler": *string, required*

The name of the handler heap object to dispatch the exchange to.

"baseURI": *string, optional*

Overrides request URLs constructed by container, making requests relative to a new base URI. Only scheme, host and port are used in the supplied URI. Default: use container URL.

Javadoc

org.forgerock.openig.servlet.HandlerServlet

Name

HttpServlet — create stock servlet in heap environment

Description

Creates and initializes a stock servlet in a heap environment.

Usage

```
{
  "name": string,
  "type": "javax.servlet.http.HttpServlet",
  "config": {
    "initParams": {
      name: string, ...
    }
  }
}
```

Properties

"initParams": *object of name-value pairs*

Initialization parameters to supply to servlet. The name is a string containing the name of the initialization parameter.

Javadoc

org.forgerock.openig.servlet.HttpServletHeaplet

Name

ServletFilter — create stock servlet filter in heap environment

Description

Creates and initializes a stock servlet filter in a heap environment.

Usage

```
{
  "name": string,
  "type": "javax.servlet.Filter",
  "config": {
    "initParams": {
      name: string, ...
    }
  }
}
```

Properties

"initParams": *object of name-value pairs*

Initialization parameters to supply to servlet filter. The name is a string containing the name of the initialization parameter.

Javadoc

org.forgerock.openig.servlet.ServletFilterHeaplet

Handlers

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

Name

Chain — dispatch exchange to ordered list of filters

Description

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

Usage

```
{
  "name": string,
  "type": "Chain",
  "config": {
    "filters": [ string, ... ],
    "handler": string
  }
}
```

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.

Example

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

Javadoc

org.forgerock.openig.filter.Chain

Name

ClientHandler — submit exchange requests to remote servers

Description

Submits exchange requests to remote servers.

Note

This handler does not verify hostnames for outgoing SSL connections. This is because the gateway usually accesses the SSL endpoint using a raw IP address rather than a fully-qualified hostname.

Usage

```
{
  "name": string,
  "type": "ClientHandler",
  "config": {
    "connections": number
  }
}
```

Properties

"connections": *number, optional*

The maximum number of concurrent connections to open. Default: 64

Javadoc

org.forgerock.openig.handler.ClientHandler

Name

DispatchHandler — dispatch to one of a list of handlers

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.

Usage

```
{
  "name": string,
  "type": "DispatchHandler",
  "config": {
    "bindings": [
      {
        "condition": expression,
        "handler": string,
        "baseURI": string,
      }, ...
    ]
  }
}
```

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.

Example

```
{
  "name": "Dispatcher",
  "type": "DispatchHandler",
  "config": {
    "bindings": [
```

```
        {
            "condition": "${exchange.request.uri.path == '/login.php'}",
            "handler": "LoginChain",
        },
        {
            "condition": "${exchange.request.uri.path == '/logout.php'}",
            "handler": "LogoutChain",
        },
        {
            "handler": "ClientHandler",
        }
    ]
}
```

Javadoc

org.forgerock.openig.handler.DispatchHandler

Name

SequenceHandler — process exchange through sequence of handlers

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.

Usage

```
{
  "name": string,
  "type": "SequenceHandler",
  "config": {
    "bindings": [
      {
        "handler": string,
        "postcondition": expression
      }
    ]
  }
}
```

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.

Javadoc

org.forgerock.openig.handler.SequenceHandler

Name

StaticResponseHandler — create static response in HTTP exchange

Description

Creates a static response in an HTTP exchange.

Usage

```
{
  "name": string,
  "type": "StaticResponseHandler",
  "config": {
    "status": number,
    "reason": string,
    "version": string,
    "headers": {
      name: [ expression, ... ], ...
    },
    "entity": string
  }
}
```

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, required*

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

"entity": *string, optional*

The message entity to write in the response. Conforms to the set Content-Type header and sets Content-Length.

Example

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

```
}
```

Javadoc

`org.forgerock.openig.handler.StaticResponseHandler`

Filters

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

Name

AssignmentFilter — conditionally assign values to expressions

Description

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

Usage

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

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.

Example

This is an example of how you would capture credentials and store them in the Gateway 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}",
      }
    ]
  }
}
```

Javadoc

org.forgerock.openig.filter.AssignmentFilter

Name

CaptureFilter — capture request and response messages

Description

Captures request and response messages for further analysis.

Usage

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

Properties

"file": *string, 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*

Indicates that message entity should be captured. Default: true.

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,
    }
  }
```

Normal usage of the CaptureFilter is to create an OutgoingChain. This enables you to call filters like the CaptureFilter, before sending out the request:

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

Javadoc

org.forgerock.openig.filter.CaptureFilter

Name

CookieFilter — manage, suppress, relay cookies

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.

Usage

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

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".

Javadoc

org.forgerock.openig.filter.CookieFilter

Name

EntityExtractFilter — extract pattern from message entity

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.

Usage

```
{
  "name": string,
  "type": "EntityExtractFilter",
  "config": {
    "messageType": string,
    "charset": string,
    "target": lvalue-expression,
    "bindings": [
      {
        "key": string,
        "pattern": regex-pattern,
        "template": pattern-template
      }, ...
    ]
  }
}
```

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.

"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.

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`:

```
{
  "name": "WikiNoncePageExtract",
  "type": "EntityExtractFilter",
  "config": {
    "messageType": "response",
    "target": "${exchange.wikiNonce}",
    "bindings": [
      {
        "key": "wpLoginToken",
        "pattern": "wpLoginToken\"|s.*value=\\\"(.*)\\\"|\"",
        "template": "$1"
      }
    ]
  }
}
```

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:

```
{
  "name": "FindLoginPage",
  "type": "EntityExtractFilter",
  "config": {
    "messageType": "response",
    "target": "${exchange.isLoginPage}",
    "bindings": [
      {
        "key": "found",
        "pattern": "OpenAM\\s\\(Login\\)",
        "template": "true"
      }
    ]
  }
}
```

Javadoc

`org.forgerock.openig.filter.EntityExtractFilter`

Name

ExceptionHandler — catch exceptions when handling request

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.

Usage

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

Properties

"handler": *string, required*

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

Javadoc

org.forgerock.openig.filter.ExceptionFilter

Name

FileAttributesFilter — retrieve record from a file

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.

Usage

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

Properties

"file": string, 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 that will contain the record.

"key": *string, required*

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

"value": *expression, required*

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

Javadoc

org.forgerock.openig.filter.FileAttributesFilter

Name

HeaderFilter — remove and add headers

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.

Usage

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

Properties

"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.

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:

```
{
  "name": "SetCustomHeaders",
  "type": "HeaderFilter",
  "config": {
    "messageType": "REQUEST",
    "add": {
      "custom1": [ "12345", "6789" ],
      "custom2": [ "abcd" ]
    }
  }
}
```

Javadoc

org.forgerock.openig.filter.HeaderFilter

Name

CryptoHeaderFilter — encrypt, decrypt headers

Description

Encrypts or decrypts headers in a request or response.

Usage

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

Properties

"messageType": *string, required*

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

"operation": *string, required*

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

"algorithm": *string, required*

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

"key": *string, required*

Base64 encoded key value.

"headers": *array of strings, optional*

The names of header fields to encrypt or decrypt.

Example

```
{
  "name": "DecryptReplayPasswordFilter",
  "type": "CryptoHeaderFilter",
  "config": {
    "messageType": "REQUEST",
    "operation": "DECRYPT",
    "algorithm": "DES/ECB/NoPadding",
```

```
    "keyType": "DES",  
    "key": "oqdP3DJdE1Q=",  
    "headers": [ "replaypassword" ]  
}
```

Javadoc

org.forgerock.openig.filter.CryptoHeaderFilter

Name

HttpBasicAuthFilter — perform HTTP Basic authentication

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.

Usage

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

Properties

"username": *expression, required*

Expression that yields the username to supply during authentication.

"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.

Example

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

```
}
```

Javadoc

`org.forgerock.openig.filter.HttpBasicAuthFilter`

Name

SqlAttributesFilter — execute SQL query

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.

Usage

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

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.

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": "${exchange.sql}",
    "dataSource": "java:comp/env/jdbc/mysql",
    "preparedStatement": "SELECT f.value AS 'first', l.value AS
      'last', u.mail AS 'email', GROUP_CONCAT(CAST(r.rid AS CHAR)) AS
      'roles'
    FROM sessions s
    INNER JOIN users u
    ON ( u.uid = s.uid AND u.status = 1 )
    LEFT OUTER JOIN profile_values f
    ON ( f.uid = u.uid AND f.fid = 1 )
    LEFT OUTER JOIN profile_values l
    ON ( l.uid = u.uid AND l.fid = 2 )
    LEFT OUTER JOIN users_roles r
    ON ( r.uid = u.uid )
    WHERE (s.sid = ? AND s.uid <> 0) GROUP BY s.sid;",
    "parameters": [ "${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.

Javadoc

org.forgerock.openig.filter.SqlAttributesFilter

Name

StaticRequestFilter — create new request within exchange object

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.

Usage

```
{
  "name": string,
  "type": "StaticRequestFilter",
  "config": {
    "method": string,
    "uri": string,
    "version": string,
    "headers": {
      name: [ expression, ... ], ...
    },
    "form": {
      field: [ expression, ... ], ...
    }
  }
}
```

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".

"headers": *array of objects, required*

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

"form": *array of objects, required*

A form to include in the request. The field specifies the field name, with an associated array of expressions to evaluate as values.

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": [ "1" ]
  }
}
```

Javadoc

org.forgerock.openig.filter.StaticRequestFilter

Name

SwitchFilter — divert exchange to other handler

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.

Usage

```
{
  "name": string,
  "type": "SwitchFilter",
  "config": {
    "onRequest": [
      {
        "condition": expression,
        "handler": string,
      }, ...
    ],
    "onResponse": [
      {
        "condition": expression,
        "handler": string,
      }, ...
    ]
  }
}
```

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.

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:

```
{
  "name": "SwitchFilter",
  "type": "SwitchFilter",
  "config": {
    "onResponse": [
      {
        "condition": "${exchange.response.status == 200}",
        "handler": "LoginRequestHandler"
      }
    ]
  }
}
```

Javadoc

org.forgerock.openig.filter.SwitchFilter

Miscellaneous Heap Objects

Name

ConsoleLogSink — log to standard error

Description

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

Usage

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

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.

Example

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

Javadoc

org.forgerock.openig.log.ConsoleLogSink

org.forgerock.openig.log.LogLevel

Name

TemporaryStorage — cache streamed content

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.

Usage

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

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.

Javadoc

org.forgerock.openig.io.TemporaryStorage

Expressions

Many configuration parameters support dynamic expressions.

Name

Expressions — expression configuration parameter values

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.

General Syntax

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

Value Expressions

A value expression references a value relative to the scope supplied to the expression. In the current version of the gateway, the supplied scope is always the HTTP exchange object. For example `"${exchange.request.method}"` references the method of an incoming HTTP request in the exchange scope. A *lvalue-expression* is a specific type of value expression, which references a value to be written. For example, `"${exchange.session.gotoURL}"` specifies a session attribute named `gotoURL` to write a value to. Attempts to write values to read-only values are ignored.

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.

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: ?, :

Functions

A number of built-in functions can be called within an expression. Syntax is `${function(param, ...)}`, where zero or more parameters are supplied to the function. For example, `${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.

Examples

Lines are folded for readability in these example. 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.uri.value}"

"${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'}"

"${not (exchange.response.status 302 and not empty exchange.session.gotoURL)}"

"${exchange.response.headers['Set-Cookie'][0]}"

"${exchange.request.headers['host'][0]}"
```

See Also

Exchange

Functions

Name

Functions — built-in functions to call within expressions

Description

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

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.

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.

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.

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.

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.

matches

```
matches(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.

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.

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.

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.

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.

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.

Javadoc

org.forgerock.openig.el.Functions

Name

Patterns — regular expression patterns

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.

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 `"$0"` 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.

See Also

Java `Pattern` class

Regular Expressions tutorial

Exchange Object Model

Expressions are evaluated within an exchange object model scope.

Name

Exchange — HTTP exchange of request and response

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.

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.

Javadoc

org.forgerock.openig.http.Exchange

Name

Request — HTTP exchange request

Description

An HTTP request message in an exchange object model.

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).

Javadoc

org.forgerock.openig.http.Request

Name

Principal — user principal in HTTP exchange

Description

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

Properties

"name": *string*

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

Javadoc

`java.security.Principal`

Name

Response — HTTP exchange response

Description

An HTTP response message in an exchange object model.

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).

Javadoc

org.forgerock.openig.http.Response

Name

URI — Uniform Resource Identifier in HTTP exchange

Description

Represents a Uniform Resource Identifier (URI) reference in an exchange object model. URI properties are read-only.

Properties

"scheme": *string*

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

"schemeSpecificPart": *string*

The decoded scheme-specific part of the URI, never null.

"authority": *string*

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

"userInfo": *string*

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

"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.

"query": *string*

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

"fragment": *string*

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

Javadoc

java.net.URI

Index

E

Exchange Object Model

- Exchange, 83
- Principal, 87
- Request, 85
- Response, 89
- URI, 91

Expressions

- Expressions, 69
- Functions, 73
- Patterns, 79

F

Filters

- AssignmentFilter, 37
- CaptureFilter, 39
- CookieFilter, 41
- CryptoHeaderFilter, 51
- EntityExtractFilter, 43
- ExceptionFilter, 45
- FileAttributesFilter, 47
- HeaderFilter, 49
- HttpBasicAuthFilter, 53
- SqlAttributesFilter, 55
- StaticRequestFilter, 57
- SwitchFilter, 59

H

Handlers

- Chain, 25
- ClientHandler, 27
- DispatchHandler, 29
- SequenceHandler, 31
- StaticResponseHandler, 33

M

Miscellaneous Heap Objects

- ConsoleLogSink, 63
- TemporaryStorage, 65

R

Required configuration

Gateway servlet, 9

Heap objects, 11

S

Servlets

- DispatchServlet, 15
- HandlerServlet, 17
- Servlet, 19
- ServletFilter, 21

