Red Hat JBoss Portal 6.2 Administration and Configuration Guide

---

For use with Red Hat JBoss Portal 6.2 and its patch releases.

Jared Morgan Aakanksha Singh

Red Hat JBoss Portal 6.2 Administration and Configuration Guide

For use with Red Hat JBoss Portal 6.2 and its patch releases.

Jared Morgan
Red Hat, Ltd. Customer Content Services

Aakanksha Singh
Red Hat, Ltd. Customer Content Services

Legal Notice

Abstract

This book provides information to administrators for configuring and running Red Hat JBoss Portal. It forms part of the complete document suite available at .

Table of Contents

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **`gedit`** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **`Next`**. The character you sought will be highlighted in the

> **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above: username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

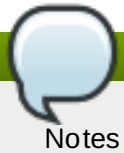
Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
         int r = 0;
         struct kvm_assigned_dev_kernel *match;

         mutex_lock(&kvm->lock);

         match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                       assigned_dev->assigned_dev_id);
         if (!match) {
                 printk(KERN_INFO "%s: device hasn't been assigned
```

```
before, "
                      "so cannot be deassigned\n", __func__);
                    r = -EINVAL;
                    goto out;
            }

            kvm_deassign_device(kvm, match);

            kvm_free_assigned_device(kvm, match);

out:
            mutex_unlock(&kvm->lock);
            return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com. From the Customer Portal, you can:

- Search or browse through a knowledge base of technical support articles about Red Hat products.
- Submit a support case to Red Hat Global Support Services (GSS).
- Access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click the name of any mailing list to subscribe to that list or to access the list archives.

2.2. We Need Feedback

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you. Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product Red Hat JBoss Portal.

When submitting a bug report, be sure to mention the manual's identifier: *Administration_and_Configuration_Guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Part I. Management

Chapter 1. Portal Management

The portal management component is used to manage portal's resources over common interfaces such as REST, CLI, and portlets or gadgets. This chapter discusses these interfaces and makes you familiar with general portal management concepts and terms. The chapter also covers specific management extensions included in the portal.

Report a bug

1.1. Important Terms used in Portal Management

Table 1.1. Important Terms

Term	Description
Management Extension	An extension to the management system which defines a managed component.
Managed Component	A managed component, which has been registered via an extension, serves as the root managed resource for a component.
Managed Resource	A managed resource is a uniquely identified self-describing 'resource' which can have operations and sub-resources registered to it.
Sub-resource	A managed resource whose parent is also a managed resource.
Address	A management address is the path of the managed resource with syntax similar to a file on a file system, such as: /foo/bar
Operation	An action that can be performed on a managed resource.
Attributes	Custom input parameters available to operations.

Report a bug

1.2. Goals for Portal Management

The goal of management component is to provide a foundation for managing portal side components. The management component allows management extensions to register resources and operations over a set of APIs. These APIs allows developers to expose management extensions over common interfaces such as REST and CLI. This approach makes management extensions independent of their interfaces and allows adding other interfaces without changing the extensions. The usage of APIs to obtain managed resources results in building consistent interfaces. Management of the same component becomes similar irrespective of the interface.

Report a bug

1.3. Operations for Management Extensions

The table lists the commonly used management extensions.

Table 1.2. Operations

Operation Name	Description

Operation Name	Description
read-resource	The **read-resource** operation is responsible for reading the managed resource. It describes itself and lists any operations or sub-resources it may contain. This is the primary operation to obtain information about a managed component and its managed resources.
remove-resource	The **remove-resource** operation is responsible for removing or deleting an existing managed resource.
update-resource	The **update-resource** operation is responsible for updating an existing managed resource's state.
read-config-as-xml	The **read-config-as-xml** operation is responsible for representing the current managed resource as an XML configuration.
export-resource	The **export-resource** operation is responsible for exporting a managed resource in a format that is acceptable and used in an import. The export-resource has a built-in functionality to recursively traverse managed resources until it finds a resource that supports an export-resource operation. So, you can register an export-resource operation on a sub-resource and it will be executed even by calling export-resource on any of its ancestors.
import-resource	The **import-resource** operation is responsible for importing managed resources that are exported through an export-resource operation.

Note

The read-resource operation is the only global operation supported in the core framework. All other operations need to be implemented by the extensions, since they are extension-specific.

Report a bug

1.4. Content Type for Portal Management

Content type defines the format of management requests and responses. The supported content types include JSON, XML, and Zip.

Restriction

Since **read-config-as-xml (XML)**, **export-resource (Zip)**, and **import-resource (Zip)** are content type specific operations, the response must be in the same format.

Aside from this restriction, it is up to the extension as to which content type is supported for each operation.

Report a bug

1.5. Path Templates used by Management Extensions

Path templates are used by management extensions to define dynamic content while registering resources. These path template variables are used during an export-resource operation to filter these managed resources. By specifying the **filter** attribute of an **export-resource** operation, managed resources can be explicitly included or excluded during export.

Example 1.1. Filter attribute syntax

```
[path-var]:(!)?[name],...[name];...[path-var]:(!)?[name],...[name]
```

where, **path-var** is the path template variable **name** is the name of a managed-resource and '!' char, which is optional, is to exclude that resource rather than include it.

Examples that use the path template variable **foo**:

Example 1.2. Include managed resource bar

```
foo:bar
```

Example 1.3. Include managed resource bar and foo-bar

```
foo:bar,foo-bar
```

Example 1.4. Exclude managed resource bar

```
foo:!bar
```

Example 1.5. Exclude managed resource bar and foo-bar

```
foo:!bar,foo-bar
```

Example 1.6. Multiple path template variables (foo and baz) separated by the ';' char

```
foo:bar,foo-bar;baz:blah
```

Note

For more information on the usage of path templates specific to a management extension, see chapter *Management Extensions*.

Report a bug

Chapter 2. REST Interface

The management REST component maps REST requests to management requests. To map the requests first the REST component locates the managed resource then maps the request URL to a management address and invokes an operation on that managed resource. The management REST component defines an entry point for REST clients, and exposes the registered managed resources and operations over REST.

Report a bug

2.1. Accessing REST Interface

To gain access to management resources and operations over REST a RESTful client must know the entry point URL.

Example 2.1. Entry point URL syntax

```
http(s)://<host>:<port>/<rest-context-name>/private/managed-components
```

where the rest-context-name is the portal container's rest context name.

For the default portal, the rest context name is 'rest'.

For a portal running on localhost and port 8080 the URL changes.

Example 2.2. URL for a portal running on localhost

```
http://localhost:8080/rest/private/managed-components/
```

Note

The REST URL is protected, and the authenticated user must belong to the 'administrators' group of the portal.

Report a bug

2.2. Resource URLs for REST Interface

REST resource URLs are mapped to management addresses, and since every managed resource has a unique management address, each managed resource can be represented by a unique REST URL.

Example 2.3. URL for identifying managed resources

```
http://<host>:<port>/<rest-context-name>/private/managed-components/<component-name>/<managed-resource-name>/<sub-resource-name>/.../<sub-resource-name>
```

Example 2.4. URL for identifying managed sub-resource

This URL uniquely identifies a managed resource named 'foo-bar', which is a sub-resource of 'bar', and 'bar' being a managed resource of the managed component 'foo'.

```
http://localhost:8080/rest/private/managed-components/foo/bar/foo-bar
```

Report a bug

2.3. HTTP Method

Table 2.1. Mapping HTTP methods to Operation names.

HTTP Method	Management Operation
GET	read-resource
PUT	update-resource
POST	add-resource
DELETE	remove-resource

This means that the same URL can invoke four different operations just by changing the HTTP method of the REST request.

Report a bug

2.4. Using Request Parameter

The management system supports more than four operations. These operations can be explicitly defined by including HTTP parameters as part of the REST request.

For example by adding the query parameter **op** to the request URL, clients can define what operation to invoke.

Example 2.5. Custom operation defined in request parameter

```
http://localhost:8080/rest/private/managed-components/foo/bar?op=some-custom-operation
```

Best practice to use the HTTP method

You can use the HTTP method to dictate the operation name. But a client can explicitly set operation names as request parameters.

URLs equivalent to a GET request are:

- http://localhost:8080/rest/private/managed-components/foo/bar
- http://localhost:8080/rest/private/managed-components/foo/bar?op=read-resource

Report a bug

2.5. Using URL Extension

REST resources can be represented as files, so two URL extensions have been added to support two common operations: **`read-config-as-xml`** and **`export-resource`**.

To invoke these operations add the following URL extensions at the end of the URL

Table 2.2. List of URL Extensions

URL Extension	Management Operation	Example URL
xml	read-config-as-xml	http://localhost:8080/rest/private/managed-components/foo/bar.xml
zip	export-resource	http://localhost:8080/rest/private/managed-components/foo/bar.zip

It is easier to specify the operation name as a file extension instead of specifying it as a request parameter.

The following URLs mean the same:

- http://localhost:8080/rest/private/managed-components/foo/bar.xml
- http://localhost:8080/rest/private/managed-components/foo/bar?op=read-config-as-xml

Report a bug

2.6. Management Attributes

Management attributes are part of a management request. They are mapped by including all request parameters of the HTTP request as attributes. If an operation supports certain attributes, query parameters can be added to the request URL to be used as attributes of the management request.

Example 2.6. Attributes first-name and last-name as request parameters

```
http://localhost:8080/rest/private/managed-components/foo/bar?first-name=john&last-name=doe
```

Report a bug

2.6.1. Multivalue Attributes

Management attributes can be multi-valued which means having more than one value associated with an attribute. HTTP query parameters can be multi-valued.

Example 2.7. Multi-valued attribute `colors` as request parameters

```
http://localhost:8080/rest/private/managed-components/foo/bar?
colors=red&colors=green&colors=blue
```

Report a bug

2.7. Content Control

The management framework defines Content Type to indicate the format of management requests and responses. Clients can dictate the content type of the management request by specifying the **`Accept`** and **`Content-Type`** headers of the HTTP request. The **`Accept`** header indicates the format of the response and the **`Content-Type`** header specifies the format of the request.A list of request headers that map to the Content Type of the management system is as follow.

Table 2.3. List of Request headers

Header	Content Type
application/json	JSON
application/xml	XML
application/zip	ZIP

Note

JSON is the default content type.

Report a bug

2.7.1. Browser Content Control

To control the content type of management requests through the browser, the REST component supports the **`format`** HTTP parameter to dictate the format of the response. This is because most browsers already send an **`Accept`** header.

Example 2.8. Specifying that the response should be returned as XML

```
http://localhost:8080/rest/private/managed-components/foo/bar?
format=xml
```

The **`format`** HTTP parameters that map to Content Types are shown in here:

Table 2.4. HTTP parameter format

Format parameter	Content Type
format=json	JSON
format=xml	XML

Important

Content negotiation is ignored for content type specific operations such as **`read-config-as-xml`** and **`export-resource`** since these cannot return different formats.

Report a bug

2.7.2. Operation Control

The **`read-resource`** operation is a built-in operation provided by the management component. The response is formatted as follows.

Example 2.9. HTTP Request

```
GET /managed-components
or
GET /managed-components?op=read-resource
```

Example 2.10. Response as JSON

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "description": "Available operations and children (sub-
resources).",
    "children": [
        {
            "name": "foo",
            "description": "Some description",
            "link": {
                "rel": "child",
                "href": "http://localhost:8080/rest/private/managed-
components/foo"
            }
        }
    ],
    "operations": [
        {
            "operation-name": "read-resource",
            "operation-description": "Lists information about a managed
resource, including available operations and children (sub-
resources).",
            "link": {
                "rel": "self",
                "href": "http://localhost:8080/rest/private/managed-
components"
                "type": "application/json"
                "method": "get"
```

```
            }
        }
    ]
}
```

Example 2.11. Response as XML

```
HTTP/1.1 200 OK
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resource>
    <description>Available operations and children (sub-
resources).</description>
    <children>
        <child>
            <name>foo</name>
            <description>Some description</description>
            <link href="http://localhost:8080/rest/private/managed-
components/foo" rel="child"/>
        </child>
    </children>
    <operations>
        <operation>
            <operation-name>read-resource</operation-name>
            <operation-description>Lists information about a managed
resource, including available operations and children (sub-
resources).</operation-description>
            <link href="http://localhost:8080/rest/private/managed-
components" rel="self" type="application/xml" method="get"/>
        </operation>
    </operations>
</resource>
```

Report a bug

Chapter 3. Command Line Interface

The command line interface component provides an interactive shell. The shell uses CRaSH to map commands to management requests. It connects over SSH, using the CRaSH SSH plugin.

For more information on CRaSH, see http://code.google.com/p/crsh/.

Report a bug

3.1. Deploying the Command Line Interface

The portal distribution includes an archive of the gatein-management-cli application.

To deploy the application, copy **JPP_DIST/gatein-management/gatein-management-cli.war** to **JPP_DIST/standalone/deployments/**.

Report a bug

3.2. Running the Command Line Interface

The syntax of the command to connect to CLI over SSH is as follows:

```
ssh -p <port> <user>@<host>
```

Example 3.1. SSH

Command: **ssh -p 2000 root@localhost**

You can change the default port that SSH listens on by changing the property **crash.ssh.port** in the **JPP_DIST/standalone/deployments/gatein-management-cli.war/WEB-INF/crash/crash.properties** file.

Example 3.2. crash.properties

```
# VFS configuration
crash.vfs.refresh_period=1

# SSH configuration
crash.ssh.port=2000
```

Important

Ensure the configured port is open, and not blocked by firewall settings.

Report a bug

3.3. Management Commands

The CLI component consist of management commands. These commands execute operations on managed resources in the portal. To view the list of available commands type **help** in the shell.

You can add the help option **-h** or **--help** for each command or execute the **man** command for detailed information about the command.

Report a bug

3.3.1. Using the mgmt command

The mgmt command allows a user to connect and disconnect from the management system. It also provides an **exec** command which allows custom control over executing operations with the management system.

Example 3.3. mgmt help

```
% mgmt -h
usage: mgmt[-h | --help] COMMAND [ARGS]

The most commonly used mgmt commands are:
   exec             Manually executes a management operation
   connect          login to gatein management
   disconnect       disconnect from management system

% mgmt connect -h
usage: mgmt [-h | --help] connect [-u | --username] [-p | --password] [-
c | --container]

   [-h | --help]      command usage
   [-u | --username]  the user name
   [-p | --password]  the user password
   [-c | --container] portal container name (default is portal)

%
```

The **mgmt connect** command allows you to connect to the management system. You can optionally specifying a portal container (default is 'portal') overriding the default container. The management commands are administrative operations so you must authenticate again in order to validate authorization to the portal container.

Example 3.4. Connecting to default portal container

```
% mgmt connect
Password:
Successfully connected to gatein management system: [user=root,
container='portal', host='127.0.0.1']

[ /]%
```

Example 3.5. Connecting to portal container 'sample-portal' as user john

```
% mgmt connect -u john -c sample-portal
Password:
Successfully connected to gatein management system: [user=john,
container='sample-portal', host='127.0.0.1']

[ /]%
```

The **mgmt exec** command allows complete control over what to send to the management system.

Example 3.6. mgmt exec help

```
[ /]% mgmt exec -h
usage: mgmt [-h | --help] exec [-c | --contentType] [-f | --file] [-a |
--attribute] [-o | --operation] path

   [-h | --help]        command usage
   [-c | --contentType] content type of an operation
   [-f | --file]        File name
   [-a | --attribute]   Specifies an attribute.
   [-o | --operation]   Operation name
   path

[ /]%
```

Example 3.7. Executing operation read-config-as-xml on managed component foo

```
[ /]% mgmt exec --operation read-config-as-xml --contentType xml /foo
<?xml version='1.0' encoding='UTF-8'?>
<data>...</data>

[ /]%
```

Report a bug

3.3.2. Using the cat command

The **cat** command executes the **read-config-as-xml** operation on a managed resource and outputs the XML data to the shell. The managed resource must support the **read-config-as-xml** operation.

Example 3.8. cat help

```
usage: cat [-h | --help] path

   [-h | --help] command usage
   path

[ /]% cat /foo
```

```
<?xml version='1.0' encoding='UTF-8'>
<data>...</data>

[ /]% [ /]% cat -h
```

Report a bug

3.3.3. Using the cd command

The cd command changes the current path of the CLI.

Example 3.9. cd help

```
  [ /]% cd -h
usage: cd [-h | --help] path

   [-h | --help] command usage
   path

[ /]%
```

Example 3.10. Change path to /foo/bar

```
[ /]% cd /foo/bar

[bar]%
```

Report a bug

3.3.4. Using the ls command

The **ls** command executes the **read-resource** operation on the current (or specified by the path) managed resource.

Example 3.11. ls help

```
 [ /]% ls -h
usage: ls [-h | --help] path

   [-h | --help] command usage
   path

[ /]% ls
foo
bar
```

```
[ /]% ls /foo
baz

[ /]%
```

Report a bug

3.3.5. Using the pwd command

The **pwd** command prints out the current resource path of the CLI.

Example 3.12. pwd help

```
[ /]% pwd -h
usage: pwd [-h | --help]

   [-h | --help] command usage

[ /]% pwd
/
[ /]% cd foo/baz

[baz]% pwd
/foo/baz

[baz]%
```

Report a bug

3.3.6. Using the export command

The **export** operation executes the **export-resource** operation on a managed resource and writes the content to the file.

Important

The CLI is connected to the portal server over SSH so the export command writes to the server's file system.

Example 3.13. export help

```
[ /]% export -h
usage: export [-h | --help] [-f | --file] [-r | --filter] path

   [-h | --help]   command usage
   [-f | --file]   File name
   [-r | --filter] For example, specifies the value of the filter to
```

```
use during an export.
   path

[ /]%
```

Example 3.14. Export resource foo to /tmp directory

```
[ /]% export --file /tmp foo
Export complete ! File location: /tmp/foo_2011-10-21_18-29-36.zip

[ /%]
```

Note

If you only specify a directory the export command will write a file with the name of the managed resource and a timestamp.

Example 3.15. Export resource /foo to /tmp/export-example.zip file

```
[ /]% export --file /tmp/export-example.zip foo
Export complete ! File location: /tmp/export-example.zip

[ /%]
```

Example 3.16. Export resource filtering on path template variable bar

```
[ /]% export --file /tmp/export-filter-example.zip --filter
bar:baz,foo-bar foo
Export complete ! File location: /tmp/export-filter-example.zip

[ /%]
```

Report a bug

3.3.7. Using the import command

The **`import`** command executes the **`import-resource`** operation on a managed resource.

Example 3.17. import help

```
[ /]% import -h usage: importfile [-h | --help] [-f | --file] [-m | --
importMode] path [-h | --help] command usage [-f | --file] File name [-
m | --importMode] The import mode for an import operation path [ /]%
```

Important

Since the CLI is connected to the portal server over SSH, the import command must specify a file that exists on the server.

Example 3.18. Import file /tmp/foo.zip to resource foo

```
  [ /]% import --file /tmp/foo.zip /foo Successfully imported file
/tmp/foo.zip [ /]%
```

Example 3.19. Import file /tmp/foo.zip to resource foo using importMode overwrite

```
 [ /]% import --file /tmp/foo.zip --importMode overwrite /foo
Successfully imported file /tmp/foo.zip [ /]%
```

Note

You can auto complete the import modes by typing --importMode and pressing the Tab key.

Report a bug

3.3.8. Using the Secure copy (SCP) command

The CLI application supports SCP to execute the **`export-resource`** and **`import-resource`** operations. These operations are executed in order to export to and import from a host other than the portal.

Exporting

To execute the **`export-resource`** operation using SCP, the source of the SCP command is the portal server and the target is the local file to which the resource is to be exported. The command syntax for exporting is: **`$ scp -P <port> <user>@<host>:{portal-container}:{path} <file>`** The **`portal-container`** is optional with default value being 'portal'. The **`path`** is the path of the resource supporting the **`export-resource`** operation. So to export 'foo/bar' resource assuming the CLI is deployed to 'example.org'.

Example 3.20. Export of foo/bar resource

```
$ scp -P 2000 'root@example.org:portal:/foo/bar' bar.zip
```

Example 3.21. Export of foo/bar resource with filter

Attributes like 'filter' can be added to the path of the SCP command just as you would add query parameters to a URL. **`$ scp -P 2000 'root@example.org:portal:/foo/bar?filter=bar:baz,foo-bar' foo.zip`**

Importing

To execute the **import-resource** operation using SCP, the source of the SCP command is the local file and the target is the portal server.The command syntax for exporting is: **$ scp -P <port> <file> <user>@<host>:{portal-container}:{path}** The **portal-container** is optional with default value being 'portal'. The **path** is the path of the resource that supports the **import-resource** operation. So to import a foo.zip file to the /foo resource assuming the CLI is deployed to 'example.org'.

> **Example 3.22. Import foo.zip to foo resource**
>
> ```
> $ scp -P 2000 foo.zip 'root@example.org:portal:/foo'
> ```

More specific examples of the SCP command can be found below pertaining to that particular management extension.

Report a bug

Chapter 4. Model Object for Portal (MOP) Management Extension

The MOP management extension registers the 'mop' managed component. The managed component is responsible for managing pages, navigation, and site layout. It primarily supports exporting and importing the data through the **`export-resource`** and **`import-resource`** operations. It also supports the **`read-config-as-xml`** operation to expose the portal's metadata as XML.

Report a bug

4.1. Types of Operations

4.1.1. Understanding read-config-as-xml Operation

The **`read-config-as-xml`** operation can only be executed on the site layout, pages, and navigation managed resources. The XML format returned is defined by the **`gatein-objects`** XSD, which can be found at http://www.gatein.org/xml/ns/. These resources are exposed in the same format a portal extension would accept for importing data into the portal.

Report a bug

4.1.2. Understanding export-resource Operation

The **`export-resource`** operation can be executed on any resource of the MOP extension (including the **`mop`** component). Since the management system recursively searches for all sub-resources that have **`export-resource`** defined (which is defined at the site layout, page, and navigation level), exports can be very granular.

Report a bug

4.1.3. Understanding import-resource Operation

The **`import-resource`** operation can only be executed at the mop component (root managed resource of the mop extension). This is because the exported Zip file contains path information (like site type and site name). So executing an **`import-resource`** operation on a group site, when the Zip contains data from a portal site, does not make sense.

The MOP **`import-resource`** operation defines the **`importMode`** attribute as follows during import.

Table 4.1. importMode Attribute

importMode Attribute	Description
Conserve	Import data only if no artifacts exist for that site. For example, if one page exists for site *classic*, nothing will be imported.
Insert	Import data when data does not exist, otherwise do nothing.
Merge (default)	Import data when data does not exist, update data when it does exist.

importMode Attribute	Description
Overwrite	Delete all data for that artifact of the site, then import new data. For example, if the Zip file only contains pages for site *classic*, then all pages for that site are deleted and the contents of the Zip file imported.

Report a bug

4.2. Using Path Template Variables

The path template variables are defined in the MOP management extension. The path template variables are used for filtering during export operation

Path template variables

site-type

These are the site types of the portal to include or exclude. Available values are: **portal** , **group** , and **user**.

site-name

The sites to include or exclude. Examples could be **classic** and **/platform/administrators**.

site-layout

The name of the site layout depending on the site type. Available values are: **portal**, **group**, **user**.

page-name

The name of the page(s) to include or exclude.

nav-uri

The URI of the navigation node to include or exclude.

Report a bug

4.3. REST API Management

4.3.1. MOP Component Resource

The **mop** managed component resource (root managed resource) is the only resource that accepts the **import-resource** operation.

Example 4.1. HTTP Request

```
PUT /mop

Headers:
   Content-Type: application/zip
```

Example 4.2. HTTP Response

```
HTTP/1.1 200 OK
```

Report a bug

4.3.2. Site Layout Resource

The site layout resource represents the site layout of the portal. It is the data defined in the **portal.xml**, **group.xml** , and **user.xml** files (depending on site type) used in portal extensions to configure data.

Example 4.3. URL

```
URL: /mop/{site-type}sites/{site-name}/{site-layout}
```

Example 4.4. config-as-xml

Example of reading the site layout as XML for site *classic*.

Example 4.5. HTTP Request

```
GET /mop/portalsites/classic/portal.xml

or

GET /mop/portalsites/classic/portal?op=read-config-as-xml
```

Example 4.6. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<portal-config>
   <portal-name>classic</portal-name>
   <label>Classic</label>
   <description>GateIn default portal</description>
   <locale>en</locale>
...
</portal-config>
```

Example 4.7. export-resource

Example of exporting the site layout for site *classic*.

Example 4.8. HTTP Request

```
GET /mop/portalsites/classic/portal.zip

or

GET /mop/portalsites/classic/portal?op=export-resource
```

Example 4.9. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/zip

[binary data]
```

Report a bug

4.3.3. Page Resource

Pages Resource

The **pages** resource represents the pages of the portal. It is the data defined in the **pages.xml** used in portal extensions to configure data.

Example 4.10. URL

```
URL: /mop/{site-type}sites/{site-name}/pages/{page-name}
```

config-as-xml

Example of reading all pages as XML for site *classic*.

Example 4.11. HTTP Request

```
GET /mop/portalsites/classic/pages.xml

or

GET /mop/portalsites/classic/pages?op=read-config-as-xml
```

Example 4.12. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<page-set>
```

```
   <page>
      <name>homepage</name>
      <title>Home Page</title>
      <access-permissions>Everyone</access-permissions>
      <edit-permission>*:/platform/administrators</edit-permission>
      <show-max-window>false</show-max-window>
      <portlet-application>
...
</page-set>
```

Example of reading the homepage as XML for site *classic*.

Example 4.13. HTTP Request

```
GET /mop/portalsites/classic/pages/homepage.xml

or

GET /mop/portalsites/classic/pages/homepage?op=read-config-as-xml
```

Example 4.14. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<page-set>
   <page>
      <name>homepage</name>
      <title>Home Page</title>
      <access-permissions>Everyone</access-permissions>
      <edit-permission>*:/platform/administrators</edit-permission>
      <show-max-window>false</show-max-window>
      <portlet-application>
...
</page-set>
```

export-resource

Example of exporting all pages of site *classic*.

Example 4.15. HTTP Request

```
GET /mop/portalsites/classic/pages.zip

or

GET /mop/portalsites/classic/pages?op=export-resource
```

Example 4.16. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/zip

[binary data]
```

Report a bug

4.3.4. Navigation Resource

Navigation Resource

The navigation resource represents the navigation of the portal. It is the data defined in the **`navigation.xml`** used in portal extensions to configure data.

Example 4.17. URL

```
URL: /mop/{site-type}sites/{site-name}/navigation/{nav-uri}
```

Examples of Navigation resources

config-as-xml

Example of reading all navigation as XML for site *classic*.

Example 4.18. HTTP Request

```
GET /mop/portalsites/classic/navigation.xml

or

GET /mop/portalsites/classic/navigation?op=read-config-as-xml
```

Example 4.19. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<node-navigation>
   <priority>1</priority>
   <page-nodes>
      <node>
         <name>home</name>
         <label xml:lang="en">Home</label>
         ...
         <visibility>DISPLAYED</visibility>
         <page-reference>portal::classic::homepage</page-reference>
```

```
      </node>
      <node>
         <name>sitemap</name>
...
</node-navigation>
```

Example of reading the **home** node as XML for site *classic*.

Example 4.20. HTTP Request

```
GET /mop/portalsites/classic/navigation/home.xml

or

GET /mop/portalsites/classic/navigation/home?op=read-config-as-xml
```

Example 4.21. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/xml

<node-navigation>
   <priority>1</priority>
   <page-nodes>
      <parent-uri></parent-uri>
      <node>
         <name>home</name>
         <label xml:lang="en">Home</label>
         ...
         <visibility>DISPLAYED</visibility>
         <page-reference>portal::classic::homepage</page-reference>
      </node>
   </page-nodes>
</node-navigation>
```

export-resource

Example of exporting all navigation of site *classic*.

Example 4.22. HTTP Request

```
GET /mop/portalsites/classic/navigation.zip

or

GET /mop/portalsites/classic/navigation?op=export-resource
```

Example 4.23. HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/zip

[binary data]
```

Report a bug

4.3.5. Exporting and Filtering

Filtering is activated when the **`filter`** attribute is passed to an **`export-resource`** operation. The filter attribute is a multi-value attribute that is passed as request parameters of the HTTP request.

Note

You can either include multiple filter parameters (?filter=foo:bar&filter=baz:foo-bar) or separate via ';' character (?filter=foo:bar;baz:foo-bar)

Example 4.24. Export only registry and pageManagement navigation nodes

```
GET /mop/groupsites/platform/administrators/navigation.zip?filter=nav-uri:/administration/registry,/administration/pageManagement
```

Example 4.25. Export all site types but user and group

```
GET /mop.zip?filter=site-type:!user,group
```

Report a bug

4.4. Command Line Interface

The commands included in the management component provide us the tools to perform management operations on these MOP artifacts: site layout, pages, and navigation.

Report a bug

4.4.1. Resource Paths

The paths of the MOP resources are exactly the same as the REST URLs (without the URL syntax). For example the path of the homepage for the *classic* site would be:

Example 4.26. Path of homepage for the classic site

```
[ /]% cd /mop/portalsites/classic/pages/homepage

[homepage]% pwd
/mop/portalsites/classic/pages/homepage
```

Note

All resources/paths can be autocompleted by pressing the **Tab** key.

Report a bug

4.4.2. Exporting and Filtering

Filtering is activated when the **filter** attribute is passed to an **export-resource** operation. The filter attribute is a multi-value attribute that is passed to the CLI using the **--filter** attribute for the **export** command.

Example 4.27. Export all portal site types

```
export --file /tmp/mop.zip --filter site-type:portal /mop
```

Example 4.28. Export all sites types but user

```
export --file /tmp/mop.zip --filter site-type:!user /mop
```

The option can be specified multiple times for multiple values.

Example 4.29. Export only the /platform/administrators group site

```
export --file /tmp/mop.zip --filter site-type:group --filter site-
name:/platform/administrators /mop
```

Also as discussed in the *Path Templates* section, the filter attribute can separate different path templates by the **;** character.

Example 4.30. Export only pages named homepage, navigation named home for site *classic*

```
export --file /tmp/classic.zip --filter page-name:homepage;nav-uri:home
/mop/portalsites/classic
```

Important

All three artifacts (site layout, navigation, and pages) are included in export by default. In other words if you do not specify their path template in the filter, the data will be included.

Report a bug

4.5. Using Secure Copy Command

The SCP command can be used to export and import MOP artifacts: site layout, pages, and navigation over different hosts. The following examples assume the portal and CLI is running at host 'example.org' with default ssh port configured to 2000.

The **export-resource** command is supported on all resources of the MOP extension. Use the following examples as syntax guides to export artifacts based on the configuration of the production network.

Example 4.31. Export MOP artifacts

```
$ scp -P 2000 'root@example.org:portal:/mop/portalsites/classic'
classic.zip $ scp -P 2000
'root@example.org:portal:/mop/portalsites/classic/pages/homepage'
homepage.zip $ scp -P 2000 'root@example.org:portal:/mop' mop.zip
```

The **import-resource** operation is only supported at the **mop** resource, so for every import the path must be '/mop'.

Example 4.32. Import MOP artifacts

```
$ scp -P 2000 classic.zip 'root@example.org:portal:/mop' $ scp -P 2000
homepage.zip 'root@example.org:portal:/mop?importMode=overwrite' $ scp
-P 2000 mop.zip 'root@example.org:portal:/mop'
```

Report a bug

Part II. Domain Mode

Chapter 5. Configuring JBoss Portal Domain Mode

5.1. Deployment Notes for Domain Configurations

Portal applications and Java Enterprise Edition applications can be deployed using the domain mode hot-deploy mechanism. Containers and Extensions must be moved into the deployment directory, and will be loaded when JBoss Portal initially starts.

See Also:

- Section 5.2, “Domain Configuration Variables”

Report a bug

5.2. Domain Configuration Variables

JBoss EAP stores configuration elements relating to the application server itself in a separate directory location to JBoss Portal configuration files. The JBoss Portal configuration files are stored in the **JBOSS_HOME/standalone/configuration/gatein/** directory. Configuring JBoss Portal to operate in domain mode requires the location of portal-specific configuration to be specified for each instance in a domain.

JBoss Portal provides system property variables that allow portal-specific configuration file paths to be defined once, and reused as variables in other configuration.

exo.conf.dir

Specifies the directory that contains the eXo standalone configuration information the domain server will use.

gatein.conf.dir

Important

When this parameter is defined in system properties, it must be commented or deleted from the **configuration.properties** file. Issues with server start will occur if this recommendation is ignored.

Specifies the directory that contains the portal standalone configuration the domain server will use.

gatein.portlet.config

Specifies the directory that contains the portlet standalone configuration the domain server will use.

gatein.deploy.config

Specifies the directory that contains the main portal container **gatein.ear**. The default value is **JBOSS_HOME/gatein/**.

Important

Portal containers can not be deployed using the domain mode mechanism. Containers must be manually deployed by moving the container into the specified directory. JBoss Portal loads containers on initial server start through a custom service specific to JBoss Portal.

gatein.extensions.config

Important

Portal extensions can not be deployed using the domain mode mechanism. Extensions must be manually deployed by moving the container into the specified directory. JBoss Portal loads extensions on initial server start through a custom service specific to JBoss Portal.

Specifies the directory that contains any portal extensions. The default value is **`JBOSS_HOME/gatein/extensions`**.

See Also:

- Section 5.3, “Individual Domain Configuration”
- Section 5.4, “Shared Domain Configuration”

Report a bug

5.3. Individual Domain Configuration

When configuring domain nodes for a managed domain, the configuration variables can be defined using shared domain configuration, or isolated to each server instance.

It is necessary to create the directory structure for domain mode support manually, as JBoss Portal does not support this operation mode by default. The expected domain mode directory structure is **`JBOSS_HOME/domain/servers/myinstance/configuration/gatein`**.

Example 5.1. Defining per-server domain configuration

This example describes the system property configuration required in the **`host.xml`** file for each server instance.

```
<server name="server-one" group="main-server-group">
  <system-properties>
    <property name="exo.conf.dir"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.conf.dir"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.portlet.config"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
  </system-properties>
</server>
```

If a domain server instance is manually created using the JBoss EAP management console or the command line interface, the domain configuration variables must be added to each server configuration's system properties to allow the domain instance to start successfully. For a comprehensive overview of management console operations, see the *Management Interfaces* chapter in the JBoss EAP *Administration and Configuration Guide* available from https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Enterprise_Application_Platform/6.1/html-single/Administration_and_Configuration_Guide/index.html.

See Also:

- Section 5.1, “Deployment Notes for Domain Configurations”
- Section 5.2, “Domain Configuration Variables”

Report a bug

5.4. Shared Domain Configuration

When configuring domain nodes for a managed domain, the configuration variables can be defined using individual domain configuration, or shared amongst server groups.

It is necessary to create the directory structure for domain mode support manually, as JBoss Portal does not support this operation mode by default. The expected domain mode directory structure is **`JBOSS_HOME/domain/servers/myinstance/configuration/gatein`**.

Example 5.2. Defining shared domain configuration

This example describes the system property configuration required in the **`domain.xml`** file for the server group.

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <heap size="1303m" max-size="1303m"/>
    <permgen max-size="256m"/>
  </jvm>
  <socket-binding-group ref="full-sockets"/>
  <system-properties>
    <property name="exo.conf.dir"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.conf.dir"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.portlet.config"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
  </system-properties>
</server-group>
```

Defining a server group and adding the domain configuration parameters described in Example 5.2, “Defining shared domain configuration” is also possible from the JBoss EAP management interface or command-line interface. For a comprehensive overview of management console operations, see the *Management Interfaces* chapter in the JBoss EAP *Administration and Configuration Guide* available

from https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Enterprise_Application_Platform/6.1/html-single/Administration_and_Configuration_Guide/index.html.

Report a bug

5.5. Domain Mode Quickstart

To test the functionality of domain mode, a demonstration is available that features the following configuration highlights:

- **`domain.xml`**/**`host.xml`** configuration.
- Default and Full profiles.
- Single-host domain with two JBoss Portal instances.
- Shows how the **`gatein.extensions.dir`** configuration parameter is used to allow servers to coexist on the same host.

The demonstration establishes two instances of JBoss Portal that listen on different ports. The instance listening on port 8080 provides WSRP integration and mobile support, while the instance listening on port 8330 provides mobile support only.

One server instance uses the default extensions directory **`JPP_HOME/gatein/extensions`** and the second server instance uses an alternative extension location specified in the **`host.xml`** file. Both the servers coexist on the same host and a copy of the **`JPP_HOME/standalone/configuration/gatein`** directory is created for each server.

Task: Setup and Run the Domain Demonstration

Start the demonstration and use the management console to deploy a portlet application to both server instances.

1. In a terminal, navigate to **`JPP_HOME/bin/`**.
2. Run the demonstration setup script.

```
./demo-domain-setup.sh
```

3. Once the demonstration setup completes, run JBoss Portal in Domain mode to start the host controller.

```
./domain.sh
```

4. Open http://localhost:9990/console to verify the domain controller is running.

Report a bug

5.6. Multiple Node Domain Scenario

5.6.1. Environment Assumptions

This process describes how to configure a multi-host domain in detail. The machines provisioned as part of this process are described below. Familiarize yourself with the information, because it will be referenced through the configuration steps.

Machine_A

The domain controller, which has an IP address of **192.168.1.17**.

Machine_B

A host controller with JBoss Portal instances, which has an IP address of **192.168.100.10**.

Machine_C

A host controller with JBoss Portal instances, which has an IP address of **192.168.100.20**.

Next Step in Multiple Node Domain Scenario

- Section 5.6.2, “Configure the Domain Controller (Machine A)”

Report a bug

5.6.2. Configure the Domain Controller (Machine A)

Previous Step in Multiple Node Domain Scenario

- Section 5.6.1, “Environment Assumptions”

Configuring Machine_A

Configure and start the Machine_A domain controller.

1. Run **JPP_HOME/bin/add-user.sh** and create a user with the following parameters:
   - User name: **gateinhc**
   - Password: **gateinhc123.** (including the trailing full-stop)

   The secret value **Z2F0ZWluaGMxMjMu** created is used to allow host controller remote connections. Take note of this value.
2. Change the name of the machine in the **JPP_HOME/domain/configuration/host-master.xml** <host> element from **master** to **Machine_A**
3. In **domain.xml**, add the domain configuration parameters inside the main-server-group block.

```
<server-group name="main-server-group" profile="full"><server-group
name="main-server-group" profile="full">
  <jvm name="default">
    <heap size="1303m" max-size="1303m"/>
    <permgen max-size="256m"/>
  </jvm>
  <socket-binding-group ref="full-sockets"/>
  <system-properties>
    <property name="exo.conf.dir"
```

```
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.conf.dir"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
    <property name="gatein.portlet.config"
value="${jboss.home.dir}/standalone/configuration/gatein"/>
  </system-properties>
</server-group>
```

4. Run the following command to start the domain controller on Machine_A.

   JBOSS_HOME/bin/domain.sh --domain-config=domain.xml --host-config=host-master.xml -Djboss.bind.address.management=192.168.1.17

Next Step in Multiple Node Domain Scenario

- Section 5.6.3, “Configure Host Controller One (Machine B)”

Report a bug

5.6.3. Configure Host Controller One (Machine B)

Previous Step in Multiple Node Domain Scenario

- Section 5.6.2, “Configure the Domain Controller (Machine A)”

Configuring Host Controller One (Machine B)

Configure and start the Machine_B host controller.

1. Change the name of the machine in the **JPP_HOME/domain/configuration/host-slave.xml** <host> element from slave to **Machine_B**.

2. Update the <server-identities> block with the secret value created in Section 5.6.2, “Configure the Domain Controller (Machine A)” to permit the host machine to connect to the domain controller.

```
<host name="Machine_B" xmlns="urn:jboss:domain:1.4">

    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <server-identities>
                    <secret value="Z2F0ZWluaGMxMjMu"/>
                </server-identities>
```

3. Add the domain controller user name to the **host-slave.xml** <domain-controller> block.

```
<domain-controller>
  <remote host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}" username="gateinhc"
security-realm="ManagementRealm"/>
</domain-controller>
```

4. In the **JPP_HOME/standalone/configuration/gatein/configuration.properties** file for **Machine_B**, comment or delete the **gatein.conf.dir** property.

5. Start **Machine_B** with the following command:

```
bin/domain.sh --host-config=host-slave.xml -
Djboss.bind.address=192.168.100.10 -
Djboss.bind.address.management=192.168.100.10 -
Djboss.domain.master.address=192.168.1.17
```

Report a bug

5.6.4. Configure Host Controller Two (Machine C)

Configuring Host Controller Two (Machine C)

Configure and start the Machine_C host controller.

Completing this task provisions the last host controller, after which the domain is ready for use.

1. Change the name of the machine in the **JPP_HOME/domain/configuration/host-slave.xml** <host> element from slave to **Machine_C**.

2. Update the <server-identities> block with the secret value created in Section 5.6.2, “Configure the Domain Controller (Machine A)” to permit the host machine to connect to the domain controller.

```
<host name="Machine_B" xmlns="urn:jboss:domain:1.4">

    <management>
        <security-realms>
            <security-realm name="ManagementRealm">
                <server-identities>
                    <secret value="Z2F0ZWluaGMxMjMu"/>
                </server-identities>
```

3. Add the domain controller user name to the **host-slave.xml** <domain-controller> block.

```
<domain-controller>
  <remote host="${jboss.domain.master.address}"
port="${jboss.domain.master.port:9999}" username="gateinhc"
security-realm="ManagementRealm"/>
</domain-controller>
```

4. In the **JPP_HOME/standalone/configuration/gatein/configuration.properties** file for **Machine_C**, comment or delete the **gatein.conf.dir** property.

5. Start **Machine_C** with the following command:

```
bin/domain.sh --host-config=host-slave.xml -
Djboss.bind.address=192.168.100.20 -
Djboss.bind.address.management=192.168.100.20 -
Djboss.domain.master.address=192.168.1.17
```

Report a bug

Part III. Administration and Monitoring

Chapter 6. JBoss Operations Network (JON) Plug-in

The GateIn JON plug-in itself does not provide any operations; it's sole purpose is reporting and statistics relating to applications/portlets and sites. The JBoss Operations Network AS plug-in supports many management operations, and is included in the portal binaries. With the AS plug-in, the following non-exhaustive list of actions are possible:

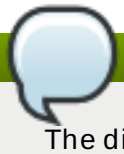
- Server control: start, stop, restart.
- Application control: deploy, and remove EAR and WAR applications.
- Resource control: manage other components of the platform, such as data sources.

For detailed information about JBoss Operations Network, and how it can be used to manage platforms, see the *How to Manage JBoss Servers* guide, chapter *JBoss Operations Network* available from https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Operations_Network/.

Report a bug

6.1. Metrics Collected by JON Plug-in

The metrics collected by the GateIn JON plug-in include site and application/portlet statistics. Each portlet contains statistics relevant to the individual portlet, and one additional throughput statistic for site portlets. See, table *Site and Portlet Statistics* for a description of the available metrics.



Note

The discovery of sub-resources such as Sites and Portlet Applications is performed once a day by JON. Hence, it is normal that new Sites and Applications do not appear immediately in JON, unless one triggers a manual discovery.

The manual discovery can be triggered in the JON UI, under the top level server node for example, **`"localhost.localdomain" > Operations > Schedules > New > Operation: "Run Autodiscovery", Detailed Discovery: yes.`**

The metrics and statistics are runtime only values, and do not persist on the server. When the server is restarted, the runtime values are reset. JBoss Operations Network keeps a history of these statistics and metrics to give an accurate view of the data over time.

Table 6.1. Site and Portlet Statistics

Metric	Description
Average Execution Time	The average execution time in seconds for a page or application/portlet to render within a site.
Execution Count	The amount of times the application/portlet or site has been accessed.
Maximum Execution Time	The maximum execution time in seconds for a page or application/portlet to render within a site.

Metric	Description
Minimum Execution Time	The minimum execution time in seconds for a page or application/portlet to render within a site.
Throughput (only for site portlets)	The number of requests per second for a site.

Report a bug

6.2. About JBoss Operation Network Plug-in

The portal provides a JBoss Operations Network plug-in (*GateIn JON Plug-in*) to assist with monitoring the platform.

The plug-in captures application/portlet and site statistics. A different set of statistics are collected depending on the context of each portlet. The basic JON interface is shown in Figure 6.1, “GateIn JON Plug-in Interface”.

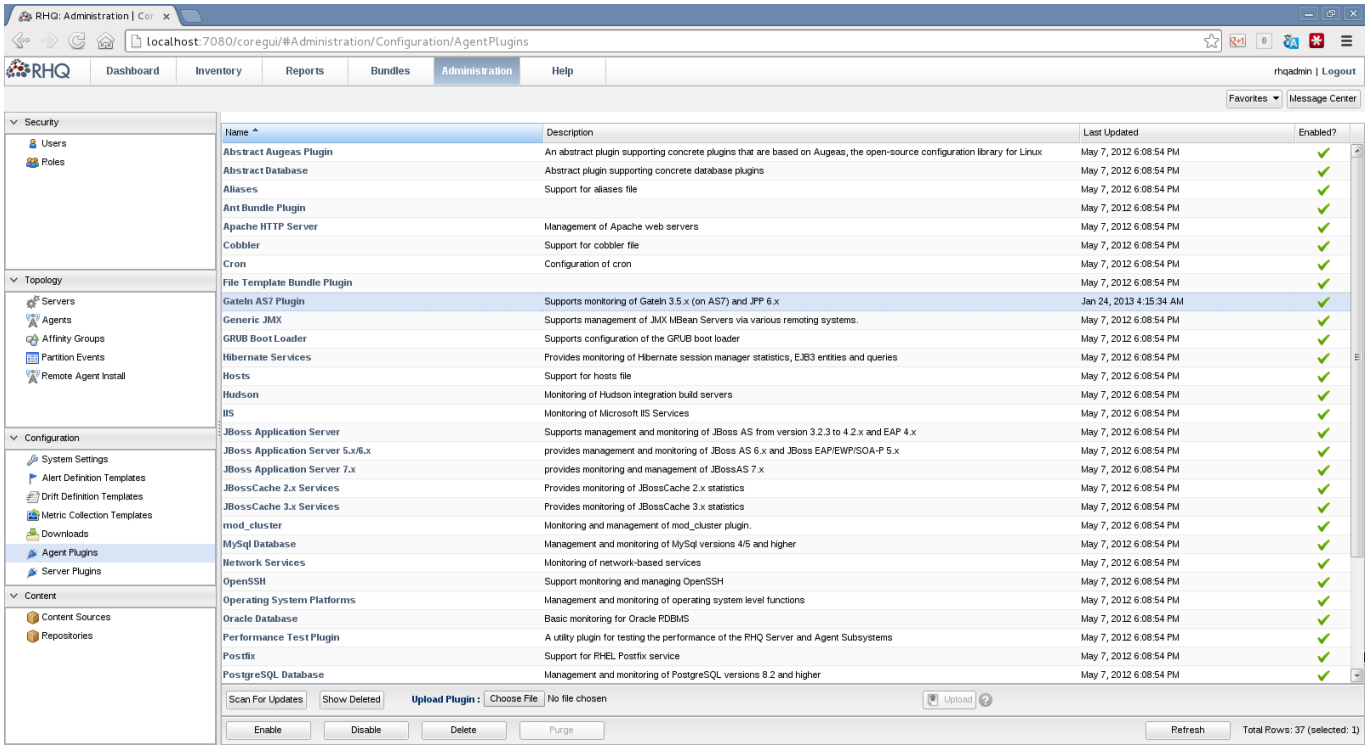


Figure 6.1. GateIn JON Plug-in Interface

Follow the download and installation instructions in the *Installation Guide* to activate administration and monitoring.

Report a bug

Part IV. Authentication and Authorization

Chapter 7. Authentication and Authorization

Authentication in the portal is based on JAAS and by default it is a standard J2EE FORM based authentication.

Report a bug

7.1. Authentication Methods

The portal supports the following authentication methods:

- J2EE FORM based authentication
- The **RememberMe** authentication method (wherein the user checks the **Remember my login** check box on the log in form).
- SSO server integration (CAS, JOSSO, OpenSSO).
- SPNEGO authentication with a Kerberos ticket.
- SAML2 based authentication.
- Cluster authentication with load balancer or with JBoss SSO valve.

See Also:

- Section 12.2, “Single Sign-on (SSO) Configuration”
- Chapter 15, *Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO)*
- Chapter 18, *Security Assertion Markup Language (SAML2)*
- Chapter 16, *Single Sign-on in a Cluster*

Report a bug

7.2. Authentication Workflow

Authentication workflow consists of HTTP requests and redirects which include handshakes. Currently only Servlet 3.0 containers are supported, so authentication is triggered programmatically from Servlet API.

In ***JPP_DIST*/gatein/gatein.ear/portal.war/WEB-INF/web.xml**, authentication is triggered by accessing a secured URL **/dologin**:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>user authentication</web-resource-name>
      <url-pattern>/dologin</url-pattern>
        <http-method>POST</http-method>
        <http-method>GET</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-name>users</role-name>
    </auth-constraint>
```

```
    <user-data-constraint>
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

This means that access to URLs (such as http://localhost:8080/portal/dologin) will directly trigger J2EE authentication in the case that the user is not already logged in.

Access to this URL means that the user needs to be in the JAAS group *users*, otherwise they can authenticate but will receive an HTTP error: *403 Forbidden*.

In the next part of the file we can see that authentication is FORM based and it starts by redirection to */login* URL, which is mapped to **`LoginServlet`**.

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>gatein-domain</realm-name>
  <form-login-config>
    <form-login-page>/login</form-login-page>
    <form-error-page>/login</form-error-page>
  </form-login-config>
</login-config>
```

`LoginServlet` redirects the user to the login page placed in ***`JPP_DIST`*`/gatein/gatein.ear/portal.war/login/jsp/login.jsp`**.

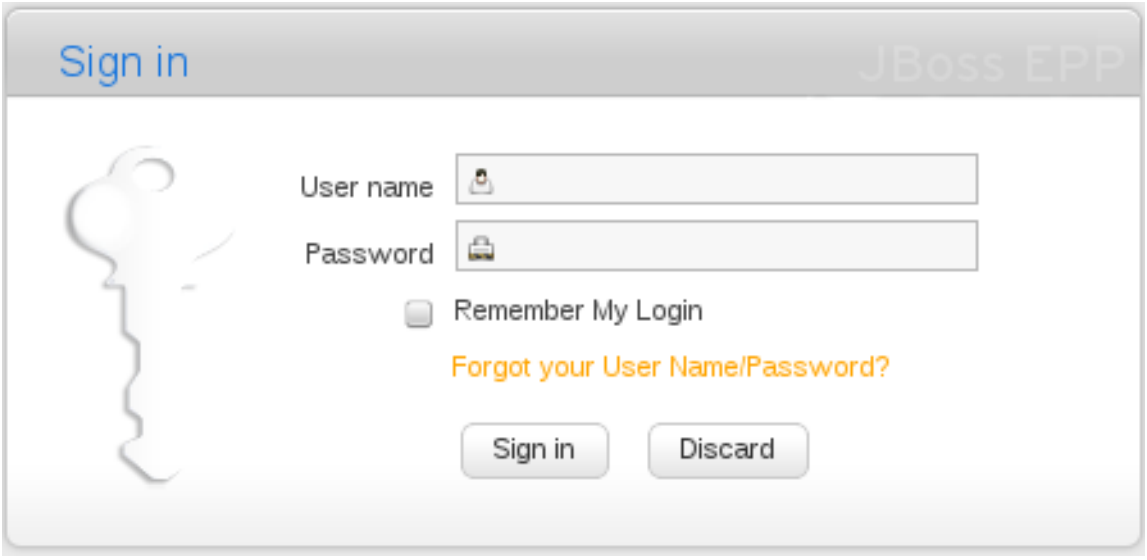


Figure 7.1. Default Login Form on the login.jsp Page

Changes to the appearance of the login page can be made in this JSP file. Alternatively you can create an extension and override this page via extension if you don't want to edit it directly. You can also change images or CSS placed in ***`JPP_DIST`*`/gatein/gatein.ear/login/skin`**.

After a user submits the login form, the LoginServlet will store credentials and trigger WCI login, which delegates to Servlet API (method HttpServletRequest.login(String username, String password) available in Servlet 3.0) and additionally triggers WCI Authentication listeners. Login through Servlet API delegates to JAAS.

Report a bug

7.2.1. RememberMe Authentication

In the default login dialog is the Remember my login checkbox, which persist a user's login on their workstation. The default validity period of the **RememberMe** cookie is one day, so a user can be logged for a whole day before needing to re-authenticate. The validity period is configurable.

The way the RememberMe authentication operates is described below:

- User checks the checkbox **RememberMe** on the login screen of the portal, then submits the form.
- Form data such as **username**, **password** and **rememberme** are sent in an HTTP POST request to the **/portal/login** URL, with the **rememberme** parameter set to true.
- Request is processed by PortalLoginController servlet. The servlet obtains an instance of *RemindPasswordTokenService* and saves user credentials into JCR. It generates and returns special token (key) for later use, then creates a cookie called *RememberMe* and uses the returned token as the cookie's value.

Report a bug

7.2.2. Re-authentication

- After some time, the user wants to re-authenticate. It is assumed that his HTTP Session is already expired but his RememberMe cookie is still active.
- The user sends the HTTP request to some portal pages (for example, http://localhost:8080/portal/classic).
- There is a special HTTP Filter RememberMeFilter configured in **web.xml**, which checks the RememberMe cookie and then it retrieves credentials of user from RemindPasswordTokenService. Now filter redirects request to PortalLoginController and authentication process goes in same way as for normal FORM based authentication.

Report a bug

7.2.3. RemindPasswordToken Service

This is a special service used during the RememberMe authentication workflow. It is configurable in the file ***JPP_DIST*/gatein/gatein.ear/portal.war/WEB-INF/conf/common/remindpwd-configuration.xml**.

You can also encrypt passwords before storing them in JCR.

See Also:

- Chapter 10, *Token Service*
- Section 8.1, “Hashing and Salting of Passwords in PicketLink IDM”

Report a bug

7.3. Login Modules

From the WCI servlet API login, the user is redirected to JAAS authentication. The portal uses its own security domain (**gatein-domain**) with a set of predefined login modules. Login module configuration for *gatein-domain* is contained in the **JPP_HOME/standalone/configuration/standalone.xml** file.

Below is the default login modules stack:

```
<security-domain name="gatein-domain" cache-type="default">
  <authentication>
    <login-module
code="org.gatein.sso.integration.SSODelegateLoginModule" flag="required">
      <module-option name="enabled"
value="${gatein.sso.login.module.enabled}" />
      <module-option name="delegateClassName"
value="${gatein.sso.login.module.class}" />
      <module-option name="portalContainerName" value="portal" />
      <module-option name="realmName" value="gatein-domain" />
      <module-option name="password-stacking" value="useFirstPass" />
    </login-module>
    <login-module
code="org.exoplatform.services.security.j2ee.JBossAS7LoginModule"
flag="required">
      <module-option name="portalContainerName" value="portal"/>
      <module-option name="realmName" value="gatein-domain"/>
    </login-module>
  </authentication>
</security-domain>
```

New login modules can be added or the stack completely replaced with custom modules.

Authentication starts with the login method of each login module being invoked. After all login methods are invoked, the authentication is continued by invoking the commit method on each login module. Both login and commit methods can throw LoginException. If it happens, then the whole authentication ends unsuccessfully, which in turn invokes the abort method on each login module. By returning "false" from the login method, you can ensure that the login module is ignored. This is not specific to the portal, but it is generic to JAAS. See http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html for more information about login modules in general.

Report a bug

7.3.1. Types of Login Modules

The existing login modules include SSODelegateLoginModule, JBossAS7LoginModule, CustomMembershipLoginModule, InitSharedStateLoginModule,and SharedStateLoginModule.

Existing Login Modules

SSODelegateLoginModule

It is useful only if SSO authentication is enabled (disabled by default). SSO authentication can be enabled through properties in **`configuration.properties`** file and it delegates the work to another real login module for SSO integration. If SSO is disabled, **`SSODelegateLoginModule`** is ignored. If SSO is used and SSO authentication succeeds, the special Identity object is created and saved into shared state map (Map, which is shared between all login modules), so that this Identity object can be used by JBossAS7LoginModule or other login modules in the JAAS chain.

JBossAS7LoginModule

The most important login module, which is normally used to perform the whole

authentication by itself. First it checks if an Identity object has been already created and saved into the sharedState map by previous login modules (like SSODelegateLoginModule, CustomMembershipLoginModule or SharedStateLoginModule). If not, it triggers real authentication of the user with usage of the Authenticator interface and it will use **`Authentication.validateUser(Credential[] credentials)`** which performs real authentication of user names and passwords against OrganizationService and the portal identity database.

In the **`JBossAS7LoginModule.commit`** method, the Identity object is registered to IdentityRegistry, which will be used later for authorization. Also some JAAS principals (UserPrincipal and RolesPrincipal) are assigned to our authenticated Subject. This is needed for JBoss Enterprise Application server, so that it can properly recognize the name of the logged user and its roles on an application server level.

InitSharedStateLoginModule

It can read credentials from JAAS callback and add them into shared state. It's intended to be used in JAAS chain before SharedStateLoginModule

SharedStateLoginModule

It reads the username and password from sharedState map from attributes **`javax.security.auth.login.name`** and **`javax.security.auth.login.password`**. Then it calls Authenticator.validateUser(Credential[] credentials), to perform authentication of username and password against OrganizationService and portal identity database. The result of successful authentication is an Identity object, which is saved to the sharedState map.

See Also:

- Section 12.3, “Central Authentication Service (CAS)”
- Section 7.3.6, “Authenticator Interface”
- Section 7.3.7, “RolesExtractor Interface”

Report a bug

7.3.2. About Custommembership Login Module

Custommembership login module is a special login module, that can be used to add a user to existing groups during a successful login of this user. The group name is configurable and by default is /platform/users group. This login module is not used because in normal environment, users are already in the /platform/users group. It is useful only for some special setups like read-only LDAP, where groups of an LDAP user are taken from the LDAP tree so that users may not be in the /platform/users group, which is needed for successful authorization.

Note

The CustomMembershipLoginModule cannot be the first login module in the LoginModule chain because it assumes that the Identity object is already available in the shared state. So there are two possible cases. For a non-SSO case, you may need to chain this login module with other login modules, which can be used to establish Identity and add it into shared state. Those login modules can be **`InitSharedStateLoginModule`** and **`SharedStateLoginModule`**. For an SSO case, you can add **`CustomMembershipLoginModule`** between **`SSODelegateLoginModule`** and **`JBossAS7LoginModule`**.

Report a bug

7.3.3. Configuring Custommembership Login Module

To configure Customermembership Login module with SSO disabled and SSO enabled is shown as example in this topic.

Example 7.1. CustomMembershipLoginModule and Disabled SSO

```
 <login-module
code="org.exoplatform.web.security.InitSharedStateLoginModule"
flag="required">
    <module-option name="portalContainerName" value="portal"/>
    <module-option name="realmName" value="gatein-domain"/>
 </login-module>
 <login-module
code="org.exoplatform.services.security.jaas.SharedStateLoginModule"
flag="required">
    <module-option name="portalContainerName" value="portal"/>
    <module-option name="realmName" value="gatein-domain"/>
 </login-module>
 <login-module
code="org.exoplatform.services.organization.idm.CustomMembershipLoginMo
dule" flag="required">
   <module-option name="portalContainerName" value="portal"/>
   <module-option name="realmName" value="gatein-domain"/>
   <module-option name="membershipType" value="member" />
   <module-option name="groupId" value="/platform/users" />
 </login-module>
 <login-module
code="org.exoplatform.services.security.j2ee.JBossAS7LoginModule"
flag="required">
   <module-option name="portalContainerName" value="portal"/>
   <module-option name="realmName" value="gatein-domain"/>
 </login-module>
```

Example 7.2. Configuration with SSO Enabled

```
 <login-module
code="org.gatein.sso.integration.SSODelegateLoginModule"
flag="required">
  <module-option name="enabled"
value="${gatein.sso.login.module.enabled}" />
  <module-option name="delegateClassName"
value="${gatein.sso.login.module.class}" />
  <module-option name="portalContainerName" value="portal" />
  <module-option name="realmName" value="gatein-domain" />
  <module-option name="password-stacking" value="useFirstPass" />
</login-module>
<login-module
code="org.exoplatform.services.organization.idm.CustomMembershipLoginMo
dule" flag="required">
  <module-option name="portalContainerName" value="portal"/>
  <module-option name="realmName" value="gatein-domain"/>
  <module-option name="membershipType" value="member" />
  <module-option name="groupId" value="/platform/users" />
</login-module>
<login-module
code="org.exoplatform.services.security.j2ee.JBossAS7LoginModule"
flag="required">
  <module-option name="portalContainerName" value="portal"/>
  <module-option name="realmName" value="gatein-domain"/>
</login-module>
```

Report a bug

7.3.4. Creating a Login Module

Before creating your own login module, it is recommended that you study the source code of existing login modules to better understand the JAAS authentication process. You need to have good knowledge so that you can properly decide where your login module should be placed and if you need to replace some existing login modules or simply attach your own module to the existing chain.

Report a bug

7.3.5. Levels of Authentication

There are two levels of authentication available. The overall result of JAAS authentication handles both cases:

Authentication on Application Server Level

The Application server must properly recognize that the user is successfully logged and it has their JAAS roles assigned. Unfortunately this part is not standardized and is specific for each application server. For example in Red Hat JBoss Enterprise Application Platform, you need to ensure that JAAS Subject has an assigned UserPrincipal with user name and also a RolesPrincipal, which contains a list of JAAS roles. This part is actually done in **`JBossAS7LoginModule.commit()`**.

After successful authentication, the user must be at least in JAAS role **`users`** because this role is declared in **`web.xml`** as you saw above. JAAS roles are extracted RolesExtractor from portal memberships.

Authentication on the Portal Level

The login process creates a special object **`org.exoplatform.services.security.Identity`** and registers this object into the portal component **`IdentityRegistry`**. The Identity object encapsulates the user name of the authenticated user, memberships of this user, and JAAS roles. The Identity object can be created with the **`Authenticator`** interface.

See Also:

- Section 7.3.7, “RolesExtractor Interface”

Report a bug

7.3.6. Authenticator Interface

Authenticator is an important component in the authentication process. The interface *org.exoplatform.services.security.Authenticator* looks like this:

```
public interface Authenticator
{

   /**
    * Authenticate user and return userId.
    *
    * @param credentials - list of users credentials (such as
name/password, X509
    *          certificate etc)
    * @return userId
    */
   String validateUser(Credential[] credentials) throws LoginException,
Exception;

   /**
    * @param userId.
    * @return Identity
    */
   Identity createIdentity(String userId) throws Exception;

}
```

Method *validateUser* is used to authenticate the given credentials (username and password). It returns a username if the credentials are correct, otherwise a **`LoginException`** is thrown.

Method *createIdentity* is used to create an instance of object *org.exoplatform.services.security.Identity*, which encapsulates all important information about a single user:

- Username
- Set of Memberships (MembershipEntry objects) associated with the **`user`**. *Membership* is an object, which contains information about *membershipType* (manager, member, validator, ...) and about *group* (/platform/users, /platform/administrators, /partners, /organization/management/executiveBoard, ...).

- Set of Strings with JAAS roles of given user. JAAS roles are simple Strings, which are mapped from MembershipEntry objects. There is another special component *org.exoplatform.services.security.RolesExtractor*, which is used to map JAAS roles from MembershipEntry objects.

Report a bug

7.3.7. RolesExtractor Interface

RolesExtractor interface looks like this:

```
public interface RolesExtractor
{

   /**
    * Extracts J2EE roles from userId and|or groups the user belongs to
both
    * parameters may be null
    *
    * @param userId
    * @param memberships
    */
   Set<String> extractRoles(String userId, Set<MembershipEntry>
memberships);
}
```

Default implementation *DefaultRolesExtractorImpl* is based on a special algorithm, which uses the name of the role from the root of the group (for example for role "/organization/management/something" we have JAAS role "organization"). The only exception is the "platform" group where we use second level as the name of the group. For example from group "/platform/users" we have JAAS role "users".

For example, the user root has the following memberships:

- member:/platform/users
- manager:/platform/administrators
- validator:/platform/managers
- member:/partners
- member:/customers/acme
- member:/organization/management/board

In this case we will have JAAS roles: users, administrators, managers, partners, customers, organization.

The default implementation of Authenticator is *OrganizationAuthenticatorImpl*, which is implementation based on *OrganizationService*. See the API documentation in the *Development Guide* for further information.

You can override the default implementation of the mentioned **`Authenticator`** and **`RolesExtractor`** interfaces if the default behavior is not suitable for your needs.

Report a bug

7.4. Authorization

Authorization in the portal happens on two levels:

- Servlet Container Authorization
- Portal Authorization

Report a bug

7.4.1. Servlet Container Authorization

The first round of authorization is through the servlet container authorization based on the secured URL from **web.xml**. The **web.xml** code sample shows that secured URLs are accessible only for users with the *users* role.

```
<auth-constraint>
  <role-name>users</role-name>
</auth-constraint>
```

This means a user must be in the portal role **/platform/users**.

If the user is successfully authenticated, but is not in the **/platform/users** group, the user will not be assigned a correct JAAS role **users**. This will result in an error 403 Forbidden thrown by the servlet container. For example in LDAP setup, users may not be in **/platform/users** by default, but **CustomMembershipLoginModule** can fix this problem.

Important

It is possible to change the behavior and add more **auth-constraint** elements into **web.xml**, however this protection of resources based on **web.xml** is not a standard portal method and is mentioned here mainly for illustration purposes.

See Also:

- Section 7.3.6, “Authenticator Interface”
- Section 7.3.7, “RolesExtractor Interface”
- Section 7.3, “Login Modules”

Report a bug

7.4.2. Portal Authorization

The second round of authorization is based on the component **UserACL**. You can declare access, and edit permissions for portals, pages and/or portlets. UserACL is then used to check if our user has particular permissions to access or edit specified resources. An important object containing information about the roles of our user is the *Identity* object created during JAAS authentication.

Authorization on portal level looks like this:

- The user sends a HTTP request to some URLs in portal;
- The HTTP request is processed through **`SetCurrentIdentityFilter`**, which is declared in ***`JPP_DIST`*`/gatein/gatein.ear/portal.war/WEB-INF/web.xml`**.
- SetCurrentIdentityFilter reads the user name of current user from *HttpServletRequest.getRemoteUser()*. Then it looks for Identity of this user in IdentityRegistry, where Identity has been saved during authentication. The Identity found is then encapsulated into a **`ConversationState`** object and bound into the ThreadLocal variable.
- UserACL is able to obtain Identity of current user from method *UserACL.getIdentity()*, which calls *ConversationState.getCurrent().getIdentity()* to find the current Identity bound to ThreadLocal. Now the UserACL has the identity of the user and can perform any security checks.

See Also:

- Section 25.4.1, “Setting up Default Configuration for JBoss Portal”

Report a bug

Chapter 8. Password Encryption using PicketLink IDM Framework

8.1. Hashing and Salting of Passwords in PicketLink IDM

The portal uses PicketLink IDM framework (http://www.jboss.org/picketlink/IDM) to store information about identity objects (users/groups/memberships). For better security, PicketLink IDM does not save user passwords into database in plaintext, but it uses **`CredentialEncoder`**, which encodes password and saves the encoded form into PicketLink IDM database. Later when the user wants to authenticate, they need to provide their password in plaintext through a web login form. The provided password is then encoded and compared to an encoded password in the PicketLink IDM database. The portal is then able to authenticate the user based on this comparison.

Report a bug

8.2. Implementing Credential Encoder

8.2.1. Default Implementation of CredentialEncoder

Default implementation of CredentialEncoder is DatabaseReadingSaltEncoder. It uses password hashing with a SHA-256 algorithm, and salting with random salts. Those salted hashes are then stored in the database.

Warning

The default DatabaseReadingSaltEncoder is safe solution, but it is not backward compatible with previous releases of the portal before v6.0. If migrating from older release of the portal, switch to HashingEncoder and configure it. See the *Migration Guide* for more info.

If starting from fresh database (no migration from a previous portal release), it is sufficient to keep the default choice provided by DatabaseReadingSaltEncoder, or possibly switch to FileReadingSaltEncoder.

See Also:

- Section 8.2.2, “Choosing CredentialEncoder Implementation”

Report a bug

8.2.2. Choosing CredentialEncoder Implementation

The implementation of CredentialEncoder is configured in Picketlink IDM configuration file ***`JPP_HOME`*`/gatein/gatein.ear/portal.war/WEB-INF/conf/organization/picketlink-idm/picketlink-idm-config.xml`**.

Usually the most important are options of realm idm_portal starting with prefix credentialEncoder. Possible implementations are HashingEncoder, DatabaseReadingSaltEncoder, and FileReadingSaltEncoder.

Note

If you are using LDAP, the location of Picketlink IDM configuration file can be different.

See Also:

- Section 9.1, “Introduction to PicketLink IDM”
- Chapter 17, *LDAP Integration*

Report a bug

8.2.3. Configuring Hashing Encoder

This implementation uses only hashing of passwords with MD5 algorithm without salting. It is not the safest solution but it is backward compatible with the previous portal releases, so there are no issues with database migration. The configuration is as follows:

Example 8.1. Configuration of Hashing Encoder

```
<option>
  <name>credentialEncoder.class</name>
  <value>org.picketlink.idm.impl.credential.HashingEncoder</value>
</option>
<option>
  <name>credentialEncoder.hashAlgorithm</name>
  <value>MD5</value>
</option>
```

Report a bug

8.2.4. Configuring DatabaseReadingSaltEncoder

This is the default choice. This implementation provides salting of password in addition to hashing. The salt is unique for each user, so it is much more complicated to decrypt password via brute force, if an attacker steals encoded passwords from your database. The salt is generated randomly for each user and stored in the PicketLink IDM database as an attribute. Random generation of salt ensures that all users have different salts, so even if two users have the same password, the encoded password in database will be different for them. Here is configuration example, which is using SHA-256 algorithm for hashing (more secure than MD5) and algorithm SHA1PRNG for generation of random salts.

```
<option>
  <name>credentialEncoder.class</name>

<value>org.picketlink.idm.impl.credential.DatabaseReadingSaltEncoder</va
lue>
</option>
<option>
```

```
  <name>credentialEncoder.hashAlgorithm</name>
  <value>SHA-256</value>
</option>
<option>
  <name>credentialEncoder.secureRandomAlgorithm</name>
  <value>SHA1PRNG</value>
</option>
```

Report a bug

8.2.5. Configuring FileReadingSaltEncoder

The FileReadingSaltEncoder also uses hashing and salting, so it is similar to the previous encoder. Theoretically it is even more secure, because salts are not stored in the PicketLink IDM database together with passwords. Salt for each user is generated from **`saltPrefix`** and the user's user name. **`saltPrefix`** is read from a file on the host computer.

Example 8.2. FileReadingSaltEncoder

This example describes a possible configuration of FileReadingSaltEncoder.

```
<option>
  <name>credentialEncoder.class</name>

<value>org.picketlink.idm.impl.credential.FileReadingSaltEncoder</valu
e>
</option>
<option>
  <name>credentialEncoder.hashAlgorithm</name>
  <value>SHA-256</value>
</option>
<option>
  <name>credentialEncoder.fileLocation</name>
  <value>/salt/mysalt.txt</value>
</option>
```

Important

The **`CredentialEncoder`** in Example 8.2, “FileReadingSaltEncoder” is actually used only for encoding of passwords in PicketLink IDM database. It is not used for LDAP. PicketLink IDM LDAP implementation (**`LDAPIdentityStore`**) sends passwords to the LDAP server in plain form, because password encoding is usually provided by LDAP server itself. For example OpenDS 2 uses SHA1 based hashing of passwords with random generation of user salt (quite similar to the **`DatabaseReadingSaltEncoder`** implementation).

The **`FileReadingSaltEncoder`**is arguably the most secure of all options, but in addition to **`DatabaseReadingSaltEncoder`**, you need to set the file with salt.

The **/salt/mysalt.txt** file must exist and must be readable by the user who starts the portal. The file must be properly secured so that it is not readable by every user. The content of the file can be a random phrase, such as **a4564dac2aasddsklklkajdgnioiow**.

Report a bug

8.2.6. Migration of Credential Encoder

As mentioned previously, our current default implementation DatabaseReadingSaltEncoder is not compatible with previous releases of GateIn Portal. In GateIn Portal 3.5 and older, we used MD5 hashing without salts. So to be backwards compatible, you will need to switch from default implementation of DatabaseReadingSaltEncoder and configure HashingEncoder with MD5 hashes. Without this change, your users cannot login into GateIn Portal because their passwords are stored in Picketlink IDM database as MD5 hashes, but current default encoder is using SHA-256 and salts, so comparisons will be different.

We decided to do this change and switch default algorithm, because MD5 is nowadays not treated as very secure solution.

See Also:

- Section 8.2.3, “Configuring Hashing Encoder”

Report a bug

Chapter 9. PicketLink IDM Integration

9.1. Introduction to PicketLink IDM

The portal uses the **PicketLink IDM** component to store necessary identity information about users, groups and memberships. While legacy interfaces are still used (**org.exoplatform.services.organization**) for identity management, there is a wrapper implementation that delegates to PicketLink IDM framework.

This section provides basic information about **PicketLink IDM** and its configuration.

Note

It is important to fully understand the concepts behind this framework design before changing the default configuration.

The identity models represented in the **org.exoplatform.services.organization** interfaces and the one used in PicketLink IDM have some major differences.

For example; **PicketLink IDM** provides greater abstraction. It is possible for groups in the IDM framework to form memberships with many parents (which requires recursive ID translation), while the **org.exoplatform.services.organization** model allows only pure tree-like membership structures.

Additionally, the Membership concept must be translated into the IDM Role concept. Therefore **PicketLink IDM** model is used in a limited way. All these translations are applied by the integration layer.

Report a bug

9.2. Configuring Picketlink IDM

The main configuration file is ***JPP_HOME*/gatein/gatein.ear/portal.war/WEB-INF/conf/organization/idm-configuration.xml** :

```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd
http://www.exoplaform.org/xml/ns/kernel_1_2.xsd"
               xmlns="http://www.exoplaform.org/xml/ns/kernel_1_2.xsd">

   <component>

<key>org.exoplatform.services.organization.idm.PicketLinkIDMService</key>

<type>org.exoplatform.services.organization.idm.PicketLinkIDMServiceImpl<
/type>
      <init-params>
         <value-param>
            <name>config</name>
            <value>war:/conf/organization/idm-config.xml</value>
         </value-param>
```

```
          <value-param>
             <name>portalRealm</name>
             <value>realm${container.name.suffix}</value>
          </value-param>
        </init-params>
    </component>


    <component>

<key>org.exoplatform.services.organization.OrganizationService</key>

<type>org.exoplatform.services.organization.idm.PicketLinkIDMOrganization
ServiceImpl</type>
       <init-params>
       <object-param>
         <name>configuration</name>
         <object type="org.exoplatform.services.organization.idm.Config">
           <field name="useParentIdAsGroupType">
             <boolean>true</boolean>
           </field>

           <field name="forceMembershipOfMappedTypes">
             <boolean>true</boolean>
           </field>

           <field name="pathSeparator">
             <string>.</string>
           </field>

           <field name="rootGroupName">
             <string>GTN_ROOT_GROUP</string>
           </field>

           <field name="groupTypeMappings">
             <map type="java.util.HashMap">
               <entry>
                 <key><string>/</string></key>
                 <value><string>root_type</string></value>
               </entry>

               <!-- Sample mapping -->
               <!--
               <entry>
                 <key><string>/platform/*</string></key>
                 <value><string>platform_type</string></value>
               </entry>
               <entry>
                 <key><string>/organization/*</string></key>
                 <value><string>organization_type</string></value>
               </entry>
               -->

             </map>
           </field>
```

```
          <field name="associationMembershipType">
            <string>member</string>
          </field>

          <field name="ignoreMappedMembershipType">
            <boolean>false</boolean>
          </field>
        </object>
      </object-param>
    </init-params>


  </component>

</configuration>
```

Report a bug

9.2.1. PicketlinkIDmServiceImpl Service

The **`org.exoplatform.services.organization.idm.PicketLinkIDMServiceImpl`** service has the following options:

- **`config`** (value-param) The PicketLink IDM configuration file.
- **`hibernate.properties`** (properties-param) A list of hibernate properties used to create SessionFactory that will be injected to JBoss Identity IDM configuration registry.
- **`hibernate.annotations`** A list of annotated classes that will be added to Hibernate configuration.
- **`hibernate.mappings`** A list of **`.xml`** files that will be added to hibernate configuration as mapping files.
- **`jndiName`** (value-param) If the 'config' parameter is not provided, this parameter will be used to perform JNDI lookup for **`IdentitySessionFactory`**.
- **`portalRealm`** (value-param) The realm name that should be used to obtain proper **`IdentitySession`**. The default is **`'PortalRealm'`**.
- **`apiCacheConfig`** (value-param) The Infinispan configuration file with cache configuration for PicketLink IDM API. It's different for cluster and non-cluster because Infinispan needs to be replicated in cluster environment.
- **`storeCacheConfig`** (value-param). The Infinispan configuration file with cache configuration for PicketLink IDM IdentityStore. Actually it's used only for LDAP store (not used with default DB configuration). It's different for cluster and non-cluster because Infinispan needs to be replicated in cluster environment.

Report a bug

9.2.2. PicketlinkIDMOrganizationServiceImpl Service

The **`org.exoplatform.services.organization.idm.PicketLinkIDMOrganizationServiceImpl`** key is a main entry point implementing **`org.exoplatform.services.organization.OrganizationService`**, and is dependent on

org.exoplatform.services.organization.idm.PicketLinkIDMService.

The **org.exoplatform.services.organization.idm.PicketLinkIDMOrganizationServiceImpl** service has the following options defined as fields of <object-param> of the **org.exoplatform.services.organization.idm.Config** type:

- **defaultGroupType** The name of the PicketLink IDM GroupType that will be used to store groups. The default is **GTN_GROUP_TYPE**.
- **rootGroupName** The name of the PicketLink IDM Group that will be used as a root parent. The default is **GTN_ROOT_GROUP**.
- **passwordAsAttribute** This parameter specifies if a password is stored using PicketLink IDM Credential object or as a plain attribute. The default is **false**.
- **useParentIdAsGroupType** This parameter stores the parent ID path as a group type in PicketLink IDM for any IDs not mapped with a specific type in 'groupTypeMappings'. If this option is set to **false**, and no mappings are provided under 'groupTypeMappings', then only one group with the given name can exist in the portal group tree.
- **pathSeparator** When 'userParentIdAsGroupType is set to **true**, this value will be used to replace all "/" characters in IDs. The "/" character is not allowed to be used in group type name in PicketLink IDM.
- **associationMembershipType** If this option is used, then each Membership, created with MembrshipType that is equal to the value specified here, will be stored in PicketLink IDM as simple Group-User association.
- **groupTypeMappings** This parameter maps groups added with portal API as children of a given group ID, and stores them with a given group type name in PicketLink IDM. If the parent ID ends with "/*", then all child groups will have the mapped group type. Otherwise, only direct (first level) children will use this type. This can be leveraged by LDAP if LDAP DN is configured in PicketLink IDM to only store a specific group type. This will then store the given branch in portal group tree, while all other groups will remain in the database.
- **forceMembershipOfMappedTypes** Groups stored in PicketLink IDM with a type mapped in 'groupTypeMappings' will automatically be members under the mapped parent. Group relationships linked by PicketLink IDM group association will not be necessary. This parameter can be set to false if all groups are added via portal APIs. This may be useful with LDAP configuration as, when set to true, it will make every entry added to LDAP appear in portal. This, however, is not true for entries added through the portal management user interface.
- **ignoreMappedMembershipType** If "associationMembershipType" option is used, and this option is set to true, then Membership with MembershipType configured to be stored as PicketLink IDM association will not be stored as PicketLink IDM Role.

Additionally, PicketlinkIDMOrganizationServiceImpl uses those defaults to perform identity management operations.

- The portal interface properties fields are persisted in PicketLink IDM using those attributes names: firstName, lastName, email, createdDate, lastLoginTime, organizationId, password (if password is configured to be stored as attribute).
- The portal interface properties fields are persisted in PicketLink IDM using those attributes names: label, description.

- The portal MembershipType interface properties fields are persisted in JBoss Identity IDM using those RoleType properties: description, owner, create_date, modified_date. A sample PicketLink IDM configuration file is shown below. To understand all the options it contains, see the *PicketLink IDM Reference Guide*.

```
<jboss-identity xmlns="urn:jboss:identity:idm:config:v1_0_beta"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:jboss:identity:idm:config:v1_0_alpha identity-
config.xsd">
    <realms>
        <realm>
            <id>PortalRealm</id>
            <repository-id-ref>PortalRepository</repository-id-ref>
            <identity-type-mappings>
                <user-mapping>USER</user-mapping>
            </identity-type-mappings>
        </realm>
    </realms>
    <repositories>
        <repository>
            <id>PortalRepository</id>

<class>org.jboss.identity.idm.impl.repository.WrapperIdentityStoreReposit
ory</class>
            <external-config/>
            <default-identity-store-id>HibernateStore</default-identity-
store-id>
            <default-attribute-store-id>HibernateStore</default-
attribute-store-id>
        </repository>
    </repositories>
    <stores>
        <attribute-stores/>
        <identity-stores>
            <identity-store>
                <id>HibernateStore</id>

<class>org.jboss.identity.idm.impl.store.hibernate.HibernateIdentityStore
Impl</class>
                <external-config/>
                <supported-relationship-types>
                    <relationship-
type>JBOSS_IDENTITY_MEMBERSHIP</relationship-type>
                    <relationship-type>JBOSS_IDENTITY_ROLE</relationship-
type>
                </supported-relationship-types>
                <supported-identity-object-types>
                    <identity-object-type>
                        <name>USER</name>
                        <relationships/>
                        <credentials>
                            <credential-type>PASSWORD</credential-type>
                        </credentials>
                        <attributes/>
                        <options/>
```

```
                    </identity-object-type>
                </supported-identity-object-types>
                <options>
                    <option>
                        <name>hibernateSessionFactoryRegistryName</name>
                        <value>hibernateSessionFactory</value>
                    </option>
                    <option>
                        <name>allowNotDefinedIdentityObjectTypes</name>
                        <value>true</value>
                    </option>
                    <option>
                        <name>populateRelationshipTypes</name>
                        <value>true</value>
                    </option>
                    <option>
                        <name>populateIdentityObjectTypes</name>
                        <value>true</value>
                    </option>
                    <option>
                        <name>allowNotDefinedAttributes</name>
                        <value>true</value>
                    </option>
                    <option>
                        <name>isRealmAware</name>
                        <value>true</value>
                    </option>
                </options>
            </identity-store>
        </identity-stores>
    </stores>
</jboss-identity>
```

Report a bug

Chapter 10. Token Service

The *Token Service* is used in authentication.

The token system prevents user account information being sent unencrypted within inbound requests. This increases authentication security.

The token service allows administrators to create, delete, retrieve and clean tokens as required. The service also defines a validity period of any given token. The token becomes invalid once this period expires.

Report a bug

10.1. Implementing Token Service API

All token services used in portal authentication must be implemented by subclassing an **AbstractTokenService** abstract class.

The following **AbstractTokenService** methods represents the contract between authentication runtime, and a token service implementation.

```
public Token getToken(String id) throws PathNotFoundException,
RepositoryException;
public Token deleteToken(String id) throws PathNotFoundException,
RepositoryException;
public String[] getAllTokens();
public long getNumberTokens() throws Exception;
public String createToken(Credentials credentials) throws
IllegalArgumentException,NullPointerException;
public Credentials validateToken(String tokenKey, boolean remove) throws
NullPointerException;
```

Report a bug

10.2. Configuring Token Services

The token services configuration includes specifying the token validity period. The token service is configured as a portal component in the portal scope, as opposed to the root scope.

In the XML example below, *CookieTokenService* is a subclass of **AbstractTokenService** so it has a property which specifies the validity period of the token.

The token service will initialize this validity property by looking for an ***init-param*** named **service.configuration**.

This property must have three values.

```
<component>
 <key>org.exoplatform.web.security.security.CookieTokenService</key>
 <type>org.exoplatform.web.security.security.CookieTokenService</type>
```

```
  <init-params>
    <values-param>
      <name>service.configuration</name>
      <!-- Service name -->
      <value>jcr-token</value>
      <!-- Amount of time -->
      <value>7</value>
      <!-- Unit of time -->
      <value>DAY</value>
      <value>autologin</value>
    </values-param>
  </init-params>
</component>
```

In this case, the service name is **jcr-token** and the token expiration time is one week.

The portal supports the following time units:

1. ***SECOND***
2. ***MINUTE***
3. ***HOUR***
4. ***DAY***

Report a bug

Chapter 11. Predefined User Configuration

The initial Organization configuration is specified by editing the content of ***JPP_DIST*/gatein/gatein.ear/portal.war:/WEB-INF/conf/organization/organization-configuration.xml**. This file uses the portal XML configuration schema. It lists several configuration plug-ins.

The **org.exoplatform.services.organization.OrganizationDatabaseInitializer** plug-in is used to specify the list of membership types/groups/users to be created.

The checkDatabaseAlgorithm initialization parameter determines how the database update is performed.

If its value is set to **entry**, it means that each user, group and membership listed in the configuration is checked each time GateIn Portal is started. If the entry does not exist in the database yet, it is created. If the checkDatabaseAlgorithm parameter value is set to empty, the configuration data will be updated to the database only if the database is empty.

Example 11.1. Membership Types

The predefined membership types are specified in the membershipType field of the **OrganizationConfig** plug-in parameter. See **organization-configuration.xml** for the complete configuration list.

```
<field name="membershipType">
  <collection type="java.util.ArrayList">
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$Membersh
ipType">
        <field name="type">
          <string>member</string>
        </field>
        <field name="description">
          <string>member membership type</string>
        </field>
      </object>
    </value>
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$Membersh
ipType">
        <field name="type">
          <string>owner</string>
        </field>
        <field name="description">
          <string>owner membership type</string>
        </field>
      </object>
     </value>
     <value>
       <object
type="org.exoplatform.services.organization.OrganizationConfig$Membersh
ipType">
         <field name="type">
```

```
          <string>validator</string>
        </field>
        <field name="description">
          <string>validator membership type</string>
        </field>
      </object>
    </value>
  </collection>
</field>
```

Example 11.2. Groups

The predefined groups are specified in the group field of the **`OrganizationConfig`** plug-in parameter.

```
<field name="group">
  <collection type="java.util.ArrayList">
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$Group">
        <field name="name">
          <string>portal</string>
        </field>
        <field name="parentId">
          <string></string>
        </field>
        <field name="type">
          <string>hierarchy</string>
        </field>
        <field name="description">
          <string>the /portal group</string>
        </field>
      </object>
    </value>
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$Group">
        <field name="name">
          <string>community</string>
        </field>
        <field name="parentId">
          <string>/portal</string>
        </field>
        <field name="type">
          <string>hierarchy</string>
        </field>
        <field name="description">
          <string>the /portal/community group</string>
        </field>
      </object>
    </value>
```

```
    ...
  </collection>
</field>
```

Example 11.3. Users

The predefined users are specified in the membershipType field of the **`OrganizationConfig`** plug-in parameter.

```
<field name="user">
  <collection type="java.util.ArrayList">
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$User">
        <field name="userName"><string>root</string></field>
        <field name="password"><string>exo</string></field>
        <field name="firstName"><string>root</string></field>
        <field name="lastName"><string>root</string></field>
        <field name="email"><string>exoadmin@localhost</string></field>
        <field name="groups">
<string>member:/admin,member:/user,owner:/portal/admin</string></field>
      </object>
    </value>
    <value>
      <object
type="org.exoplatform.services.organization.OrganizationConfig$User">
        <field name="userName"><string>exo</string></field>
        <field name="password"><string>exo</string></field>
        <field name="firstName"><string>site</string></field>
        <field name="lastName"><string>site</string></field>
        <field name="email"><string>exo@localhost</string></field>
        <field name="groups"><string>member:/user</string></field>
      </object>
    </value>
    ...
  </collection>
</field>
```

Example 11.4. User Creation

The **`org.exoplatform.services.organization.impl.NewUserEventListener`** plug-in specifies which groups all created users will become members of. It specifies the groups and the memberships to use (while the group is only a set of users, a membership type represents a user's role within a group). It also specifies a list of users that must not be processed (for example, administrative users such as 'root').

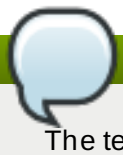
```
<component-plugin>
  <name>new.user.event.listener</name>
  <set-method>addListenerPlugin</set-method>

<type>org.exoplatform.services.organization.impl.NewUserEventListener</
type>
```

```
  <description>this listener assign group and membership to a new
created user</description>
  <init-params>
    <object-param>
      <name>configuration</name>
      <description>description</description>
      <object
type="org.exoplatform.services.organization.impl.NewUserConfig">
        <field  name="group">
          <collection type="java.util.ArrayList">
            <value>
              <object
type="org.exoplatform.services.organization.impl.NewUserConfig$JoinGrou
p">
                <field name="groupId"><string>/platform/users</string>
</field>
                <field name="membership"><string>member</string>
</field>
              </object>
            </value>
          </collection>
        </field>
        <field  name="ignoredUser">
          <collection type="java.util.HashSet">
            <value><string>exo</string></value>
            <value><string>root</string></value>
            <value><string>company</string></value>
            <value><string>community</string></value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component-plugin>
```

Report a bug

11.1. Monitoring User Creating

The plugin of type **`org.exoplatform.services.organization.impl.NewUserEventListener`** specifies which groups all the newly created users should become members of. It specifies the groups and the memberships to use (while the group is just a set of users, a membership type represents a user's role within a group). It also specifies a list of users that should not be processed (for example, administrative users like 'root').



Note

The terms 'membership' and 'membership type' refer to the same thing, and are used interchangeably.

```
<component-plugin>
```

```
  <name>new.user.event.listener</name>
  <set-method>addListenerPlugin</set-method>

<type>org.exoplatform.services.organization.impl.NewUserEventListener</ty
pe>
  <description>this listener assign group and membership to a new created
user</description>
  <init-params>
    <object-param>
      <name>configuration</name>
      <description>description</description>
      <object
type="org.exoplatform.services.organization.impl.NewUserConfig">
        <field  name="group">
          <collection type="java.util.ArrayList">
            <value>
              <object
type="org.exoplatform.services.organization.impl.NewUserConfig$JoinGroup"
>
                <field name="groupId"><string>/platform/users</string>
</field>
                <field name="membership"><string>member</string></field>
              </object>
            </value>
          </collection>
        </field>
        <field  name="ignoredUser">
          <collection type="java.util.HashSet">
            <value><string>exo</string></value>
            <value><string>root</string></value>
            <value><string>company</string></value>
            <value><string>community</string></value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component-plugin>
```

Report a bug

Chapter 12. Single Sign-on

12.1. File Name Conventions

The following naming conventions are used in file paths to improve their readability. Each convention is styled so that it stands out from the rest of the text. Where you see *[version]*, replace this with the current version of the portal when reading the directory locations.

CAS_DIR

The installation root of the Central Authentication Service (CAS) single sign-on framework. This directory is an arbitrary location chosen when CAS is downloaded and installed.

This convention is mentioned in the Administration and Configuration guide chapter SSO_Single_Sign_On_-Central_Authentication_Service

ID_HOME

The **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/organization/`** directory, which contains identity-related configuration resources.

This convention is mainly used in LDAP_Integration in Administration and configuration guide.

JPP_DIST

The installation root of the portal instance. For example, if the portal distribution archive is extracted to the **`/opt/jboss/RHJP/`** directory, the *JPP_DIST* directory is **`/opt/jboss/RHJP`**.

This directory contains the **`jboss-jpp-[version];`**, **`gatein-management`** and **`gatein-sso`** directories, and is used extensively in sections that contain configuration stored in these directories.

JPP_HOME

The **`JPP_DIST/jboss-jpp-[version];`** directory, which contains the application server and the configuration files necessary to run the portal.

This directory contains the **`gatein`**, **`modules`** and **`standalone`** directories.

TOMCAT_HOME

The installation root of the Apache Tomcat server. Apache Tomcat is a simple Java-based web server that can host servlets or JSP applications. It is not a part of the portal, however, it is used in various examples in this guide to host single sign-on authentication services.

See Also:

- Section 12.2, “Single Sign-on (SSO) Configuration”

Report a bug

12.2. Single Sign-on (SSO) Configuration

The portal provides an implementation of single sign-on (**SSO**) as an integration and aggregation platform.

When logging into the portal, users can access many systems through portlets using a single identity. In many cases, however, the portal infrastructure must be integrated with other SSO enabled systems.

There are many different Identity Management solutions available. In most cases each SSO framework provides a unique way to plug into a Java EE application.

This section will cover the implementation of four different SSO plug-ins with the portal:

- Central Authentication Service.
- Java Open Single Sign-on.
- OpenAM.
- Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO).

In this tutorial, the SSO server is being installed in a Tomcat environment.

All the packages required for SSO setup can be found in the **`JPP_DIST/gatein-sso`** directory of the portal binary package.

Warning

Users are advised to not run any portal extensions that could override the data when manipulating the **`gatein.ear`** file directly.

See Also:

- Section 12.4.1, “Downloading Central Authentication Service”
- Chapter 13, *Java Open Single Sign-on*
- Chapter 14, *OpenAM*
- Chapter 15, *Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO)*

Report a bug

12.3. Central Authentication Service (CAS)

The CAS single sign-on (SSO) plug-in enables seamless integration between the platform and the CAS SSO framework. General information about CAS can be found on the Jasig website.

Report a bug

12.3.1. Authentication Process with Central Authentication Service integration

The authentication process with CAS integration occurs in the following order:

1. A user visits the main portal page, and wishes to authenticate. The user clicks Sign in.
2. Normally this action would present the portal login dialog, however with SSO integration enabled, the action redirects the user to a marker URL such as http://localhost:8080/portal/sso.

The portal handles this user action by calling the interceptor (Servlet filter) **`LoginRedirectFilter`**, which redirects the user seamlessly away from the **`/portal/sso`** URL to the CAS server page.

3. The interceptor redirects the user to the CAS login page http://localhost:8888/cas/login. The user enters the correct authentication information, and submits the form.

   The CAS server retrieves the information from the identity store. The store could be an external database, a LDAP server, or from information obtained through an authentication plug-in such as the one shipped with the portal.

4. Once CAS determines the user has the correct access privileges to access the portal server, CAS redirects the user back to the portal through another marker URL such as http://localhost:8080/portal/initiatelogin.

   The **InitiateLoginFilter** interceptor acts on the user redirection to /portal/initiatelogin by obtaining a CAS ticket attached in the HTTP request inside the ticket parameter. The interceptor then delegates validation of this ticket to a configured **CASAgent** component.

5. The *CASAgent* validates the ticket by sending a validation request to the CAS server through a configured back channel. The CAS server validates the request, and ensures it contains the user name of the authenticated user in step 3.

6. After SSO validation, InitiateLoginFilter redirects the user to the portal login URL http://localhost:8080/portal/login, which initiates JAAS authentication.

   The **SSOLoginModule** detects whether the user has been successfully validated by CASAgent. If this is the case, the login module obtains data about user (groups, memberships) from OrganizationService and encapsulates the details into an **Identity** object.

7. JBoss Portal completes the authentication request by establishing the JAAS Subject, and saves the roleIdentity object to the IdentityRegistry.

8. After successful JAAS authentication, the user is redirected to the portal in an authenticated state.

For more information about the available Login Modules shipped with the product, see the Red Hat JBoss Application Platform *Security Guide*.

See Also:

- Section 12.4.3, “Authentication Plugin for Central Authentication Service (CAS)”
- Section 7.3, “Login Modules”

Report a bug

12.3.2. Logging out Process with Central Authentication Service integration

The logout process with CAS integration occurs in the following order:

1. The authenticated user clicks the Sign out link.

2. The **CASLogoutFilter** interceptor recognizes the logout request, and redirects the user to the CAS logout page http://localhost:8888/cas/logout.

3. The CAS server logs out the user, and invalidates the CAS cookie CASTGC.

4. CAS redirects the user back to the portal using the logout redirection.

   If the CASLogoutFilter is enabled, the user is logged out from both the portal and CAS server.

5. The logout redirection request completes the logout process on the CAS server's side, and the user is redirected to the portal's anonymous page.

See Also:

- Section 12.4.5, “Setting up Logout Redirection”

Report a bug

12.3.3. Configuration Result

For scope purposes, the setup instructions assume the following configuration outcomes:

- The CAS 3.5 is downloaded, and required changes are made to authentication plug-in, logout redirection, and CASTGC cookie configuration.
- Once configured, Apache Maven is used to create the custom CAS web archive, suitable for deployment.
- The WAR is deployed to the Apache Tomcat server, which acts as the host for the CAS.
- Apache Tomcat is configured to listen on localhost:8888.
- The portal is configured to listen on localhost:8080.

Report a bug

12.4. Configuration for Central Authentication Service (CAS)

12.4.1. Downloading Central Authentication Service

CAS can be downloaded from http://www.jasig.org/cas/download. The supported version is CAS 3.5. More recent CAS versions may also work, however have not been officially tested as part of this specific configuration exercise.

Extract the downloaded file into a suitable directory location. This location will be referred to as **`CAS_DIR`** in subsequent configuration instructions.

Report a bug

12.4.2. Modifying the Central Authentication Service (CAS) Server

To configure the CAS server correctly, the most effective way is to make the necessary changes directly in the CAS code base. Follow the instructions in the sections below to make the required changes to the CAS code base, before using Maven to build the CAS web archive.

Report a bug

12.4.3. Authentication Plugin for Central Authentication Service (CAS)

While it is possible (and perfectly acceptable) for an administrator to configure CAS to retrieve user credentials from an external database, or from a LDAP server, it is also possible to use JBoss technology.

CAS can be configured to make secure authentication callbacks to a RESTful service installed on the remote portal instance using the supplied CAS **`AuthenticationPlugin`**.

Implementing the **`AuthenticationPlugin`** on the CAS server has the advantage of leveraging a single identity storage for portal user, group and role data. If a new user is added using the portal user management interface, the user information is instantly accessible to the CAS server through the technology implemented by the **`AuthenticationPlugin`**.

The plug-in verifies user credentials by connecting to an existing portal instance using REST over the HTTP protocol. The portal serves a REST authentication callback request, and verifies the user identity against the portal's own identity storage provided by the PicketLink IDM OrganizationService. The **`AuthenticationPlugin`** receives the portal's response to the CAS server, and continues with the authentication process based on user data in the response.

Report a bug

12.4.4. Configuring the Authentication Plugin

For the plug-in to function correctly, it must be properly configured on the CAS server to connect to this service. Set up the server to authenticate against the portal using the REST call-back.

Procedure 12.1. Configuring the Authentication Plug-in

1. Open **`CAS_DIR/cas-server-webapp/src/main/webapp/WEB-INF/deployerConfigContext.xml`**.

2. Replace the default configuration, which declares the Jasig **`SimpleTestUsernamePasswordAuthenticationHandler`** Authentication Handler with the following supported Authentication Handler.

```
<bean class="org.gatein.sso.cas.plugin.AuthenticationPlugin">
  <property name="gateInProtocol"><value>http</value></property>
  <property name="gateInHost"><value>localhost</value></property>
  <property name="gateInPort"><value>8080</value></property>
  <property name="gateInContext"><value>portal</value></property>
  <property name="httpMethod"><value>POST</value></property>
</bean>
```

Note

This configuration is available in the ***`JPP_DIST`*`/gatein-sso/cas/plug-in/WEB-INF/deployerConfigContext.xml`** file. If you choose to take this configuration file, ensure the default host, port and context parameters are adjusted to match the values corresponding to the remote portal instance.

3. Copy all jars from ***`JPP_DIST`*`/gatein-sso/cas/plug-in/WEB-INF/lib/`** to the **`CAS_DIR/cas-server-webapp/src/main/webapp/WEB-INF/lib`** directory.

Report a bug

12.4.5. Setting up Logout Redirection

The CAS server displays the CAS logout page with a link to return to the portal by default. To make the CAS server redirect to the portal page after a logout, modify **`CAS_DIR/cas-server-webapp/src/main/webapp/WEB-INF/cas-servlet.xml`** to include the **`followServiceRedirects="true"`** parameter:

```
<bean id="logoutController"  class="org.jasig.cas.web.LogoutController"
  p:centralAuthenticationService-ref="centralAuthenticationService"
  p:logoutView="casLogoutView"
  p:warnCookieGenerator-ref="warnCookieGenerator"
  p:ticketGrantingTicketCookieGenerator-
ref="ticketGrantingTicketCookieGenerator"
  p:followServiceRedirects="true"/>
```

Report a bug

12.4.6. Cookie Configuration for Central Authentication Service (CAS) Single Sign-on

Jasic CAS uses a cookie named *CAS Ticket Granting Cookie* (CASTGC) to control the authentication state within the browser session. The cookie contains a Ticket Granting Ticket (TGT), which preserves SSO authentication where more than one site is controlled by the same SSO profile.

Report a bug

12.4.7. Portal Authentication using Central Authentication Service Ticket Granting Cookie (CASTGC)

Two portal servers are provisioned that use a single CAS server to manage authentication. The portals are named **`accounts`** and **`services`**.

When a user initially accesses the **`accounts`** portal, they provide their SSO credentials, and CAS authenticates them as a registered user. The user then switches to the **`services`** portal, and is authenticated when she clicks the Sign in link.

This behavior is correct given this example because the browser instance stores the browser authentication state using the CASTGC cookie. The CASTGC cookie in this instance creates new ticket for the **`services`** portal automatically based on the authentication state present for the accounts portal.

This behavior exists through a secured connection only (https connection). To benefit from authentication across two or more portals, one of the options below must be implemented. Choose the correct option based on the deployment environment:

`Testing`

Alter the CASTGC cookie to be insecure.

The cookie can be accessed through http (insecure) connections.

To configure this test behavior, open **`CAS_DIR/cas-server-webapp/src/main/webapp/WEB-INF/spring-configuration/ticketGrantingTicketCookieGenerator.xml`** and switch the attribute **`cookieSecure`** to false.

```
<bean id="ticketGrantingTicketCookieGenerator"
  p:cookieSecure="false"
  p:cookieMaxAge="-1"
  p:cookieName="CASTGC"
  p:cookiePath="/cas" />
```

Production

Correctly implement the https protocol for all production servers that rely on CAS. This configuration is the recommended method for any production server, and ensures greater security for CAS connections. See the Jasig documentation about securing CAS https://wiki.jasig.org/display/CASUM/Securing+Your+New+CAS+Server for information and resources.

Report a bug

12.4.8. Installing Apache Tomcat Server

Prerequisites:

- Section 12.1, “File Name Conventions”

Install and configure Apache Tomcat 7, which provides the host server for the CAS server.

Procedure 12.2. Configuring Apache Tomcat for CAS

1. Visit http://tomcat.apache.org/download-70.cgi and download the Tomcat 7 binary distribution.
2. Extract and install the binary on the server that is required to host CAS. This directory is now referred to as *TOMCAT_HOME*.
3. Edit **`TOMCAT_HOME/conf/server.xml`** and change port 8080 to 8888 to avoid a conflict with the default portal listen port.

Important

If the Apache Tomcat server is installed on the same machine as the portal, ensure other listen ports common to both servers are changed to prevent configuration issues. For example, change the Tomcat admin port from 8005 to 8805, and the Tomcat AJP port from 8009 to 8809.

4. Ensure all Apache Tomcat ports are open in the server firewall, and the service is enabled and running so the portal can communicate with Apache Tomcat on the same server.

Report a bug

12.5. Modifying the Portal

Before building and deploying the Jasig CAS sever, configuration must be implemented on the portal to prepare it for CAS integration.

Report a bug

12.5.1. Configuring Portal Single Sign-on

The main portal configuration file for SSO integration is **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/security-sso-configuration.xml**. All required SSO components such as agents and SSO interceptors are configured in this file.

In most cases, it will never be necessary to edit **security-sso-configuration.xml** directly when using the portal. The portal architecture allows users to override the base configuration described in this file using name/value pairs configured in one place: **JPP_HOME/standalone/configuration/gatein/configuration.properties**

The exception to this rule is where configuration present in **security-sso-configuration.xml** is fundamentally unsuitable for the production environment the server will be deployed to, or when additional underlying functionality is required (for example, another custom interceptor).

Report a bug

12.5.2. Configuration properties for Portal Single Sign-on

To prepare the portal for CAS authentication, SSO filters and login modules need to be specified in global configuration files. The location of the CAS server, as configured in a locally-running Apache Tomcat server, must also be specified.

Procedure 12.3. Configuring SSO configuration.properties for CAS

1. Open **JPP_HOME/standalone/configuration/gatein/configuration.properties** and locate the SSO section in the file.

2. Make the following changes to the file to declare the correct login module, server and portal URLs, and the logout filter.

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
gatein.sso.login.module.class=org.gatein.sso.agent.login.SSOLoginMo
dule
gatein.sso.server.url=http://localhost:8888/cas
gatein.sso.portal.url=http://localhost:8080
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.CASLogout
Filter
gatein.sso.filter.logout.url=${gatein.sso.server.url}/logout
gatein.sso.filter.login.sso.url=${gatein.sso.server.url}/login?
service=${gatein.sso.portal.url}/@@portal.container.name@@/initiat
essologin
```

gatein.sso.enabled

Specifies whether SSO integration is enabled on the portal. With this option set to "true" when a user clicks the Sign in link, the user is redirected to the **/portal/sso/** URL rather than a standard Sign in dialog.

gatein.sso.callback.enabled

Specifies whether the REST callback authentication handler is enabled.

The handler is required if the CAS server must use the SSO Authentication plug-in to handle portal authentication. The callback handler is enabled by default. Set the parameter to false if the authentication plug-in on the CAS server side is not required.

gatein.sso.login.module.enabled

Specifies whether a pre-defined SSO login module declared in **JPP_HOME/standalone/configuration/standalone.xml** is used for authentication. When the property is set to **true**, the **SSODelegateLoginModule** delegates work to another login module, as specified using the **gatein.sso.login.module.class** property. **SSODelegateLoginModule** will also resend all its options to its delegate.

This parameter removes the need to manually change any login module configuration in the **standalone.xml** file, which simplifies platform configuration.

gatein.sso.login.module.class

Specifies the classname of the login module **SSODelegateLoginModule** will delegate to. This parameter will work only if gatein.sso.login.module.enabled is specified.

gatein.sso.server.url

Specifies the URL from which the CAS server is accessible.

gatein.sso.portal.url

Specifies the URL from which the portal is accessible.

gatein.sso.filter.logout.class

Specifies the class of the logout filter. In the example above **org.gatein.sso.agent.filter.CASLogoutFilter** is the correct choice because this filter is able to redirect to the CAS server and perform logout on the CAS side.

gatein.sso.filter.logout.url

Specifies the CAS server logout URL, which is used for redirection by the logout filter.

gatein.sso.filter.logout.enabled

Optional parameter, which specifies whether the logout interceptor is enabled. To disable logout on CAS side, set the parameter value to **false**. This causes both options **gatein.sso.filter.logout.class** and **gatein.sso.filter.logout.url** to be ignored.

When a user logs out of the portal, the CAS authentication ticket is still valid for other CAS authenticated sites.

gatein.sso.filter.login.sso.url

Specifies the CAS server login URL, which is used by LoginRedirectFilter for redirection to the CAS server login page.

Note

The string **@@portal.container.name@@**is dynamically replaced when the URL is interpreted by the platform's SSO Component. It is recommended that this string is used over manually specifying the name of the portal for future maintenance and ease of configuration changes.

See Also:

- Section 12.4.3, “Authentication Plugin for Central Authentication Service (CAS)”

Report a bug

12.6. Building and Deploying Central Authentication Service (CAS)

Jasig CAS uses Apache Maven to build the **cas.war** file. Follow the instructions to produce this file, and deploy it to the Apache Tomcat server.

Procedure 12.4. Building and Deploying CAS to Tomcat

1. Install Maven by following the recommendations and links in the Building and Deploying section of the Jasig CAS user documentation.
2. In a terminal, navigate to **CAS_DIR/cas-server-webapp/**, and run **mvn clean install -Dmaven.test.skip=true**.
3. Copy **CAS_DIR/cas-server-webapp/target/cas.war** to **TOMCAT_HOME/webapps**.
4. Tomcat should be running by default, if the process has been followed up to this step. Start the portal, and verify the server is running by opening http://localhost:8080/.
5. Open http://localhost:8888/cas to verify the CAS server has correctly deployed to Tomcat. If the link does not open the CAS login page, restart Apache Tomcat and try again.

The CAS server is now deployed to Tomcat, and the portal will now redirect users to the CAS login page when they click on the Sign In link.

Report a bug

Chapter 13. Java Open Single Sign-on

Java Open Single Sign-on (JOSSO) is an open-source single sign-on solution based on Java EE. It allows multiple web servers or web applications to authenticate users with a credential store. Detailed information about JOSSO can bhttp://docbuilder.usersys.redhat.com/16459/remarks/#Configure_Salesforce_Domaine found at http://www.josso.org.

JOSSO integration with the portal requires an Apache Tomcat server instance to host JOSSO. The portal communicates with the JOSSO server through a SSO plug-in.

Setting up the integration consists of two steps:

- Setting up the JOSSO server
- Setting up the portal to use the JOSSO server

These two steps differ depending on the chosen version of JOSSO, as described in Section 13.2, “Java Open Single Sign-on Version 1.8” and Section 13.3, “Java Open Single Sign-on Version 2.2”. After completing the procedures described in either section, all links redirecting to user authentication pages will redirect to the JOSSO centralized authentication form.

Report a bug

13.1. Authenticating Java Open Single Sign-on

The login workflow for JOSSO is quite similar to that used for CAS authentication.

When a user clicks to sign into a portal, they are redirected to the JOSSO login screen, where they supply the appropriate credentials. They are then redirected (with access authorization) back to the portal.

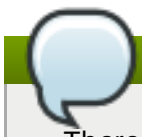
The **`JOSSOAgent`** component performs a validation of the authorization ticket with the JOSSO server via a back channel after the **`InitiateLoginFilter`** has delegated the ***`josso_assertion_id`*** request to it. The JOSSO agent and JOSSO server communicate via web services.

After a successful validation, the user identity is successfully established and the user is logged into the requested Portal.

On logout, **`JOSSOLogoutFilter`** performs a logout on both the Portal and the JOSSO server (similar to the process for CAS).

While the authentication plug-in (which is able to send REST requests to the portal, receive the response, and authenticate the user on the JOSSO side) is supported, this support is only for JOSSO 1.8 (not JOSSO 2.2 as at this release).

In this section, it is assumed that the portal is running on JBoss Enterprise Application Platform 6 using **`localhost:8080`** and that the JOSSO server is running on Tomcat, using **`localhost:8888`**.



Note

There are differences between various JOSSO minor versions (especially between JOSSO versions 1.8.1 and 1.8.2). Instructions are slightly different between various versions. This is described in procedures as required.

See Also:

- Section 12.3.1, “Authentication Process with Central Authentication Service integration”

Report a bug

13.2. Java Open Single Sign-on Version 1.8

JOSSO can be downloaded from http://sourceforge.net/projects/josso/files/. Use any 1.8.z version in a package that embeds Apache Tomcat. Once downloaded, extract the package into what will be called *JOSSO_HOME* in this example.

Report a bug

13.2.1. Setting up Java Open Single Sign-on Server

This section describes how to set up the JOSSO server to authenticate against the portal using the REST authentication plug-in.

Note

In this example, the JOSSO server is installed on Tomcat.

Procedure 13.1. Java Open Single Sign-on setup

1. Copy the contents of the **`JPP_DIST/gatein-sso/josso/josso-2.2/plugin/`** directory into the *JOSSO_HOME* directory. Among the files that will be copied, the following ones are the most important:

   Note

   It is recommended to use the SSO authentication plug-in with JOSSO.

   - ***`JOSSO_HOME`*`/lib/josso-gateway-config.xml`**

     The original file is being replaced. Create a backup of the file before adding the new file.

   - ***`JOSSO_HOME`*`/lib/josso-gateway-gatein-stores.xml`**

     This file is not present in the original *JOSSO_HOME* download.

   - **`JOSSO_HOME/webapps/josso/WEB-INF/classes/gatein.properties`**

     This file is not present in the original *JOSSO_HOME* download. You have to edit the file and change the host and port to match the portal instance. The values are used by the authentication plug-in when sending REST requests over HTTP.

2. Edit **`JOSSO_HOME/conf/server.xml`** and replace the **`8080`** port to **`8888`** to change the default Tomcat port and avoid a conflict with the default portal port (for testing purposes).

Port Conflicts

If the portal is running on the same machine as Tomcat, other ports need to be changed in addition to **8080** to avoid port conflicts. They can be changed to any free port. For example, you can change the admin port from **8005** to **8805**, and AJP port from **8009** to **8809**.

3. Tomcat allows access to **http://localhost:8888/josso/signon/login.do**. If you are using the SSO Authentication plug-in, the login will not be available as the portal platform must be configured to use the instance.

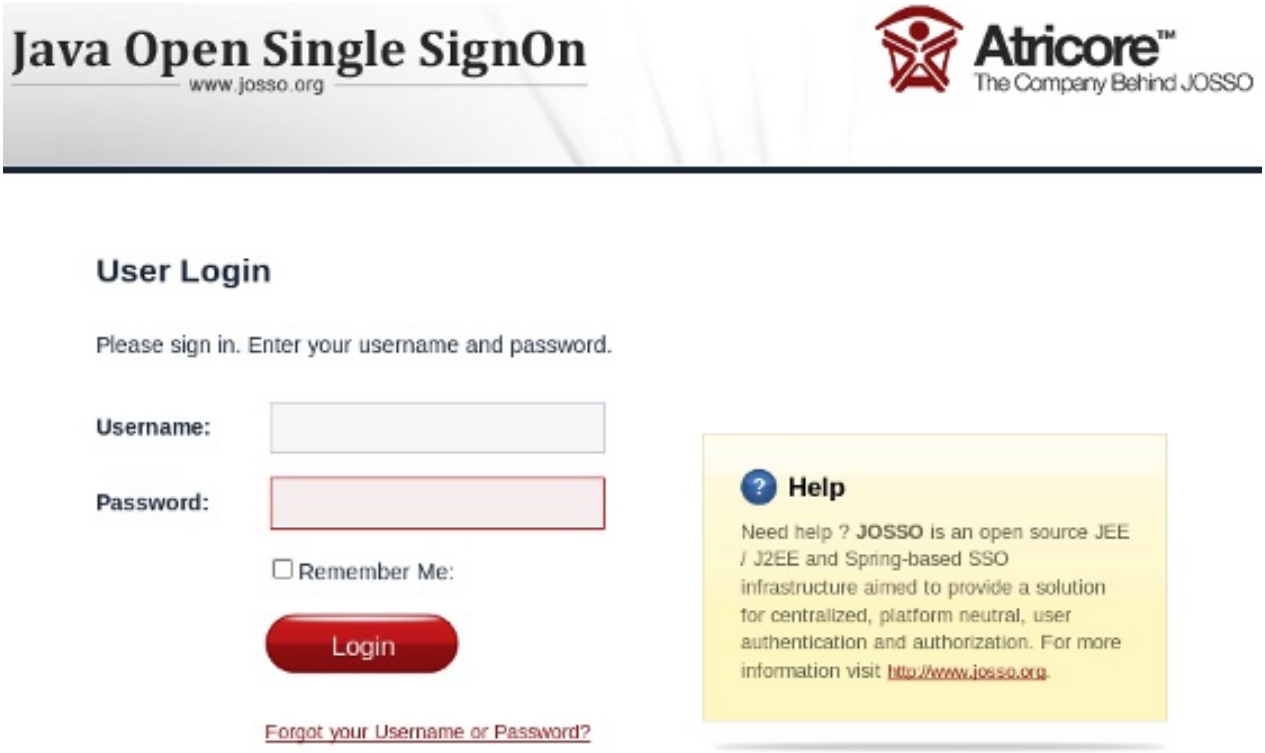


Figure 13.1. JOSSO Login Page

See Also:

- Section 13.1, “Authenticating Java Open Single Sign-on”

Report a bug

13.2.2. Setting up Java Open Single Sign-on Client

To enable JOSSO on the client side you have to modify the configuration properties file **JPP_HOME/standalone/configuration/gatein/configuration.properties**.

1. Set SSO properties

   Set the values of **#SSO** as per the SSO configuration properties:

Example 13.1. SSO configuration.properties

```
#SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
gatein.sso.login.module.class=org.gatein.sso.agent.login.SSOLogin
Module
gatein.sso.josso.agent.config.file=sso/josso/1.8/josso-agent-
config.xml
gatein.sso.josso.properties.file=file:${jboss.home.dir}/standalone
/configuration/gatein/configuration.properties
gatein.sso.josso.host=localhost:8888
gatein.sso.josso.base.url=http://${gatein.sso.josso.host}/josso/si
gnon
gatein.sso.server.url=${gatein.sso.josso.base.url}/login.do
gatein.sso.portal.url=http://localhost:8080
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.JOSSOLo
goutFilter
gatein.sso.filter.logout.url=${gatein.sso.josso.base.url}/logout.
do
gatein.sso.filter.login.sso.url=${gatein.sso.server.url}?
josso_back_to=${gatein.sso.portal.url}/@@portal.container.name@@
/initiatessologin
```

Set JOSSO properties.

- Value of Logout filter: **`org.gatein.sso.agent.filter.JOSSOLogoutFilter`**.
- Location of JOSSO server**`gatein.sso.josso.host`**.
- Change the url **`gatein.sso.portal.url`** to access the portal on any URL other than

  localhost:8080
  .
- The location of the Agent configuration file **`gatein.sso.josso.agent.config.file`** is relative to classpath. Therefore the agent file location is located at **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/josso/1.8/josso-agent-config.xml`**.

  In most of the cases this file is does not need any properties to be setup.

2. JOSSO has some specific dependencies, which differ between various versions. The original **`org.gatein.sso`** SSO module must be replaced with one appropriate for your version of JOSSO. The alternate modules are available in the JOSSO download.

   a. Delete the **`JPP_HOME/modules/org/gatein/sso`** directory.

   b. Copy the **`SSO_HOME/josso/gatein-josso-<version>/modules/org/gatein/sso`** directory into **`JPP_HOME/modules/org/gatein/`**.

From now on, all links redirecting to the user authentication pages will redirect to the JOSSO centralized authentication form. If you set Authentication plug-in for JOSSO, you can login with portal credentials (for example, john/gtn) on the JOSSO side.

Report a bug

13.3. Java Open Single Sign-on Version 2.2

JOSSO 2.2 takes a different approach to SSO than JOSSO 1.8. It is designed to allow users to create their own SSO environment by modeling it in a flash web application called **atricore-console**.

Report a bug

13.3.1. Setting up Java Open Single Sign-on Server

Prerequisites

- Download JOSSO 2.2.0 from http://www.josso.org
- Follow the instructions from the JOSSO 2 quickstart in http://www.josso.org/confluence/display/JOSSO1/JOSSO2+Quick+start .
- Access the **atricore** console at **http://*server.local.network*:8081/atricore-console** (*server.local.network* is the virtual host defined in **/etc/hosts**).

Procedure 13.2. Java Single Sign-on setup

1. Login to the portal with the username/password combination; **admin/admin**.
2. Create a new, empty Identity appliance with the following properties:

   Table 13.1.

Setting	Value
Name	***MYFIRSTIA***
Realm name	***com.mycompany.myrealm***
Appliance location	**http://server.local.network:8081**

3. Create a new Identity provider named AcmeIDP (use the default settings).
4. Create an Identity vault called IDPUsers and connect it with AcmeIDP via Identity lookup connection.
5. Create a Service provider called SP1 but let the hosts to be on

   server.local.network:8081

   .
6. Create an Identity vault called SP1Users and wire it with SP1 via Identity lookup connection.
7. Create empty temporary directory **/tmp/tomcat7**.

   Create new execution environment of type Tomcat with the following parameters in the **atricore** console.

   Table 13.2. Parameters for creating a new execution environment in the atricore console

Setting	Value
Name	***SP1EE***
Version	***7.0.x***

Setting	Value
Target host	***Local***
Install home	***/tmp/tomcat7*** (the **/tmp/tomcat7** directory must exist, but it can be empty).

8. Wire SP1 and SP1EE via an Activation connection.

Note

All parameters of the connection can keep their default values, with the exception of the **Partner application location** parameter, whose value needs to be changed to http://localhost:8080/portal.

9. Wire SP1 and AcmeIDP via Federated connection.
10. Click **Save**.
11. Go to the **Identity appliance life cycle management** tab and go through life cycle of Identity appliance (**Saved** → **Staged** → **Deployed** → **Started**) as suggested in the quickstart.
12. Go to the **Account & Entitlement management** tab and create users. Users must be created this way because REST callbacks to the Portal are not supported in this release.

    This example will create the following user/password accounts: **john**/**password**, **root**/**password** and **demo**/**password**.

Report a bug

13.3.2. Setting up Java Open Single Sign-on Client

Prerequisites:

- Section 13.3.1, “Setting up Java Open Single Sign-on Server”

Prerequisites

- Section 13.3.1, “Setting up Java Open Single Sign-on Server”.

1. Set the properties in the SSO configuration file**JPP_HOME/standalone/configuration/gatein/configuration.properties**

Example 13.2. SSO file properties

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
gatein.sso.login.module.class=org.gatein.sso.agent.login.SSOLogin
Module
gatein.sso.filter.initiatelogin.enabled=false
gatein.sso.filter.initiatelogin.josso2.enabled=true
gatein.sso.josso.agent.config.file=sso/josso/2.2/josso-agent-
```

```
config.xml
gatein.sso.josso.properties.file=file:${jboss.home.dir}/standalone
/configuration/gatein/configuration.properties
gatein.sso.portal.url=http://localhost:8080
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.JOSSOLo
goutFilter
gatein.sso.filter.logout.url=
gatein.sso.josso.host=server.local.network:8081
gatein.sso.server.url=http://${gatein.sso.josso.host}
gatein.sso.josso.identityApplianceId=MYFIRSTIA
gatein.sso.josso.partnerAppId=SP1
gatein.sso.josso.partnerAppPoint=SP1EE
gatein.sso.filter.login.sso.url=${gatein.sso.server.url}/IDBUS/${
gatein.sso.josso.identityApplianceId}/${gatein.sso.josso.partnerA
ppPoint}/JOSSO/SSO/REDIR?
josso_back_to=${gatein.sso.portal.url}/@@portal.container.name@@
/josso_security_check&josso_partnerapp_id=${gatein.sso.josso.partn
erAppId}
```

Note

The property **gatein.sso.filter.logout.url** is empty because the logout URL will be obtained from the JOSSO agent configuration set in file **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/josso/2.2/josso-agent-config.xml**.

2. Update the SSO module in JBoss Enterprise Application Platform 6:

    a. Delete the **JPP_HOME/modules/org/gatein/sso** directory.

    b. Copy the **SSO_HOME/josso/gatein-josso-182/modules/org/gatein/sso** into **JPP_HOME/modules/org/gatein/** directory.

3. Test the configuration:

    a. Start the Portal.

    b. Access **http://localhost:8080/portal** and click Sign in. You will be redirected to the JOSSO instance, but you will need to login with the username and password created via the JOSSO console (for example **john**/**password**) as REST callbacks are not supported.

After a successful login to JOSSO, you will be redirected to the portal authenticated as **john**.

Report a bug

Chapter 14. OpenAM

OpenAM is an open source access management, entitlements and federation server platform. It is a successor of OpenSSO, the access management and federation server platform, whose integration was available in Red Hat JBoss Portal 5. As the development of OpenSSO has been discontinued, the OpenSSO integration has been replaced with OpenAM integration in Red Hat JBoss Portal 6.

Report a bug

14.1. Downloading OpenAM

OpenAM must be obtained from Forgerock. Only the 9.5 and 10.0 versions of OpenAM are fully tested and supported with the portal. Integration with other versions may be functional, but has not been fully tested and is therefore not supported.

The procedures in this section assume that OpenAM is obtained in the form of a ZIP distribution. Once downloaded, extract the contents of the ZIP package into a location of your choice. This location will be referred to as **`OPENAM_HOME`** in the following text.

Report a bug

14.2. OpenAM Workflow

When the OpenAM integration is configured and a user clicks the **`Sign in`** link on a portal page, they are redirected to the OpenAM login screen, where they provide login credentials. Authentication on the OpenAM server side is performed by the portal SSO Authentication Plug-in. The plug-in sends a REST request to the portal, obtains a response, and authenticates the user on the OpenAM side based on the response.

After successful authentication with OpenAM, an OpenAM authentication ticket is stored in the **`iPlanetDirectoryPro`** cookie in the client browser and the user is redirected back to the portal page. When the portal page is requested, the **`InitiateLoginFilter`** interceptor delegates validation of the OpenAM ticket to the **`OpenSSOAgent`** component. The **`OpenSSOAgent`** then uses the OpenAM REST API to perform back channel validation of the ticket with the OpenAM server. After successful validation, user identity is established and the user is logged into the portal.

When logout is requested by clicking the **`Sign out`** button on a portal page, the OpenSSOLogoutFilter interceptor performs logout on both the portal and OpenAM servers.

The OpenAM login and logout workflow is similar to the CAS authentication workflow.

See Also:

- Section 12.3.1, “Authentication Process with Central Authentication Service integration”

Report a bug

14.3. OpenAM Server

14.3.1. Setting up OpenAM Server

This section contains procedures that need to be followed to set up an OpenAM server for authentication against the portal. The authentication set up by these procedures is ensured by the portal SSO Authentication Plug-in. The plug-in will be installed in OpenAM and configured to perform authentication against the portal using a REST callback.

Procedure 14.1. Setting up OpenAM Server

To setup OpenAM server, follow these steps.

1. Deploy the OpenAM server.
2. Add the authentication plugin.
3. Configure a realm in OpenAM user interface.

Note

Using the REST callback as presented in this section is not mandatory. You can achieve authentication on the OpenAM side by any other means according to your preference.

See Also:

- Section 14.3.2, “Deploying the OpenAM Server”
- Section 14.3.3, “Adding the Authentication Plugin”
- Section 14.3.4, “Configuring a Realm in OpenAM User Interface”

Report a bug

14.3.2. Deploying the OpenAM Server

Procedure 14.2. Deploying the OpenAM Server

1. Obtain a copy of Tomcat 7 and extract it into a suitable location. This location will be referred to as **`TOMCAT_HOME`** further in the text.
2. Deploy OpenAM to Tomcat by copying the **`OPENAM_HOME/opensso/deployable-war/opensso.war`** archive to the **`TOMCAT_HOME/webapps/`** directory.

Note

The name of the WAR file is still **`opensso.war`**, even if it contains OpenAM, which is the successor of OpenSSO. The context of the OpenAM web application has also kept the name **`/opensso`**.

3. Change the default Tomcat port to avoid conflict with the default portal port. For the purpose of this example, change it to **`8888`** by editing the **`TOMCAT_HOME/conf/server.xml`** configuration file and replacing the **`8080`** port number with **`8888`**.

4. If the portal is running on the same machine as Tomcat, other Tomcat port numbers also need to be changed to avoid conflicts. These port numbers can be changed to any unassigned port numbers available on the machine. For example, change the admin port from **8005** to **8805** and the AJP port from **8009** to **8809** if the latter are not assigned.

Report a bug

14.3.3. Adding the Authentication Plugin

Procedure 14.3. Adding the Authentication Plug-in

1. Copy the contents of the **JPP_DIST/gatein-sso/opensso/plug-in/** directory to **TOMCAT_HOME/webapps/opensso/**. This will add:

   - the **AuthenticationPlugin.xml** file to the **TOMCAT_HOME/webapps/opensso/config/auth/default/** directory. The file contains the following configuration:

     Example 14.1. AuthenticationPlugin.xml Example Content

```
<?xml version='1.0' encoding="UTF-8"?>
<!DOCTYPE ModuleProperties PUBLIC "=//iPlanet//Authentication
Module Properties XML Interface 1.0 DTD//EN"
"jar://com/sun/identity/authentication/Auth_Module_Properties.
dtd">

<ModuleProperties moduleName="AuthenticationPlugin"
version="1.0">
  <Callbacks length="2" order="1" timeout="60" header="GateIn
OpenAM Login">
    <NameCallback>
      <Prompt>
        Username
      </Prompt>
    </NameCallback>
    <PasswordCallback echoPassword="false">
      <Prompt>
        Password
      </Prompt>
    </PasswordCallback>
  </Callbacks>
</ModuleProperties>
```

   - the **sso-opensso-plugin-<VERSION>.jar**, **sso-common-plugin-<VERSION>.jar** and **commons-httpclient-<VERSION>.jar** archives to the **TOMCAT_HOME/webapps/opensso/WEB-INF/lib** directory.

   - the **gatein.properties** file to the **TOMCAT_HOME/webapps/opensso/WEB-INF/classes/** directory. If necessary, change the values specified in this file to match the configuration of the portal instance. The values are used by the authentication plug-in to establish the REST connection to the portal.

2. Start Tomcat and verify that the OpenAM user interface at http://localhost:8888/opensso is accessible.

Report a bug

14.3.4. Configuring a Realm in OpenAM User Interface

Procedure 14.4. Configuring a Realm in OpenAM User Interface

1. Open http://localhost:8888/opensso.
2. Create the default configuration.
3. Login as **amadmin** and navigate to **Configuration → Authentication**.
4. Select the **Core** link, add a new value containing the **org.gatein.sso.opensso.plugin.AuthenticationPlugin** class name and click **Save**. This step is important for the portal SSO Authentication Plug-in to be available among other OpenAM authentication modules.
5. Go to the **Access control** tab and create a new realm called **gatein**.
6. Go to the **gatein** realm and click the **Authentication** tab. In the **Authentication chaining** section at the bottom of the tab, click **ldapService**. Here, change the selection from **Datastore**, which is the default module in the authentication chain, to **AuthenticationPlugin**. This ensures that authentication of the **gatein** realm will be performed using the portal REST service instead of the OpenAM LDAP server.
7. Still on the **Authentication** tab of the **gatein** realm, click the **All Core Settings** button. When the settings are displayed, change the **UserProfile** value from **Required** to **Dynamic**. This is needed because portal users are not initially present in the OpenAM LDAP server data store. The **Dynamic** value ensures that all users are automatically added to the data store after their first successful authentication.
8. Go to **Access control → Top level realm → Privileges → All authenticated users** and mark the following check boxes:
   - **Read and write access only for policy properties**
   - **Read and write access to all realm and policy properties**

   Adding these privileges allows REST access to the OpenAM server.
9. Repeat the previous step to increase the privileges also for the **gatein** realm.

Report a bug

14.4. Configuring the Platform as an OpenAM Client

On the portal server, you need to ensure proper configuration of single sign-on properties in the **JPP_HOME/standalone/configuration/gatein/configuration.properties** file. Locate the SSO section in this file and change/add properties in the section as follows:

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
```

```
gatein.sso.login.module.class=org.gatein.sso.agent.login.SSOLoginModule
gatein.sso.server.url=http://localhost:8888/opensso
gatein.sso.openam.realm=gatein
gatein.sso.portal.url=http://localhost:8080
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.OpenSSOLogoutF
ilter
gatein.sso.filter.logout.url=${gatein.sso.server.url}/UI/Logout
gatein.sso.filter.login.sso.url=${gatein.sso.server.url}/UI/Login?
realm=${gatein.sso.openam.realm}&goto=${gatein.sso.portal.url}/@@portal.
container.name@@/initiatessologin
```

Disable the new OpenAM UI

OpenAM 12.0 has a bug on the logout procedure through the new UI, called XUI which prevents the correct redirection. The workaround is to disable the newest UI (XUI) and use the classic UI.

To disable the new UI

1. Open the **OpenAM administration console** and access **Configuration**.
2. Select **Core** and uncheck the option **XUI interface**

Most of the properties are described for CAS integration.

Values of some properties are different with OpenAM, specifically the URLs for login/logout redirection, the logout filter class and the **gatein.sso.server.url** property, which points to the location of the OpenAM server. The **gatein.sso.openam.realm** value points to the realm created on the OpenAM server.

With all the previously described configuration, all links redirecting users to authentication pages will redirect them to the OpenAM authentication form. On the OpenAM side, users will be able to log in to the **gatein** realm using their portal credentials.

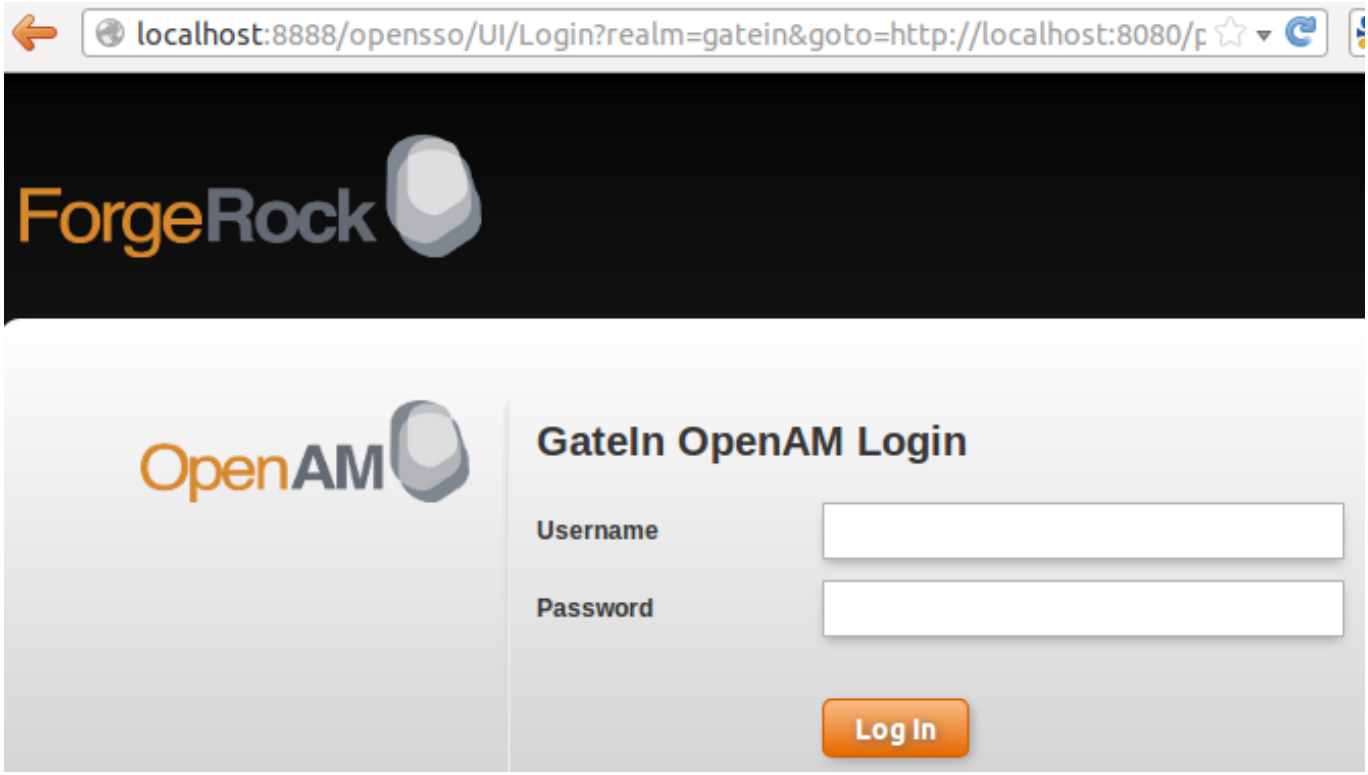


Figure 14.1. OpenAM Login Page

See Also:

- Section 12.3, “Central Authentication Service (CAS)”
- Section 14.3.4, “Configuring a Realm in OpenAM User Interface”

Report a bug

14.5. Cross-domain with OpenAM

14.5.1. Authenticating Cross-domain with OpenAM

The configuration described in Section 14.3.1, “Setting up OpenAM Server” and Section 14.4, “Configuring the Platform as an OpenAM Client” assumes that the portal and OpenAM are deployed on the same server or in the same DNS domain.

In such an environment, OpenAM adds the **`iPlanetDirectoryPro`** cookie for the shared domain to the client browser. It stores the authentication token in the cookie and performs redirection to the **`OpenSSOAgent`** in the portal. The agent is able to read the token from the cookie and perform its validation because the portal is running in the same domain.

This is not possible when the portal server and the OpenAM server are running in different domains and cannot share cookies. OpenAM provides the **`CDCServlet`** to solve this situation. Authenticated users can send requests to this servlet and the servlet responds with an encoded SAML message containing the SSO token and other information required to authenticate. The portal agent is then able to parse and validate the message, obtain the SSO token and establish the **`iPlanetDirectoryPro`** cookie for the domain where the portal is deployed. Once the OpenAM agent on the portal side has the token, it can perform further validation of this token and finish authentication of the user.

Detailed information about the **CDCServlet** servlet can be found in Sun OpenSSO Enterprise Deployment Planning Guide.

Report a bug

14.5.2. Configuring Cross-domain Authentication

Procedure 14.5. Cross-domain Authentication Configuration

For the purpose of this example, we will assume that the OpenAM server is deployed on the **opensso.mydomain.com** host and the portal server on the **portal.yourdomain.com** host.

Configure virtual hosts to simulate this configuration on a single machine. On Linux, this can be achieved by editing the **/etc/hosts** file and adding records similar to the following, with the IP addresses changed accordingly:

```
opensso.mydomain.com 192.168.2.7
portal.yourdomain.com 10.11.12.13
```

1. On the portal side, single sign-on configuration in the **configuration.properties** file must be specified as follows:

   ```
   # SSO
   gatein.sso.enabled=true
   gatein.sso.callback.enabled=${gatein.sso.enabled}
   gatein.sso.login.module.enabled=${gatein.sso.enabled}
   gatein.sso.login.module.class=org.gatein.sso.agent.login.SSOLoginMo
   dule
   gatein.sso.server.url=http://opensso.mydomain.com:8888/opensso
   gatein.sso.openam.realm=gatein
   gatein.sso.portal.url=http://portal.yourdomain.com:8080
   gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.OpenSSOLo
   goutFilter
   gatein.sso.filter.logout.url=${gatein.sso.server.url}/UI/Logout
   gatein.sso.filter.login.enabled=false
   gatein.sso.filter.login.openamcdc.enabled=true
   gatein.sso.filter.login.sso.url=${gatein.sso.server.url}/cdcservlet
   ```

   As we need to redirect requests to the **CDCServlet**, the **gatein.sso.filter.login.sso.url** property points to the URL of the servlet. It is also necessary to use a modified version of the **LoginRedirectFilter** interceptor. That is why the **gatein.sso.filter.login.openamcdc.enabled** value is changed to **true** and the **gatein.sso.filter.login.enabled** value is now set to **false**.

2. Access the OpenAM user interface at http://opensso.mydomain.com:8888/opensso and log in as **amadmin**.

3. Navigate to **Access Control** → **"gatein" realm** → **Agents** → **Web**.

4. Create a new web agent through the wizard using the following properties:

   - **Name**: GateInAgent
   - **Password**: A password of your choice.
   - **Configuration**: Centralized

- **`Server URL`**: http://opensso.mydomain.com:8888/opensso
- **`Agent URL`**: http://portal.yourdomain.com:8080

Creating this agent is required for the **`CDCServlet`** to work properly. If you have more portal servers on different hosts, you may want to create an agent for each of them. See the OpenAM Administration Guide available from http://openam.forgerock.org/doc/admin-guide/index.html for more details.

Report a bug

Chapter 15. Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO)

The Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) uses desktop credentials provided during desktop authentication to transparently authenticate a portal user through a web browser.

Let's illustrate this mechanism on a typical use case:

1. A user logs into their desktop computer with a login that is governed by an Active Directory domain.
2. The user launches a web browser to access a web application that is hosted on the portal and uses JBoss Negotiation.
3. The browser transfers the desktop credentials to the web application.
4. The portal uses background GSS messages to validate the user's Kerberos ticket with the Active Directory (or another Kerberos server).
5. The user experiences a seamless sign-on into the web application.

Report a bug

15.1. SPNEGO Server

15.1.1. Configuring the SPNEGO Server

This section describes the steps to set up a Kerberos server on Linux. The server will then be used for SPNEGO authentication against the portal. The procedure below only describes the basic steps to configure a SPNEGO server in a Linux environment. If you are already familiar with SPNEGO, or if you are using Windows and an Active Directory domain, you can proceed to Section 15.3, “Configuring SPNEGO” to see how SPNEGO can be integrated with the portal. Kerberos setup is also dependent on your Linux distribution, so the steps can be slightly different in your environment.

Procedure 15.1. Configuring the SPNEGO Server

1. Ensure correct host name to IP address mapping on the machine where Kerberos and the portal are running. For example, if the machine's IP address is **192.168.1.88** and you want it to be accessed under the **server.local.network** host name, add the following line to the **/etc/hosts** file:

```
192.168.1.88  server.local.network
```

Note

It is not recommended you use loopback addresses.

2. Install Kerberos with these packages: *krb5-admin-server*, *krb5-kdc*, *krb5-config*, *krb5-user*, *krb5-clients*, and *krb5-rsh-server*.
3. Edit the Kerberos configuration in the **/etc/krb5.config** file:

- Uncomment the following lines:

```
default_tgs_enctypes = rc4-hmac
default_tkt_enctypes = rc4-hmac
permitted_enctypes = rc4-hmac
```

- Add **local.network** as a default realm, add it to the list of realms, and remove the remaining realms. The resulting content of the file will be as follows:

```
[libdefaults]
    default_realm = LOCAL.NETWORK

# The following krb5.conf variables are only for MIT Kerberos.
    krb4_config = /etc/krb.conf
    krb4_realms = /etc/krb.realms
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

# The following encryption type specification will be used by
MIT Kerberos
# if uncommented.  In general, the defaults in the MIT Kerberos
code are
# correct and overriding these specifications only serves to
disable new
# encryption types as they are added, creating interoperability
problems.
#
# The only time when you might need to uncomment these lines and
change
# the enctypes is if you have local software that will break on
ticket
# caches containing ticket encryption types it doesn&apos;t know
about (such as
# old versions of Oracle).

    default_tgs_enctypes = rc4-hmac
    default_tkt_enctypes = rc4-hmac
    permitted_enctypes = rc4-hmac

# The following libdefaults parameters are only for Heimdal
Kerberos.
    v4_instance_resolve = false
    v4_name_convert = {
        host = {
            rcmd = host
            ftp = ftp
        }
        plain = {
            something = something-else
        }
    }
    fcc-mit-ticketflags = true

[realms]
```

```
    LOCAL.NETWORK = {
        kdc = server.local.network
        admin_server = server.local.network
    }

[domain_realm]
    .local.network = LOCAL.NETWORK
    local.network = LOCAL.NETWORK

[login]
    krb4_convert = true
    krb4_get_tickets = false
```

4. Modify KDC configuration.

   - Edit the **/etc/krb5kdc/kdc.conf** file as shown below:

```
[kdcdefaults]
kdc_ports = 750,88

[realms]
    LOCAL.NETWORK = {
        database_name = /var/lib/krb5kdc/principal
        admin_keytab = FILE:/etc/krb5.keytab
        acl_file = /etc/krb5kdc/kadm5.acl
        key_stash_file = /etc/krb5kdc/stash
        kdc_ports = 750,88
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        master_key_type = rc4-hmac
        supported_enctypes = rc4-hmac:normal
        default_principal_flags = +preauth
    }

[logging]
        kdc = FILE:/tmp/kdc.log
        admin_server = FILE:/tmp/kadmin.log
```

   - Create a KDC database.

```
sudo krb5_newrealm
```

   - Start the KDC and Kerberos servers.

```
sudo /etc/init.d/krb5-kdc restart
sudo /etc/init.d/krb-admin-server restart
```

5. Add principals and create keys.

   - Start an interactive **kadmin** session and create the necessary principals.

```
 sudo kadmin.local
```

   - Add the portal machine and keytab file.

```
addprinc -randkey HTTP/server.local.network@LOCAL.NETWORK
ktadd HTTP/server.local.network@LOCAL.NETWORK
```

- Add the default portal user accounts and enter a password for each of them.

```
addprinc john
addprinc demo
addprinc root
```

6. Issue the following command to test your setup:

```
kinit -A demo
```

- If everything is set up well, you are required to enter the password created for the **demo** user in the previous step.

  Without the **-A** option, the Kerberos ticket validation would involve reverse DNS lookups, which can be difficult to debug with certain network DNS configurations. This is a production level security feature and it is not necessary to use it in a development environment. In production environment, it is recommended to avoid using the **-A** option.

- After successful login to Kerberos, you can display your Kerberos ticket using the following command:

```
klist
```

- To log out and destroy your Kerberos ticket, issue the following command:

```
kdestroy
```

Report a bug

15.2. Configuring the SPNEGO Client

After performing the server configuration described in Section 15.1.1, “Configuring the SPNEGO Server”, you need to enable negotiated authentication in web browsers on client machines from which users will authenticate. This procedure describes how negotiated authentication can be enabled in Mozilla Firefox. For instructions on enabling negotiated authentication in a different browser, consult the browser's documentation.

Procedure 15.2. Enabling Negotiated Authentication in Mozilla Firefox

1. Start Mozilla Firefox and enter **about:config** into the address field.
2. Search for the **network.negotiate-auth** preferences and set the values as follows:

```
network.negotiate-auth.allow-proxies = true
network.negotiate-auth.delegation-uris = .local.network
network.negotiate-auth.gsslib (no-value)
network.negotiate-auth.trusted-uris = .local.network
network.negotiate-auth.using-native-gsslib = true
```

Report a bug

15.3. Configuring SPNEGO

The portal uses JBoss Negotiation to enable SPNEGO-based desktop single sign-on for the portal. The following procedure describes how to integrate SPNEGO with the portal.

Procedure 15.3. Configuring SPNEGO Integration

1. Modify the **# SSO** section of the ***JPP_HOME*/standalone/configuration/gatein/configuration.properties** file, replacing the original content with the following properties:

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=false
gatein.sso.skip.jsp.redirection=false
gatein.sso.login.module.enabled=false
gatein.sso.filter.login.sso.url=/@@portal.container.name@@/dologin
gatein.sso.filter.logout.enabled=false
gatein.sso.filter.initiatelogin.enabled=false
gatein.sso.valve.enabled=true
gatein.sso.valve.class=org.gatein.sso.spnego.GateInNegotiationAuthe
nticator
```

   The list below explains the meaning of individual properties:

   gatein.sso.enabled

   This is a general property used to inform the portal that clicking the **Sign in** link will redirect users to a URL ending with the **/portal/dologin** suffix.

   gatein.sso.callback.enabled

   This property can be set to **false** as REST callbacks are not required for SPNEGO integration.

   gatein.sso.skip.jsp.redirection

   This property must be set to **false** for SPNEGO integration, especially to ensure fallback to FORM authentication.

   gatein.sso.login.module.enabled

   This property can be set to **false** as a different set of login modules is needed for SPNEGO integration.

   gatein.sso.filter.login.sso.url

   This value ensures that clicking the **Sign in** link will redirect users to the **/portal/dologin** URL, which is a secured URL declared in the <security-constraint> section of **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/web.xml** file, allowing the **GateInNegotiationAuthenticator** valve to intercept the HTTP request.

   gatein.sso.filter.logout.enabled, gatein.sso.filter.initiatelogin.enabled

   These properties can be set to **false** as the logout filter and the **InitiateLoginFilter** are not needed for SPNEGO integration.

gatein.sso.valve.enabled

SPNEGO integration requires a custom Tomcat valve to intercept HTTP requests for secured URLs. The **SSODelegateValve** is defined in the **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/jboss-web.xml** file and is used only if this option is set to **true**. The purpose of the valve is to delegate the real work to another valve declared in the **gatein.sso.valve.class** property. This eliminates the need to edit configuration in the **jboss-web.xml** file.

gatein.sso.valve.class

The delegate valve for SPNEGO is **org.gatein.sso.spnego.GateInNegotiationAuthenticator**. It is used to establish identity of the client by exchanging several handshakes with client browser. The valve is able to obtain and parse an SPNEGO token and resend it to the JAAS layer, where login modules (especially the **SPNEGOLoginModule**) are able to verify the validity of the wrapped Kerberos token and establish user identity at the portal layer.

2. Modify configuration of the **security** subsystem in the ***JPP_HOME*/standalone/configuration/standalone.xml** file:

    a. Rename the existing **gatein-domain** security domain to **gatein-form-auth-domain**.

    b. Add a new definition of the **gatein-domain** security domain.

    c. Add a new security domain called **host**. Values of the following properties need to be specified in accordance with your environment setup:

    principal

    The HTTP server principal specified in your Kerberos keytab.

    For example, **LOCAL.NETWORK** Kerberos realm, **server.local.network** host, and the **HTTP/server.local.network@LOCAL.NETWORK** principal.

    keyTab

    The path to the file with your Kerberos keytab.

    For example the **/etc/krb5.keytab** location. This file must be readable by the user under whose account the portal is executed. Generally, Kerberos keytab files can not be read by normal operating system users because they contain encrypted keys of server principals. Consult Kerberos documentation for more details.

    debug

    Enabling this option adds more log messages into the portal console and log file. It is useful to have this option enabled during configuration, but it is advised to disable it in production to avoid unnecessarily large server logs.

The code extract below shows the expected result of the security domain configuration.

```
<security-domain name="gatein-form-auth-domain" cache-
```

```
type="default">
  <authentication>
    <login-module
code="org.gatein.sso.integration.SSODelegateLoginModule"
flag="required">
      <module-option name="enabled"
value="${gatein.sso.login.module.enabled}" />
      <module-option name="delegateClassName"
value="${gatein.sso.login.module.class}" />
      <module-option name="portalContainerName" value="portal" />
      <module-option name="realmName" value="gatein-domain" />
      <module-option name="password-stacking" value="useFirstPass"
/>
    </login-module>
    <login-module
code="org.exoplatform.services.security.j2ee.JBossAS7LoginModule"
flag="required">
      <module-option name="portalContainerName" value="portal"/>
      <module-option name="realmName" value="gatein-domain"/>
    </login-module>
  </authentication>
</security-domain>

<security-domain name="gatein-domain" cache-type="default">
  <authentication>
    <login-module code="org.gatein.sso.spnego.SPNEGOLoginModule"
flag="requisite">
      <module-option name="password-stacking"
value="useFirstPass"/>
      <module-option name="serverSecurityDomain" value="host"/>
      <module-option name="removeRealmFromPrincipal"
value="true"/>
      <module-option name="usernamePasswordDomain" value="gatein-
form-auth-domain"/>
    </login-module>
    <login-module
code="org.gatein.sso.agent.login.SPNEGORolesModule"
flag="required">
      <module-option name="password-stacking"
value="useFirstPass"/>
      <module-option name="portalContainerName" value="portal"/>
      <module-option name="realmName" value="gatein-domain"/>
    </login-module>
  </authentication>
</security-domain>

<security-domain name="host">
  <authentication>
    <login-module
code="com.sun.security.auth.module.Krb5LoginModule"
flag="required">
      <module-option name="storeKey" value="true" />
      <module-option name="useKeyTab" value="true" />
      <module-option name="principal"
value="HTTP/server.local.network@LOCAL.NETWORK" />
      <module-option name="keyTab" value="/etc/krb5.keytab" />
```

```
        <module-option name="doNotPrompt" value="true" />
        <module-option name="debug" value="true" />
      </login-module>
    </authentication>
  </security-domain>
```

3. While logged in as a user with read permissions for the Kerberos keytab file, start the portal using the following command: **`./standalone.sh -Djava.security.krb5.realm=LOCAL.NETWORK -Djava.security.krb5.kdc=server.local.network -b server.local.network`**

See Also:

- Section 15.1.1, “Configuring the SPNEGO Server”

Report a bug

15.4. Testing SPNEGO Configuration

Once you have performed the configuration above, you can follow this procedure to verify that the configured authentication is functional.

Procedure 15.4. Testing the SPNEGO Configuration

1. Log in to Kerberos using the following command:

```
kinit -A demo
```

2. In a browser, access http://server.local.network:8080/portal and click the **`Sign In`** link in the top menu of the portal. If the authentication is configured correctly, you will be automatically signed in as the **`demo`** user.

3. Destroy the previously obtained Kerberos ticket using the following command:

```
kdestroy
```

4. In the browser, log out and log back in. In case of correct configuration, you will be redirected to the login screen of the portal. This happens because you no longer have an active Kerberos ticket and a fall back to the standard FORM authentication occurred.

Report a bug

15.5. Disabling Fallback to FORM Authentication

As demonstrated in Section 15.4, “Testing SPNEGO Configuration”, users trying to sign in without a valid Kerberos ticket are automatically redirected to the portal authentication page. There, they can perform standard FORM authentication using their user name and password.

If you want to disable FORM authentication to allow only users with a valid Kerberos ticket to sign in, you need to comment out the ***`usernamePasswordDomain`*** option in the **`SPNEGOLoginModule`** configuration in the ***`JPP_HOME`***`/standalone/configuration/standalone.xml` file.

```
<login-module
```

```
    code="org.gatein.sso.spnego.SPNEGOLoginModule" flag="requisite">
    <module-option name="password-stacking" value="useFirstPass"/>
    <module-option name="serverSecurityDomain" value="host"/>
    <module-option name="removeRealmFromPrincipal" value="true"/>
<!--<module-option name="usernamePasswordDomain" value="gatein-form-auth-
domain"/>-->
</login-module>
```

Report a bug

15.6. Enabling Logging

To enable logging of events related to SPNEGO authentication, you can add the following entries to the **`logging`** subsystem in the ***`JPP_HOME`*`/standalone/configuration/standalone.xml`** file:

```
<logger category="org.gatein.sso">
    <level name="TRACE"/>
</logger>
<logger category="org.jboss.security.negotiation">
    <level name="TRACE"/>
</logger>
```

Increased logging of Kerberos events to standard output can be enabled by launching the portal with the **`-Dsun.security.krb5.debug=true`** option.

```
./standalone.sh -Djava.security.krb5.realm=LOCAL.NETWORK -
Djava.security.krb5.kdc=server.local.network -b server.local.network -
Dsun.security.krb5.debug=true
```

Report a bug

Chapter 16. Single Sign-on in a Cluster

In a cluster, the JBoss SSO valve can be used to authenticate a user on one portal node and have that authentication automatically carried across to other nodes in the cluster.

The JBoss SSO valve is enabled by default through the following JBoss Web subsystem configuration entry in the **JPP_HOME/standalone/configuration/standalone-ha.xml** file:

```
<sso cache-container="web" cache-name="sso" reauthenticate="false" />
```

When a load balancer is used in a cluster, no further configuration is needed to set up single sign-on. All portal servers in the cluster are accessed through the same URL, which is the URL of the load balancer. Automatic single sign-on is performed when the load balancer redirects client requests to individual nodes in the cluster.

Report a bug

16.1. Clustered Single Sign-on in a Shared DNS Domain

If multiple portal servers are accessed through different URLs in the same DNS domain, single sign-on can be configured by adding the ***domain*** parameter to the ***sso*** configuration entry.

```
<sso cache-container="web" cache-name="sso" reauthenticate="false"
domain="yourdomain.com"/>
```

The parameter must be added to the entry on all servers in the cluster and the name of the shared DNS domain must be specified as its value. This configuration ensures that the ***JSESSIONIDSSO*** cookie will be scoped to the specified domain, which is otherwise scoped only to the host where the initial authentication was performed.

Report a bug

16.1.1. Configuring and Testing Single Sign-on in a Shared DNS Domain

This procedure demonstrates the configuration and testing of single sign-on for two portal server instances running in a shared domain on a single physical Linux machine.

It is expected that each instance is installed in a separate directory in the machine's file system, and that the **192.168.210.101** and **192.168.210.102** virtual IP addresses are available on the machine.

1. Map the IP addresses to domain names within the same domain by adding the following lines to the **/etc/hosts** file:

   ```
   192.168.210.101 machine1.yourdomain.com
   192.168.210.102 machine2.yourdomain.com
   ```

2. Open the ***JPP_HOME*/standalone/configuration/standalone-ha.xml** file on both instances, add the ***domain*** parameter to the ***sso*** entry and specify the name of the shared DNS domain in its value:

```
<sso cache-container="web" cache-name="sso" reauthenticate="false"
domain="yourdomain.com"/>
```

3. By default, the **standalone-ha.xml** file is configured to use a shared H2 database, which is intended to be used only for testing purposes. Start the database by issuing the following command in the *JPP_HOME* directory of the first instance:

```
java -cp modules/com/h2database/h2/main/h2-<VERSION>.jar
org.h2.tools.Server
```

4. Start the first instance by issuing the following command in its **JPP_HOME/bin/** directory:

```
./standalone.sh -b machine1.yourdomain.com -c standalone-ha.xml -
Djboss.node.name=node1
```

5. Start the second instance by issuing the following command in its **JPP_HOME/bin/** directory:

```
./standalone.sh -b machine2.yourdomain.com -c standalone-ha.xml -
Djboss.node.name=node2
```

6. Access the first instance at http://machine1.yourdomain.com:8080/portal and log in as a user.
7. Access the second instance at http://machine2.yourdomain.com:8080/portal. When the page loads, you will be automatically logged in with the same user account that you used on the first server.
8. Log out on any of the two instances. Then switch to the other instance and verify that you have been logged out of it as well.

Report a bug

16.2. Reauthentication

The JBoss SSO valve can also be used to authenticate with any other web application. If that application uses the same roles as the main portal instance, no further configuration is required. Because the JBoss SSO valve includes the same JAAS principal in all HTTP requests, even in requests to other web applications, matching roles ensure successful authentication with those applications.

To enable single sign-on authentication with an application that uses different roles, set the ***reauthenticate*** parameter of the ***sso*** JBoss Web subsystem configuration entry to **true**.

```
<sso cache-container="web" cache-name="sso" reauthenticate="true" />
```

The **true** value ensures that re-authentication with user credentials will be performed against the web application's security domain in each HTTP request. This will enforce creation of a new principal with updated roles for the web application. As user credentials are used for authentication in this case, it is required that the same user credentials exist in both the web application and the portal instance.

Report a bug

Chapter 17. LDAP Integration

Lightweight Directory Access Protocol (LDAP) is a set of open protocols used to access centrally stored information over a network. It is based on the X.500 standard for directory sharing, but is less complex and resource-intensive.

Using a client/server architecture, LDAP provides a reliable means to create a central information directory accessible from the network. When a client attempts to modify information within this directory, the server verifies the user has permission to make the change, and then adds or updates the entry as requested. To ensure the communication is secure, the Secure Sockets Layer (*SSL*) or Transport Layer Security (*TLS*) cryptographic protocols can be used to prevent an attacker from intercepting the transmission.

LDAP provides the protocols required to manage the data stored in a Directory Server. A Directory Server contains information about resources available (user accounts and printers for example) and their location on the network.

See https://access.redhat.com/knowledge/articles/119833 page for a list of supported directory servers.

Note

For ease of readability the following section uses the notational device *ID_HOME* to represent the file path **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/organization/`**. This directory is the root of all portal identity-related configuration.

The portal includes several example LDAP configuration **`.xml`** files and **`.ldif`** (LDAP Data Interchange Format) data files. These examples are in the ***`ID_HOME`*`/picketlink-idm/examples`** directory and can be deployed in a testing environment to assist in configuring LDAP.

Report a bug

17.1. Setting up LDAP

Procedure 17.1. LDAP Setup

1. Install your **LDAP** server by following the installation instructions provided for the product you are using.

   See the Red Hat Directory Server *Installation Guide*, available from https://access.redhat.com/knowledge/docs/Red_Hat_Directory_Server/ for comprehensive installation guidance.

   If you are using a third-party directory server (**OpenDS**, **OpenLDAP** or **Microsoft Active Directory**), see the appropriate user documentation for that product.

   The following values provide an example of working configuration settings for the different Directory Servers:

   Table 17.1. Table Title

Directory Server	root user Distinguished Name(DN)	Password	Port	Admin Port	Base DN	Database Population	SSO/TLS
RHDS and OpenDS	MSAD	OpenLDAP					
cn=Directory manager	CN=Users	cn=Manager,dc=example,dc=com					
password		secret					
1389		1389					
4444							
dc=example,dc=com		dc=example,dc=com					
"Only create the base entry"							
no SSO, no TLS							

2. Optionally, import an **`ldif`** file to populate the Directory Server.
3. Start the Directory Server.

Report a bug

17.2. LDAP in Read only mode

If LDAP is configured to operate in read-write mode, changes to user and group information made in the portal platform is written back to the directory server. If LDAP is operating in read-only mode, existing user and group information is consumed from the directory server, and all new user data entries created using the Portal User Interface are stored in the portal database. The only exception is that passwords updated through the user interface will also be propagated into the appropriate LDAP entry.

Report a bug

17.2.1. Setting up LDAP Read-only Mode

Procedure 17.2. Set up LDAP Read-only Mode

1. Open the ***`ID_HOME`*`/idm-configuration.xml`** file.

   The portal uses the PicketLink IDM framework as the underlying identity storage system, therefore the configuration uses dedicated PicketLink settings.

2. Comment the default PicketLink **`config`** value:

```
<value>war:/conf/organization/picketlink-idm/picketlink-idm-
config.xml</value>
```

3. Uncomment the appropriate sample configuration values as described below, depending on which Directory Server you are implementing:
   - Section 17.2.2, “Setting up Red Hat Directory Server or OpenDS”
   - Section 17.2.3, “Setting up Microsoft Active Directory”
   - Section 17.2.4, “Setting up OpenLDAP”
4. To use a different LDAP server or directory data, edit the **`DS-specific.xml`** file you uncommented in the relevant sub-procedure (3a) above, and change the values to suit your requirements.
5. Start the server.
6. Navigate to the portal homepage (http://localhost:8080/portal) and log in as an administrator.
7. Navigate to **Group → Organization → Users and groups management**.
8. Create a new group called *acme* under the root node.
   - **For RHDS, OpenDS and OpenLDAP**:

     Create two sub-groups called *roles* and *organization_units*.
   - **For MSAD:**

     Create a subgroup called *roles*.
9. By default, users mapped from LDAP will be members of LDAP groups, but will not be members of the **`/platform/users`** group. This means that mapped users will not be treated as full portal users. To fix this issue, one of the following approaches is required:
   a. Add LDAP users manually to **`/platform/users`** group
   b. Configure **`CustomMembershipLoginModule`** in the JAAS login modules stack. This is special login module, which will add every logged user into group /platform/users after successful login of this user. Two additional login modules (InitSharedStateLoginModule and SharedStateLoginModule) are required for this approach.

Users defined in LDAP must be visible in "*Users and groups management*" and groups from LDAP must be present as children of */acme/roles* and */acme/organization_units*.

More information about configuration can be found in the PicketLink IDM 1.x Community Documentation.

See Also:

- Section 17.4.1, “Example 1 LDAP Configuration”
- Section 17.4.2, “Example 2 Read-only groupType Mappings”
- Section 17.4.3, “Example 3 Default groupType Mappings”
- Section 7.3.1, “Types of Login Modules”

Report a bug

17.2.2. Setting up Red Hat Directory Server or OpenDS

Procedure 17.3. Red Hat Directory Server or Open Directory Server

1. Uncomment the line under "*Read Only "ACME" LDAP Example*":

```
<!--Read Only "ACME" LDAP Example-->
<value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-LDAP-acme-config.xml</value>
```

2. Uncomment the ***groupTypeMappings*** under "*Uncomment for ACME LDAP example*":

```
<!-- Uncomment for ACME LDAP example -->
<entry>
 <key><string>/acme/roles/*</string></key>
 <value><string>acme_roles_type</string></value>
</entry>
<entry>
 <key><string>/acme/organization_units/*</string></key>
 <value><string>acme_ou_type</string></value>
</entry>
```

3. Return to Section 17.2.1, “Setting up LDAP Read-only Mode”.

See Also:

- Section 17.4.2, “Example 2 Read-only groupType Mappings”

Report a bug

17.2.3. Setting up Microsoft Active Directory

Procedure 17.4. Microsoft Active Directory

1. Uncomment the line under "*MSAD Read Only "ACME" LDAP Example*":

```
 <!--MSAD Read Only "ACME" LDAP Example-->
 <value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-msad-readonly-config.xml</value>
```

2. Uncomment the ***groupTypeMappings*** under "*Uncomment for MSAD ReadOnly LDAP example*":

```
<!-- Uncomment for MSAD ReadOnly LDAP example -->
<entry>
  <key><string>/acme/roles/*</string></key>
  <value><string>msad_roles_type</string></value>
</entry>
```

3. Return to Section 17.2.1, “Setting up LDAP Read-only Mode”.

See Also:

See Also:

- Section 17.4.2, “Example 2 Read-only groupType Mappings”

Report a bug

17.2.4. Setting up OpenLDAP

Procedure 17.5. configure OpenLDAP

1. Install the LDAP server.
2. Uncomment the line under "*OpenLDAP ReadOnly "ACME" LDAP Example*":

```
<!--OpenLDAP ReadOnly "ACME" LDAP Example-->
<value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-openLDAP-acme-config.xml</value>
```

3. Uncomment the ***groupTypeMappings*** under "*Uncomment for ACME LDAP example*":

```
<!-- Uncomment for ACME LDAP example -->
<entry>
 <key><string>/acme/roles/*</string></key>
 <value><string>acme_roles_type</string></value>
</entry>
<entry>
 <key><string>/acme/organization_units/*</string></key>
 <value><string>acme_ou_type</string></value>
</entry>
```

4. Return to Section 17.2.1, “Setting up LDAP Read-only Mode”

See Also:

- Section 17.1, “Setting up LDAP”
- Section 17.4.2, “Example 2 Read-only groupType Mappings”

Report a bug

17.3. LDAP as Default Store

Follow the procedure below to set LDAP up as the default identity store for the portal. All default accounts and some of groups that come with the portal will be created in the LDAP store.

The LDAP server will be configured to store part of the portal group tree. This means that groups under specified part of the tree will be stored in directory server while all others will be stored in database.

See Also:

- Section 17.1, “Setting up LDAP”
- Section 17.4.3, “Example 3 Default groupType Mappings”
- Section 17.4.1, “Example 1 LDAP Configuration”

Report a bug

17.3.1. Setting up LDAP as Default Identity Store

Procedure 17.6. Set up LDAP as Default Identity Store

1. Install the LDAP server.
2. Open the ***ID_HOME*/idm-configuration.xml** file.

   The portal uses the PicketLink IDM framework as the underlying identity storage system, therefore all the configurations use dedicated PicketLink settings.
3. Comment out the default Picketlink **config** value: ***war:/conf/organization/picketlink-idm/picketlink-idm-config.xml***
4. Complete the steps in the procedure that relates to the chosen LDAP server:
   - Section 17.3.2, “Setting up RHDS and OpenDS”
   - Section 17.3.3, “Setting up Microsoft Active Directory”
   - Section 17.3.4, “Setting up OpenLDAP”
5. Uncomment the ***groupTypeMappings*** under "*Uncomment for sample LDAP configuration*":

```
<entry>
 <key><string>/platform/*</string></key>
 <value><string>platform_type</string></value>
</entry>
<entry>
 <key><string>/organization/*</string></key>
 <value><string>organization_type</string></value>
</entry>
```

6. Uncomment ***ignoreMappedMembershipTypeGroupList*** under *Uncomment for sample LDAP config*.

```
<value>
  <string>/platform/*</string>
</value>
<value>
  <string>/organization/*</string>
</value>
```

Important

If this configuration is not uncommented, memberships will be used as both relationships and roles, which may cause duplicated records in the portal management interface.

Normally, the same roles being used for LDAP mapping need to be uncommented. User memberships under the specified groups will be created in PicketLink IDM only as relationships, and not as roles.

7. To use a different LDAP server or directory data, edit the DS-specific **.xml** file you uncommented in *Step 4*, and change the values to suit your requirements.

8. **Result**

   All portal groups under **/platform** and **/organization** groups (for example **/platform/users**, **/platform/administrators**, **/organization/management/executive-board**) are mapped to the LDAP tree. The location of groups in the LDAP tree are configurable through the parameter ***ctxDNs*** in the PicketLink IDM configuration file.

Report a bug

17.3.2. Setting up RHDS and OpenDS

Procedure 17.7. For RHDS and OpenDS

1. Expose the entry under "*Sample LDAP config*":

```
<!--Sample LDAP config-->
<value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-LDAP-config.xml</value>
```

2. Return to Section 17.3.1, “Setting up LDAP as Default Identity Store”

Report a bug

17.3.3. Setting up Microsoft Active Directory

Procedure 17.8. For MSAD

1. Expose the entry under "*MSAD LDAP Example*":

```
<!--MSAD LDAP Example-->
<value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-msad-config.xml</value>
```

2. To enable SSL encryption, perform the following sub-steps:

   a. Open the ***ID_HOME*/picketlink-idm/examples/picketlink-idm-msad-config.xml**.

   b. Ensure the following entries are uncommented and that the path to the **truststore** file and password are correct:

```
<option>
  <name>customSystemProperties</name>
  <value>javax.net.ssl.trustStore=/path/to/truststore</value>
  <value>javax.net.ssl.trustStorePassword=password</value>
</option>
```

   You can import a custom certificate by replacing the *certificate* and *truststore* details in the following command:

```
keytool -import -file certificate -keystore truststore
```

3. Return to Section 17.3.1, “Setting up LDAP as Default Identity Store”.

Report a bug

17.3.4. Setting up OpenLDAP

Procedure 17.9. For OpenLDAP

1. Expose the entry under "*OpenLDAP LDAP config*":

```
<!--OpenLDAP LDAP config-->
<value>war:/conf/organization/picketlink-idm/examples/picketlink-
idm-openLDAP-config.xml</value>
```

2. Return to Section 17.3.1, “Setting up LDAP as Default Identity Store”

Report a bug

17.4. Integration Examples

17.4.1. Example 1 LDAP Configuration

The following settings are stored in the PicketLink configuration file that is nominated in the **`idm-configuration.xml`** file of your deployment (under the ***`config`*** parameter of the ***`PicketLinkIDMService`*** component):

This file could be:

- The default **`picketlink-idm-config.xml`**.
- One of the three example configuration files discussed in Section 17.2.1, “Setting up LDAP Read-only Mode”:

  `picketlink-idm-LDAP-acme-config.xml`
  `picketlink-idm-msad-readonly-config.xml`
  `picketlink-idm-openLDAP-acme-config.xml`

- A custom file created by modifying one of the above files.

Configuration Options

ctxDNs

This is the DN that will be used as context for *IdentityObject* searches. More than one value can be specified.

Some examples are:

- ou=People,o=acme,dc=example,dc=com
- ou=Roles,o=acme,dc=example,dc=com
- ou=OrganizationUnits,o=acme,dc=example,dc=com
- **MSAD**: CN=Users,DC=test,DC=domain (in two places).

providerURL

The LDAP server connection URL. Formatted as "ldap://*<HOST>*:*<PORT>*". The default setting is: *ldap://localhost:1389*.

MSAD: Should use SSL connection (ldaps://*<HOST>*:636) for password update or creation to work.

adminDN

The LDAP entry used to connect to the server.

Some possible values are:

- **RHDS or OpenDS**: cn=Directory Manager
- **OpenLDAP**: cn=Manager,dc=my-domain,dc=com
- **MSAD**: TEST\Administrator

adminPassword

The password associated with the **adminDN**.

customSystemProperties

This option defines the values needed to use SSL encryption with LDAP.

Report a bug

17.4.2. Example 2 Read-only groupType Mappings

The ***groupTypeMappings*** exposed in the **idm-configuration.xml** file correspond to ***identity-object-type*** values defined in the DS-specific configuration file (referenced in *Sub-step 3a* of the DS-specific procedure above).

For RHDS, OpenDS and OpenLDAP the **picketlink-idm-LDAP-acme-config.xml** and **picketlink-idm-openLDAP-acme-config.xml** files contain the following values:

```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/readonly-opends.xml" parse="text"/>
```

Comment #1: The PicketLink IDM configuration file dictates that users and those two group types be stored in LDAP.

Comment #2: An additional option defines that nothing else (except password updates) should be written there.

All groups under **/acme/roles** will be stored in PicketLink IDM with the **acme_roles_type** group type name and groups under **/acme/organization_units** will be stored in PicketLink IDM with **acme_ou_type** group type name.

For MSAD, the ***identity-object-types*** values in **picketlink-idm-msad-readonly-config.xml** change to:

```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/readonly-msad.xml" parse="text"/>
```

The difference is that this configuration maps only one group type and points to the same container in LDAP for both users and mapped groups.

Report a bug

17.4.3. Example 3 Default groupType Mappings

The ***groupTypeMappings*** exposed in the **idm-configuration.xml** file correspond to ***identity-object-type*** values defined in the DS-specific configuration file (referenced in *Sub-step 3a* of the DS-specific procedure above).

All of the supported LDAP configurations use the following values when implemented as the default identity store:

```
   <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/default-ldap.xml" parse="text"/>
```

Comment #1: The ***groupTypeMappings*** define that all groups under ***/platform*** should be stored in PicketLink IDM with the ***platform_type*** group type name and groups under ***/organization*** should be stored in PicketLink IDM with ***organization_type*** group type name.

The PicketLink IDM configuration file repository maps users and those two group types as stored in LDAP.

Report a bug

Chapter 18. Security Assertion Markup Language (SAML2)

About SAML2

SAML (Security Assertion Markup Language) is an Oasis standard for exchanging authentication and authorization data between security domains. SAML 2.0 is an XML-based protocol that uses security tokens containing assertions to pass information about a principal (usually an end user) between an identity provider and a web service. SAML 2.0 enables web-based authentication and authorization scenarios including single sign-on (SSO). General information relating to SAML2 is located in PDF form at http://docs.oasis-open.org/security/saml/v2.0/. In Red Hat JBoss Portal, SAML2 support is provided by the PicketLink Federation. Read the documentation hosted at https://docs.jboss.org/author/display/PLINK/SAML+v2.0 for an excellent overview of what configuration is required for SAML2.

About Identity Provider (IDP)

An Identity Provider is defined by OASIS as a kind of provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, such as with web browser profiles. A Federation in this definition means an association comprising any number of service providers and identity providers.

About Identity Provider (IDP)

An Identity Provider is defined by OASIS as a kind of provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, such as with web browser profiles. A Federation in this definition means an association comprising any number of service providers and identity providers.

What is a Service Provider (SP)

A Service Provider is defined by OASIS as a role donned by a system entity, where the system entity provides services to principals or other system entities.

About Assertion

In SAML2, an assertion is a packet of security information. Each assertion contains statements used by service providers to make access-control decisions. The assertion statements provided by SAML2 include: authentication; attribute; and authorization decision statements.

Report a bug

18.1. Authentication in SAML2

The following workflow phases apply when Red Hat JBoss Portal is used as a SAML2 Service Provider (SP).

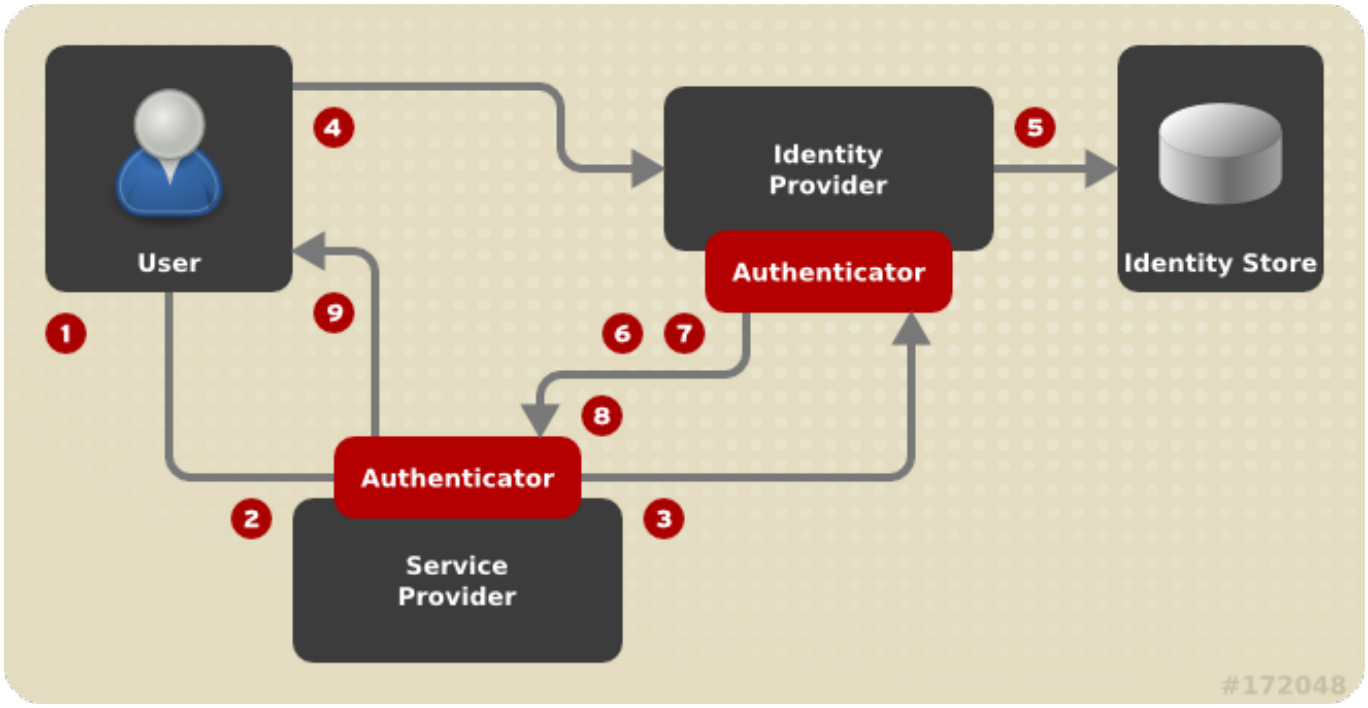


Figure 18.1. SAML2 Authentication Workflow

When a user attempts to authenticate on a SAML2-enabled system, the request is handled by an authenticator that sits atop the Service Provider (SP).

If a user attempts to access a protected resource, for example http://localhost:8080/portal/dologin, the authenticator verifies whether the user has a SAML assertion known to the SP. If the user is authenticated, the authenticator redirects the user to the requested resource.

If the user is not yet authenticated, the following work flow is initiated.

Phase 1

User requests access to a protected resource.

Phase 2

The authenticator verifies whether the user has a SAML assertion known to the SP.

Phase 3

Red Hat JBoss Portal (SP) determines that it does not have a valid SAML assertion for the user.

The SAML request is encapsulated into a HttpRequest and is redirected to the Identity Provider (IDP) using the SAML Redirect Binding (sent as a Base64, URL-encoded, GET request parameter).

Phase 4

The IDP returns a login screen to the user, prompting for valid credentials.

Phase 5

The IDP receives the user credentials through a JAAS login module (SAML2IdpLoginModule). The login module sends a callback request through the REST API, back to Red Hat JBoss Portal to check the user exists in the identity store.

Phase 6

After the user is verified to exist in the identity store by the login module, the IDP authenticator issues a SAML assertion ticket that contains all roles for the user.

Phase 7

The assertion ticket is encapsulated in a HttpRequest, and is redirected back to the Service Provider (SP).

Phase 8

The SP decodes the HttpRequest. The SP login module (SAML2IntegrationLoginModule) parses the authenticated username, and determines whether the assertion is valid according to the known roles the SP has about the user.

The login module then creates an identity object for the user, and registers the user in the IdentityRegistry.

Phase 9

The user is now successfully authenticated, and is redirected back to the secure resource. The secure resource will then redirect the user to Red Hat JBoss Portal as an authenticated user.

If the user needs to authenticate against a different SP application within the same browser session, credentials are not required to re-authenticate because the user is already known to the IDP.

An example of this would include another instance of Red Hat JBoss Portal on a different host, or a completely different web application within the same IDP federation.

Report a bug

18.2. Configuring a Basic SAML2 Instance

18.2.1. SAML2 Configuration Scenario

This basic scenario describes how to configure the necessary **`configuration.properties`** file on each virtual host portal instance. The scenario uses a bundled test keystore, which is not suitable for production environments.

See Also:

- Section 18.4, “Implementing Keystores”

Report a bug

18.2.2. Configuring a SAML2 Service Provider

Prerequisites

- Two available virtual hosts running on the local machine.
- One instance of the portal deployed to a directory referred to as *JPP_SP_DIST*.
- A separate instance of the portal deployed to another directory referred to as *JPP_IDP_DIST*.

Procedure 18.1. Configure the SAML2 SP

- Open ***JPP_SP_DIST*/standalone/configuration/gatein/configuration.properties** and add the following configuration parameters.

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
gatein.sso.login.module.class=org.gatein.sso.agent.login.SAML2Integrati
onLoginModule
# Comment #1
gatein.sso.filter.login.sso.url=/@@portal.container.name@@/dologin
# Comment #2
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.SAML2LogoutF
ilter
gatein.sso.filter.initiatelogin.enabled=false
gatein.sso.valve.enabled=true
# Comment #3
gatein.sso.valve.class=org.picketlink.identity.federation.bindings.tomc
at.sp.ServiceProviderAuthenticator
# Comment #4
gatein.sso.saml.config.file=../gatein/gatein.ear/portal.war/WEB-
INF/classes/sso/saml/picketlink-sp.xml
# Comment #5
gatein.sso.idp.host=www.idp.com
# Comment #6
gatein.sso.idp.url=http://${gatein.sso.idp.host}:8080/portal/dologin
# Comment #7
gatein.sso.sp.url=http://www.sp.com:8080/portal/dologin
# Comment #8
# WARNING: This bundled keystore is only for testing purposes. Generate
and use your own keystore in production!
gatein.sso.picketlink.keystore=/sso/saml/jbid_test_keystore.jks
```

Comment #1

For integration as SAML2 SP, use the login module class **org.gatein.sso.agent.login.SAML2IntegrationLoginModule**, which acts as a JAAS delegate from **SSODelegateLoginModule**.

Comment #2

The specific filter **class org.gatein.sso.agent.filter.SAML2LogoutFilter** is needed to enable support for SAML2 single logout. SAML2 single logout will take place and will log you out from both portal IDP and SP servers.

1. A user clicks **Sign out**.
2. **SAML2LogoutFilter** will redirect them to the SAML2 IDP application where they will be logged out.
3. The IDP application will also manage the logout from all other SP applications.
4. The user will be redirected to the portal, and logged out.

Comment #3

There is a special valve; **`org.picketlink.identity.federation.bindings.tomcat.sp.ServiceProviderAuthenticator`** provided by the Picketlink federation library. This valve manages the creation and parsing of **`SAMLRequest`** and **`SAMLResponse`** messages and so is vital for a successful SAML2 integration.

Comment #4

This element points to the location of the SAML2 SP configuration file. The path is relative to the portal WAR application. The relative path to the file is **`../gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml/picketlink-sp.xml`**. For this simple scenario, there is no need to edit this file. However, in a production environment, you will likely need to generate and use your own keystore and thus you must then configure the use of those keys in this file.

Comment #5

This points to the host serving the SAML2 IDP.

Comment #6

This element points to the URL of the IDP application. This guide assumes that you will use another instance of the portal as the SAML2 IDP, so the request path will be /portal/dologin in this case.

Comment #7

This points to URL of this GateIn Portal, which will be used as SAML2 SP.

Comment #8

This points to the location of the keystore file. It is relative to the classpath of the portal WAR application. The absolute location is **`JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml/jbid_test_keystore.jks`**.

See Also:

- Section 18.3, “Disabling SAML2 Single Logout”
- Section 18.4, “Implementing Keystores”

Report a bug

18.2.3. Configuring a SAML2 Identity Provider

Prerequisites

- Two available virtual hosts running on the local machine.
- One instance of the portal deployed to a directory referred to as referred to as *JPP_SP_DIST*.
- A separate instance of the portal deployed to another directory referred to as *JPP_IDP_DIST*.

Procedure 18.2. Configuring the portal as a SAML2 Identity Provider

1. Open ***JPP_IDP_DIST*/standalone/configuration/gatein/configuration.properties** and add the following configuration parameters.

```
# SSO
# Comment #1
gatein.sso.enabled=false
gatein.sso.valve.enabled=true
# Comment #2
gatein.sso.valve.class=org.gatein.sso.saml.plugin.valve.PortalIDPWe
bBrowserSSOValve
# Comment #3
gatein.sso.saml.config.file=/WEB-INF/conf/sso/saml/picketlink-
idp.xml
# Comment #4
gatein.sso.idp.url=http://www.idp.com:8080/portal/dologin
# Comment #5
gatein.sso.idp.listener.enabled=true
# Comment #6
gatein.sso.sp.domains=sp.com
# Comment #7
gatein.sso.sp.host=www.sp.com
# Comment #8
# WARNING: This bundled keystore is only for testing purposes.
Generate and use your own keystore in production!
gatein.sso.picketlink.keystore=/sso/saml/jbid_test_keystore.jks
```

Comment #1

In this IDP configuration, the **gatein.sso.enabled** parameter has been disabled.

This is because the portal IDP does not use any external SSO provider, but will now act as an SSO provider itself (SAML2 Identity Provider) for the portal SP.

Comment #2

For IDP we need to use **org.gatein.sso.saml.plugin.valve.PortalIDPWebBrowserSSOValve** valve, which is able to handle SAML request/response messages from SP applications and react on them.

Comment #3

The location of configuration file is relative to the portal WAR. The absolute path is **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-idp.xml**.

For this simple scenario, there is no need to edit this file. However, in a production environment, you will likely need to generate and use your own keystore and thus you must then configure the use of those keys in this file.

You will also need to manually add validating aliases in the keystore section if you have multiple SP applications on different hosts.

Comment #4

The URL of the portal, which will act as the SAML2 IDP.

Comment #5

This will enable a special session listener to clean up records about SAML tickets of expired hosts.

Comment #6

Comma-separated list of all SP domains to be trusted by this IDP.

Comment #7

Host for the portal SP. To add more SP applications, manually edit the **picketlink-idp.xml** file and add a **ValidatingAlias** element for each application.

Comment #8

The keystore in **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml/jbid_test_keystore.jks**.

2. Navigate to **JPP_SP_DIST/bin**, and start the SP instance by running:

```
./standalone.sh -b www.sp.com
```

3. Navigate to **JPP_IDP_DIST/bin**, and start the IDP instance by running:

```
./standalone.sh -b www.idp.com
```

See Also:

- Section 18.4, “Implementing Keystores”

Report a bug

18.2.4. Testing the Configuration

Procedure 18.3.

1. Navigate to http://www.sp.com:8080/portal/classic and click to Sign in.
2. From here, SAML request will be sent to www.idp.com and you will be redirected to login screen of the portal IDP instance on http://www.idp.com:8080/portal/dologin.
3. After a successful login, the SAML response will be sent back to www.sp.com and you will be redirected to the portal on www.sp.com as a logged-in user.
4. Now you can go to www.idp.com:8080/portal and you will see that you are logged-in as the SSO service has already logged you into this host.
5. Return to www.sp.com:8080/portal and click Sign Out. SAML2 single log out will trigger, and you will be logged out from both portals on the IDP and SP servers.
6. You can check that you are logged out from both www.sp.com:8080/portal and www.idp.com:8080/portal

Report a bug

18.3. Disabling SAML2 Single Logout

SAML2 Single logout is very powerful feature of SAML2 protocol, because triggering logout from any SP application will enforce "global" logout also from IDP and all other SP applications. It makes sense that you may not want this feature to be enabled, so logging out from JBoss Product Platform (click to Sign out link) will only logout the user from the JBoss Product Platform on www.sp.com, but user will still be logged in JPP IDP on www.idp.com and in all other SP applications.

Disabling this feature is as easy as switching an option in file GATEIN_SP_HOME/standalone/configuration/gatein/configuration.properties :

```
gatein.sso.filter.logout.enabled=false
```

Now when you click Sign out in www.sp.com, you will still be logged in JBoss Product Platform on www.idp.com . There won't be any SAML communication between SP and IDP during logout from www.sp.com.

Report a bug

18.4. Implementing Keystores

For secure and trusted communication, you will need your own keystores with your own keys. The default keystore is useful only for testing purposes, and must not be used in production. Separate keys for the portal SP, and for the IDP are created in this scenario. Follow the instructions to create a keystore for secured and trusted communication between SAML2 components.

Procedure 18.4. Configuring a Keystore for Secure Communication Between SP and IDP

1. Go to **JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml/** directory and run the command:

   ```
   keytool -genkey -alias secure-key -keyalg RSA -keystore secure-keystore.jks
   ```

   You need to choose keystore password and private key password. Other values do not matter. This guide assumes that your keystore password is *keystorepass* and a private key password is *keypass*.

2. For simplification purposes, this guide will use the same keystore for both the SP and IDP servers.

   Copy the keystore file generated in the last step to the IDP server directory; **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml/**.

3. Configure the new keystore in file **JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-sp.xml** and replace existing ***KeyProvider*** definition with:

   ```
   <KeyProvider
   ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
     <Auth Key="KeyStoreURL" Value="/sso/saml/secure-keystore.jks"/>
     <Auth Key="KeyStorePass" Value="keystorepass"/>
     <Auth Key="SigningKeyPass" Value="keypass"/>
   ```

```
  <Auth Key="SigningKeyAlias" Value="secure-key"/>
  <ValidatingAlias Key="${gatein.sso.idp.host}" Value="secure-
key"/>
</KeyProvider>
```

4. Configure the keystore in **`JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-idp.xml`** similarly:

```
<KeyProvider
ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKey
Manager">
  <Auth Key="KeyStoreURL" Value="/sso/saml/secure-keystore.jks"/>
  <Auth Key="KeyStorePass" Value="keystorepass"/>
  <Auth Key="SigningKeyPass" Value="keypass"/>
  <Auth Key="SigningKeyAlias" Value="secure-key"/>
  <ValidatingAlias Key="${gatein.sso.sp.host}" Value="secure-key"/>
</KeyProvider>
```

5. Restart both the SP and IDP servers.

   They will now use your new keystore instead of the default **`jbid_test_keystore.jks`**.

   While an argument could be made to use certificates signed by certification authority, self-signed certificates are fine for this purpose.

   For more information see http://docs.oracle.com/javase/tutorial/security/sigcert/index.html.

Report a bug

18.5. Setting up PicketLink IDP using REST callback

This basic scenario describes how to configure the necessary **`configuration.properties`** file. In the following procedure, the portal is configured as the SP and the *PicketLink IDP application* as the IDP. Additionally, the PicketLink IDP application will be configured to use REST callback to the portal instance to authenticate users (it will use the portal as an identity store).

Prerequisites

- One instance of the portal deployed to a directory called ***`JPP_SP_DIST`***.
- Another instance of the portal deployed to JPP_IDP_DIST. This server will run on www.idp.com and will host the PicketLink IDP application. Alternatively, if you have access to EAP 6, you can use it instead of the portal, however you will need to copy the PicketLink libraries required by the Picketlink IDP application from the portal SP instance. The libraries are located in **`JPP_SP_DIST/modules/org/picketlink/gatein`**

Important

The scenario uses a bundled test keystore, which is not suitable for production environments. For production environments, follow the instructions in Section 18.4, “Implementing Keystores” to provision a production-ready keystore.

Procedure 18.5. Configuring PicketLink IDP using REST callback

1. **Configure the portal as a SP compatible with PicketLink IDP**

   Open ***JPP_SP_DIST*/standalone/configuration/gatein/configuration.properties** and add the following configuration parameters.

```
# SSO
gatein.sso.enabled=true
gatein.sso.callback.enabled=${gatein.sso.enabled}
gatein.sso.login.module.enabled=${gatein.sso.enabled}
gatein.sso.login.module.class=org.gatein.sso.agent.login.SAML2Integ
rationLoginModule
gatein.sso.filter.login.sso.url=/@@portal.container.name@@/dologin
gatein.sso.filter.logout.class=org.gatein.sso.agent.filter.SAML2Logo
utFilter
gatein.sso.filter.initiatelogin.enabled=false
gatein.sso.valve.enabled=true
gatein.sso.valve.class=org.picketlink.identity.federation.bindings.
tomcat.sp.ServiceProviderAuthenticator
gatein.sso.saml.config.file=/WEB-INF/conf/sso/saml/picketlink-sp.xml
gatein.sso.idp.host=www.idp.com
gatein.sso.idp.url=http://${gatein.sso.idp.host}:8080/idp-sig/
gatein.sso.sp.url=http://www.sp.com:8080/portal/dologin
# WARNING: This bundled keystore is only for testing purposes.
Generate and use your own keystore in production!
gatein.sso.picketlink.keystore=/sso/saml/jbid_test_keystore.jks
```

2. Navigate to **JPP_SP_DIST/bin**, and start the SP instance by running:

```
./standalone.sh -b www.sp.com
```

3. **Configure Picketlink Identity Provider**

   a. Copy **JPP_DIST/gatein-sso/saml/idp-sig.war** to **IDP_DIST/standalone/deployments/**.

   b. Create **IDP_DIST/standalone/deployments/idp-sig.war.dodeploy** to force the application server to deploy the **idp-sig.war**. This file is required because the **idp-sig.war** is in an exploded format, not an archive format.

   c. Create a new security domain in **IDP_DIST/standalone/configuration/standalone.xml** that contains the following configuration.

```
<security-domain name="idp" cache-type="default">
   <authentication>
      <login-module
code="org.gatein.sso.saml.plugin.SAML2IdpLoginModule"
flag="required">
         <module-option name="rolesProcessing"
value="STATIC"/>
         <module-option name="staticRolesList"
value="manager,employee,sales"/>
         <module-option name="gateInURL"
value="http://www.sp.com:8080/portal"/>
```

```
          <module-option name="httpMethod" value="POST"/>
        </login-module>
    </authentication>
</security-domain>
```

4. **Start the IDP instance**

    a. Navigate to **IDP_DIST/bin** and execute the following command:

    ```
    ./standalone.sh -b www.idp.com -Dsp.host=www.sp.com -
    Dsp.domains=sp.com -
    Dpicketlink.keystore=/jbid_test_keystore.jks
    ```

    b. Navigate to http://www.sp.com:8080/portal and click Sign in.

    You will be redirected to the idp-sig application on http://www.idp.com:8080/idp-sig/.

    You are able to log in with portal credentials because the REST callback will be sent to the portal instance on www.sp.com and it will manage user authentication.

    After authentication you will be redirected to the portal on www.sp.com and logged-in.

Note

The **idp-sig.war** is a sample application that can be deployed on a different application server. To deploy it on Red Hat JBoss Portal Platform, which is running on an instance of EAP, change the sample application to add the following to **idp-sig.war/META-INF/jboss-deployment-structure.xml**:

```
<jboss-deployment-structure>
 <deployment>
   <!-- Add picketlink module dependency -->
   <dependencies>
     <module name="org.picketlink.idm"/>
     <module name="org.picketlink.federation"/>
     <module name="org.picketlink.federation.bindings"/>
     <module name="org.apache.httpcomponents" />
   </dependencies>
 </deployment>
</jboss-deployment-structure>
```

Note

In production environment, use your own keystore while deploying the **idp-sig.war** sample application on Red Hat JBoss Portal Platform, which is running on an instance of EAP, instead of the prebundled **jbid_test_keystore.jks**, which is in **idp-sig** application in **WEB-INF/classes/**.

Report a bug

18.6. Additional Information for SAML2

- **Videos** - You can follow some videos on the vimeo channel, where you can see SAML2 integration in action. The videos are:
  - http://vimeo.com/45841256 - Video with basic use cases showing the portal as a SAML2 SP and as a SAML2 IDP.
  - http://vimeo.com/45895919 - Video showing integration with Salesforce and Google Apps
- **SAML tracer** is a useful plug-in to Firefox browser, which allows you to monitor HTTP requests and see wrapped SAML messages in XML format. It could be useful especially for troubleshooting. You can download plug-in here
- **Logging** - Like for other SSO solutions, it may be useful to enable trace logging, for example for **`org.gatein.sso`**. In case of SAML, it is also good to enable logging for **`org.picketlink.identity.federation`**, which is the base package of the PicketLink Federation library. You can add these categories to **`GATEIN_HOME/standalone/configuration/standalone.xml`** :

Example 18.1. Trace Logging Example

```
<logger category="org.gatein.sso">
  <level name="TRACE"/>
</logger>
<logger category="org.picketlink.federation">
  <level name="TRACE"/>
</logger>
<logger category="org.jboss.security">
  <level name="TRACE"/>
</logger>
<logger category="org.picketbox">
  <level name="TRACE"/>
</logger>
<logger category="org.exoplatform.services.security">
  <level name="TRACE"/>
</logger>
```

Report a bug

Chapter 19. Using JBoss Portal SSO with Salesforce and Google Apps

The JBoss Portal SSO component contains support for Salesforce (http://www.salesforce.com) and Google Apps (http://www.google.com/enterprise/apps/business/ integration for SAML2 based SSO.

Three scenarios are described in this chapter, with links to sections detailing the configuration changes required to implement the scenarios.

Scenario One

Using JBoss Portal as the SAML Identity Provider (IDP) and Salesforce as the SAML Service Provider (SP).

Scenario Two

Using JBoss Portal as the IDP and Google Apps as the SP.

Scenario Three

Using Salesforce as the IDP and JBoss Portal as the SP.

A video showing the integration between Salesforce and Google Apps is available at http://vimeo.com/45895919. This is based on a community release of the GateIn Portal, so the configuration described in this video may vary slightly to enterprise configuration described in this chapter.

Report a bug

19.1. JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

19.1.1. IDP (JBoss Portal) and SP (Salesforce) Prerequisites

This configuration uses JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP). Ensure all prerequisites are completed before proceeding further with subsequent configuration.

To use Salesforce as a Service Provider (SP), specific configuration is required to Security Assertion Markup Language (SAML) Single Sign-on (SSO) within a Salesforce domain. An excellent overview that discusses how Federated Authentication operates is available at http://wiki.developerforce.com/page/Single_Sign-On_with_SAML_on_Force.com. Consider reading this to better understand information in the Salesforce configuration steps described in related tasks.

Prerequisites

- Understand file path abbreviations described in Section 12.1, “File Name Conventions”.
- Configure the Keystore using the suggested password in Section 18.4, “Implementing Keystores”.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.2, “Obtain a Salesforce developerforce Account”

Report a bug

19.1.2. Obtain a Salesforce developerforce Account

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.1, “IDP (JBoss Portal) and SP (Salesforce) Prerequisites”

Task: Create a new Salesforce developerforce Account

Create a developerforce account, which is required to provision a Salesforce domain for use as a SP or IDP.

1. Visit http://developer.force.com/.
2. Click **`Join Now`** to open the Sign up form.
3. Complete the fields, and choose a user name to access the Developer Edition Account with.
4. Click **`Sign me up`** to submit the registration details.
5. Complete registration by clicking the confirmation link in the email sent to your specified email account.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.3, “Creating a Salesforce Domain”

Report a bug

19.1.3. Creating a Salesforce Domain

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.2, “Obtain a Salesforce developerforce Account”

Task: Create a Salesforce Domain

Create a Salesforce domain which supports SP-initiated SAML login workflow. This configuration will allow a user to access the Salesforce domain, and have Salesforce forward a SAML request to JBoss Portal for authentication.

1. Log onto **`http://developer.force.com`**.
2. Click the user name in the top right corner of the page.
3. Click **Setup**+**Company Profile**+**My Domain**
4. Specify the name of the Salesforce domain, following the suggested format in the My Domain screen.
5. Save the details to complete domain registration.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.4, “Configure SAML SSO SP Settings”

Report a bug

19.1.4. Configure SAML SSO SP Settings

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.3, “Creating a Salesforce Domain”

Task: Configure Salesforce SAML SSO Settings

Configure the required values for SAML SSO in the Single Sign-on Settings page.

1. Log onto **http://developer.force.com**.
2. Click the user name in the top right corner of the page.
3. Click **Setup**+**Security Controls**+**Single Sign-On Settings**
4. Configure the following fields and settings:

   SAML Enabled

   Ensure the check box is marked.

   SAML Version

   Specify 2.0 as the value.

   Issuer

   Specify the issuer as **http://www.idp.com:8080/portal/dologin**, which is used as the Identity Provider for the Salesforce domain.

   Identity Provider Login URL

   Specify the Identity Provider as **http://www.idp.com:8080/portal/dologin**, which is the URL Salesforce SAML SSO sends the SAML Requests for authentication.

   Identity Provider Logout URL

   Set this value to **http://www.idp.com:8080/portal/dologin**. If you have a custom page that users are directed to when they log out, this field can contain the URL of the page.

   SAML User ID Type

   Set this value to **User ID contains the Federation ID from the User object**.

   SAML User ID Location

   Set this value to **User ID is the NameIdentifier element of the Subject statement**.

   Entity ID

   Set this value to **https://saml.salesforce.com**.

   Service Provider Initiated Request Binding

Set this value to **HTTP POST**.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.5, “Import Message Signing Certificate into Salesforce”

Report a bug

19.1.5. Import Message Signing Certificate into Salesforce

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.4, “Configure SAML SSO SP Settings”

Task: Import the JBoss Portal Message Signing Certificate into Salesforce

Import the message signing certificate from the JBoss Portal IDP server to the Salesforce server, which enables Salesforce to verify SAML responses sent from the JBoss Portal IDP.

1. Use the secure-keystore.jks SAML message signing keystore in the **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/sso/saml** directory to create a certificate.

    a. In the **/saml** directory, run **keytool -export -file portal-idp.crt -keystore secure-keystore.jks -alias secure-key**.

    b. When prompted, provide the keystore password **keystorepass**.

    This password is the default suggested password created as part of the Section 18.4, “Implementing Keystores” process. If you chose a different password, ensure this is provided at the prompt.

    The command will generate a certificate named **portal-idp.crt**.

2. Import the **portal-idp.crt** file into Salesforce.

    On http://developer.force.com, click the user menu, and then click **Setup → Security Controls → Single Sign-On Settings → Identity Provider Certificate**.

    Import the **portal-idp.crt** certificate into Salesforce, and set the alias to **secure-key**.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.6, “Create Salesforce and Portal Users”

Report a bug

19.1.6. Create Salesforce and Portal Users

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.5, “Import Message Signing Certificate into Salesforce”

Create identical Salesforce and Portal Users

In order to test the SAML SSO configuration, users must be provisioned in both JBoss Portal and Salesforce that meet requirements for Federated ID matching.

1. Create the following users in JBoss Portal, following the instructions in the *Register New Accounts* chapter in the *User Guide*.
   - Mary Citizen

     User name of **Mary**.
   - John Citizen

     User name of **John**.
2. Create the following users in Salesforce, following the instructions in the Salesforce Wiki Documentation located at http://login.salesforce.com/help/doc/en/adding_new_users.htm.

   Important

   The Federation ID of users in Salesforce must be identical to the user name in JBoss Portal.

   - Mary Citizen

     Federation ID of **Mary**.

     User name of `mary@example.com`.
   - John Citizen

     Federation ID of **John**.

     User name of `john@example.com`.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.7, “Obtain the Salesforce Client Certificate”

Report a bug

19.1.7. Obtain the Salesforce Client Certificate

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.6, “Create Salesforce and Portal Users”

Task: Download the Salesforce Client Certificate

Download the Salesforce Client Certificate. **`proxy-salesforce-com.123`** is required to sign SAML assertions, and the public key is required to verify the SAML assertions sent from Salesforce are genuine on the client side.

1. Review the information on http://wiki.developerforce.com/page/Client_Certificate.
2. Download the certificate from the link in the *What do I do next?* section of the wiki page.

3. Extract the certificate chain archive to a working directory to gain access to the **proxy-salesforce-com.123** file.

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.8, “Configure JBoss Portal as the IDP”

Report a bug

19.1.8. Configure JBoss Portal as the IDP

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.7, “Obtain the Salesforce Client Certificate”

Task: Configuring JBoss Portal as the Identity Provider for Salesforce

Set up JBoss Portal to act as the IDP for Salesforce.

1. Login to http://developer.force.com.
2. From the user menu, click **Setup → Security Controls → Single Sign-on Settings → Download Metadata** to download the Service Provider metadata from Salesforce.
3. Download the Salesforce metadata file to **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/sp-metadata.xml**.
4. In the directory where the **proxy-salesforce.com.123** certificate is saved, run the **keytool -import -keystore secure-keystore.jks -file proxy-salesforce-com.123 -alias salesforce-cert** command to import the Salesforce client certificate into the portal keystore.
5. Open **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-idp.xml** and add the following ValidatingAlias directive.

```
<ValidatingAlias Key="saml.salesforce.com" Value="salesforce-cert"
/>
```

6. In the **picketlink-idp.xml** file, declare the Metadata Provider directive immediately after the </KeyProvider> directive.

```
<!-- Preceeding content removed for readability-->
         </KeyProvider>

         <MetaDataProvider
ClassName="org.picketlink.identity.federation.core.saml.md.providers.FileBasedEntitiesMetadataProvider">
           <Option Key="FileName" Value="/WEB-INF/sp-metadata.xml"/>
         </MetaDataProvider>
```

7. Open **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/sp-metadata.xml** and add the single logout service and entities descriptor blocks.

```
<md:SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
<!-- Comment #1 -->
Location="[Salesforce.com Single Logout URL]"
index="0" isDefault="true"/>
<md:EntitiesDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
<md:EntityDescriptor
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="https://saml.salesforce.com"
  <!-- Comment #2 -->
  </md:EntityDescriptor>
</md:EntitiesDescriptor>
```

Comment #1

The Location URL is unique to every Salesforce domain. Obtain the URL specific to the configured domain from the **`Salesforce.com Single Logout URL`** field on the Salesforce Federated ID page.

Comment #2

Note that other entity descriptions may be present in this block. Examples of entity descriptions that would belong in this block include the Google Apps entity description, which is configured in another scenario.

8. Declare the trusted domains **`sp.com`** and **`idp.com`** in **`JPP_IDP_HOME/standalone/configuration/gatein/configuration.properties`**

```
gatein.sso.sp.domains=sp.com,idp.com,salesforce.com
```

Next Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.9, “Test the IDP (JBoss Portal) and SP (Salesforce) Configuration”

Report a bug

19.1.9. Test the IDP (JBoss Portal) and SP (Salesforce) Configuration

Previous Step in JBoss Portal as the Identity Provider (IDP) and Salesforce as the Service Provider (SP)

- Section 19.1.8, “Configure JBoss Portal as the IDP”

Task: Testing the IDP and SP Configuration

Test the configuration using JBoss Portal as the SAML Identity Provider and Salesforce as the SAML Service Provider.

Note

In the Salesforce SSO menu, a basic tool is available that tracks the last SAML response sent to the Salesforce Service Provider. This tool can prove useful when analyzing the response and attempting to determine the root cause of an error.

1. Visit the Salesforce domain. For example, https://yourdomain.my.salesforce.com.
2. Once redirected to the JBoss Portal login screen (for example, http://www.idp.com:8080/portal/dologin), enter the user credentials for a user with a Salesforce Federated ID.
3. Once redirected back to Salesforce, verify that the user is authenticated as their Salesforce Federated ID.

Note

SAML2 Single logout initiated by Salesforce is not supported. When you click "Sign out" in Salesforce, Salesforce will not send any SAML2 LogoutRequest to IDP, so you will be still logged on JBoss Portal IDP. Salesforce only supports handling SAML2 LogoutRequests sent from IDP. If you have deployment with one instance of JBoss Portal as SP, Salesforce as SP, and one instance of JBoss Portal as IDP, and trigger a log out request from JBoss Portal, the SP will log you out from all 3 applications (in this case Salesforce is not initiator of Single logout but it handles LogoutRequest sent from IDP).

Report a bug

19.2. JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

19.2.1. IDP (JBoss Portal) and SP (Google Apps) Prerequisites

This scenario uses JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP).

Prerequisites

- Understand file path abbreviations described in Section 12.1, “File Name Conventions”.
- Be familiar with the content available at http://www.google.com/enterprise/apps/business/.
- Configure JBoss Portal to act as the SAML IDP as described in Section 19.1.8, “Configure JBoss Portal as the IDP”.

Next Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.2, “Create A Google Apps for Business Account”

Report a bug

19.2.2. Create A Google Apps for Business Account

Previous Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.1, “IDP (JBoss Portal) and SP (Google Apps) Prerequisites”

Creating a Google Apps for Business Account

Create a Google Apps for Business account for the purposes of testing SSO. The created account is initially provided as a full-featured trial for 30 days, after which time it can be converted to a paid account.

1. Visit http://www.google.com/enterprise/apps/business/.
2. Click **Get Started**.
3. Complete the fields on the **About you** form as suggested by the field help.
4. On the **Your Business Domain Address** screen, make a selection based on your requirements.

   Use your own domain

   Choose this option if you intend to configure Google Apps for Business for production use.

   mygbiz.com Domain (recommended for this example process)

   Choose this option if you are experimenting with SSO, and do not want to affect your primary business domain. You can upgrade the account to use a domain you own any time after testing out the functionality.
5. If you choose the recommended free domain option, provide a unique name for the domain in the field, and click **Next**.
6. On the Create your Google apps account form, provide the information requested.
7. Click Accept and signup.
8. Verify you can access your account by authenticating using the credentials you created as part of this procedure.

   If successful, the Google Admin Console will display.

Next Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.4, “Configuring Google Apps as the SP”

Report a bug

19.2.3. Create Default Google Apps for Business Users

Create Identical Google Apps for Business and Portal Users

In order to test the SAML SSO configuration, users must be provisioned in both JBoss Portal and Google Apps for Business that meet requirements for Federated ID matching.

1. Create the following users in JBoss Portal, following the instructions in the *Register New Accounts* chapter in the *User Guide*.
   - Mary Citizen

     User name: **mary**
   - John Citizen

     User name: **john**
2. From the Google Admin Console, click **Users**.
3. Click **Add more users**, and select **Add a user manually**.
4. Create the following users in Google Apps for Business.

Important

The user names set for users in Google Apps must be identical to those in JBoss Portal because the portal user name is connected with the Google Apps user name.

   - Mary Citizen

     Primary email address: mary@[domain].mygbiz.com .

     This creates a user name of **mary** and an email address of mary@[domain].mygbiz.com.
   - John Citizen

     Primary email address: john@[domain].mygbiz.com.

     This creates a user name of **john** and an email address of john@[domain].mygbiz.com.

Report a bug

19.2.4. Configuring Google Apps as the SP

Previous Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.2, “Create A Google Apps for Business Account”

Task: Configuring Google Apps for Business as the Service Provider.

Configure Google Apps for Business to act as the Service Provider by changing the security settings of the test domain.

1. Open https://admin.google.com and sign-in if necessary.
2. If the Security icon is not visible, click **More Controls** and drag the Security icon onto the Admin console main screen.
3. Click **Security**+**Advanced Settings** → **Set up single sign-on (SSO)**.

4. In the Set up single sign-on (SSO) form, configure the following fields and settings:

   Enable Single Sign-on

   Select the tick box.

   Sign-in page URL

   Specify the sign-in page as **http://www.idp.com:8080/portal/dologin**.

   Sign-out page URL

   Specify the sign-out page as **http://www.idp.com:8080/portal/dologin**.

   Change password URL

   Specify the change password page as **http://www.idp.com:8080/portal/dologin**.

   Verification certificate

   Export certificates from the JBoss Portal keystore into a file (for example, **portal-idp.crt**), and upload the certificate into the form.

   Exporting certificates from the JBoss Portal keystore is described in Section 19.1.5, “Import Message Signing Certificate into Salesforce”.

   Use a domain specific issuer

   Select the tick box.

5. Verify all settings are correct, and click **Save changes**.

Next Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.5, “Configuring JBoss Portal as the IDP”

Report a bug

19.2.5. Configuring JBoss Portal as the IDP

Previous Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.4, “Configuring Google Apps as the SP”

Task: Configuring JBoss Portal as the Identity Provider for Google Apps

Set up JBoss Portal to act as the IDP for Google Apps for Business.

1. In the **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-idp.xml** file, declare the Metadata Provider directive immediately after the </KeyProvider> directive.

```
<!-- Preceeding content removed for readability-->
        </KeyProvider>

        <MetaDataProvider
```

```
ClassName="org.picketlink.identity.federation.core.saml.md.providers.FileBasedEntitiesMetadataProvider">
            <Option Key="FileName" Value="/WEB-INF/sp-metadata.xml"/>
        </MetaDataProvider>
```

2. Open **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/sp-metadata.xml** and specify that SAML requests from the Google Apps for Business SP will not be signed.

```
<md:EntitiesDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  <!-- Comment#1 -->
   <md:EntityDescriptor
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="google.com/a/[domain.mygbiz.com]" validUntil="2022-06-13T21:46:02.496Z">
  <!-- Comment#2 -->
      <md:SPSSODescriptor AuthnRequestsSigned="false"
WantAssertionsSigned="true"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
/>
   </md:EntityDescriptor>
</md:EntitiesDescriptor>
```

Comment #1

The entityID="google.com/a/[domain.mygbiz.com]" [domain.mygbiz.com] value must be replaced with the domain name specified in Section 19.2.2, “Create A Google Apps for Business Account”

Comment #2

Adding the **AuthnRequestsSigned** attribute as described prevents SAMLRequests from being validated. This is necessary because Google Apps for Business does not add signatures to its SAML Requests.

3. Declare the trusted domains **sp.com**, **idp.com** and **google.com** in **JPP_IDP_HOME/standalone/configuration/gatein/configuration.properties**. If other domains are declared for this parameter, append the trusted domains to the line.

```
gatein.sso.sp.domains=sp.com,idp.com,google.com
```

4. In the **JPP_IDP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-idp.xml** file, declare a ValidatingDomain directive.

```
<ValidatingAlias Key="127.0.0.1" Value="secure-key"/>
```

Note

Even though Google SAMLRequests are not signed, PicketLink requires that each SAMLRequest must have a key. When a key is not found for a specific domain (in this case google.com), PicketLink will search for keys with the alias **`127.0.0.1`**

Next Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.6, “Testing the IDP (JBoss Portal) and SP (Google Apps) Configuration”

Report a bug

19.2.6. Testing the IDP (JBoss Portal) and SP (Google Apps) Configuration

Prerequisites:

- Section 19.2.4, “Configuring Google Apps as the SP”
- Section 19.2.5, “Configuring JBoss Portal as the IDP”

Previous Step in JBoss Portal as the Identity Provider (IDP) and Google Apps as the Service Provider (SP)

- Section 19.2.5, “Configuring JBoss Portal as the IDP”

After configuring both the Identity Provider and the Service Provider, test the Google Apps configuration.

Procedure 19.1.

1. Visit the Google Apps domain configured in Section 19.2.2, “Create A Google Apps for Business Account”.

   The Google Apps domain URL takes the structure https://www.google.com/a/mydomain.mygbiz.com/ServiceLogin. Replace *mydomain* with the custom value of your domain.

2. Once redirected to the JBoss Portal login screen (for example, http://www.idp.com:8080/portal/dologin), enter the JBoss Portal user credentials for a user created in Section 19.2.3, “Create Default Google Apps for Business Users”.

3. Once redirected back to the Google Apps domain, verify that the JBoss Portal user is authenticated correctly.

Report a bug

19.3. Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

19.3.1. IDP (Salesforce) and SP (JBoss Portal) Prerequisites

This configuration uses Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP). Ensure all prerequisites are completed before proceeding further with subsequent configuration.

If you have experimented with the Section 19.1.1, “IDP (JBoss Portal) and SP (Salesforce) Prerequisites” process, some configuration can be reused, and is described in each task. The process steps assume you have not configured anything regarding Salesforce, and guide you through from start to finish.

Prerequisites

- Understand file path abbreviations described in Section 12.1, “File Name Conventions”.
- Configure JBoss Portal to act as the SAML SP as described in Section 18.2.2, “Configuring a SAML2 Service Provider”.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.2, “Obtain a Salesforce developerforce Account”

Report a bug

19.3.2. Obtain a Salesforce developerforce Account

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.1, “IDP (Salesforce) and SP (JBoss Portal) Prerequisites”

Task: Create a new Salesforce developerforce Account

Create a developerforce account, which is required to provision a Salesforce domain for use as a SP or IDP.

1. Visit http://developer.force.com/.
2. Click **`Join Now`** to open the Sign up form.
3. Complete the fields, and choose a user name to access the Developer Edition Account with.
4. Click **`Sign me up`** to submit the registration details.
5. Complete registration by clicking the confirmation link in the email sent to your specified email account.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.3, “Creating a Salesforce Domain”

Report a bug

19.3.3. Creating a Salesforce Domain

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.2, “Obtain a Salesforce developerforce Account”

Task: Create a Salesforce Domain

Create a Salesforce domain which supports SP-initiated SAML login workflow. This configuration will allow a user to access the Salesforce domain, and have Salesforce forward a SAML request to JBoss Portal for authentication.

1. Log onto **http://developer.force.com**.
2. Click the user name in the top right corner of the page.
3. Click **Setup**+**Company Profile**+**My Domain**
4. Specify the name of the Salesforce domain, following the suggested format in the My Domain screen.
5. Save the details to complete domain registration.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.4, “Disable SP Single Sign-on in Salesforce”

Report a bug

19.3.4. Disable SP Single Sign-on in Salesforce

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.3, “Creating a Salesforce Domain”

Task: Disabling Salesforce SAML SSO Settings

If you have previously configured Salesforce to act as a SAML SSO Service Provider, this configuration must be disabled before Salesforce can act as an Identity Provider.

1. Log onto **http://developer.force.com**.
2. Click the user name in the top right corner of the page.
3. Click **Setup**+**Security Controls**+**Single Sign-On Settings**
4. Clear the SAML Enabled tick box to disable all SAML SSO configuration.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.5, “Create and Apply a Salesforce IDP Message Signing Certificate”

Report a bug

19.3.5. Create and Apply a Salesforce IDP Message Signing Certificate

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.4, “Disable SP Single Sign-on in Salesforce”

Task: Generate a Salesforce IDP Message Signing Certificate, and apply it to Salesforce

Identity Provider message signing certificates are specific to each Salesforce domain. Create a Salesforce IDP certificate and apply the certificate to the Salesforce server.

1. Log onto http://developer.force.com.
2. Click the user name in the top right corner of the page.
3. Click **Setup** → **Security Controls** → **Identity Provider** → **Enable Identity Provider**.

   Any certificates already generated by Salesforce are displayed.
4. Click **Create a new certificate...**, and specify the following values:

   Label

   salesforce-idp-cert

   Unique Name

   salesforce-idp-cert
5. Download the created certificate by clicking the **`Download Certificate`** button on the Identity Provider screen. Save the certificate to **`/tmp/salesforce_idp_cert.cer`**.
6. To apply the created certificate navigate back to the **Enable Identity Provider** screen and apply the certificate created in the previous step.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.6, “Create Salesforce and Portal Users”

Report a bug

19.3.6. Create Salesforce and Portal Users

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.5, “Create and Apply a Salesforce IDP Message Signing Certificate”

Create identical Salesforce and Portal Users

In order to test the SAML SSO configuration, users must be provisioned in both JBoss Portal and Salesforce that meet requirements for Federated ID matching.

1. Create the following users in JBoss Portal, following the instructions in the *Register New Accounts* chapter in the *User Guide*.
   - Mary Citizen

     User name of **`Mary`**.
   - John Citizen

     User name of **`John`**.

2. Create the following users in Salesforce, following the instructions in the Salesforce Wiki Documentation located at http://login.salesforce.com/help/doc/en/adding_new_users.htm.

Important

The Federation ID of users in Salesforce must be identical to the user name in JBoss Portal.

- Mary Citizen

  Federation ID of **Mary**.

  User name of mary@example.com.

- John Citizen

  Federation ID of **John**.

  User name of john@example.com.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.7, “Configuring Salesforce as the IDP”

Report a bug

19.3.7. Configuring Salesforce as the IDP

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.6, “Create Salesforce and Portal Users”

Task: Configuring Salesforce as the Identity Provider

Configure Salesforce as the Identity Provider, in preparation for JBoss Portal to act as the Service Provider.

1. On the Salesforce home page, click the user name and then click **Setup → Security Controls → Identity Provider → New**

2. Configure the following fields and settings:

   Name

   Specify portal-sp as the value.

   ACS URL

   Specify the Assertion Consumer Service URL as **http://www.sp.com:8080/portal/dologin**

   This value is the address of the Service Provider, in this case the JBoss Portal instance on the sp.com domain.

   Entity ID

Specify **http://www.sp.com:8080/portal/dologin** as the value.

Start URL

Leave this parameter blank.

Subject Type

Set this value to Federation ID.

Service Provider Certificate

Use the certificate exported in Section 18.4, “Implementing Keystores”, if you are using the same certificate for the Service Provider and the Identity Provider. For reference, Section 19.1.5, “Import Message Signing Certificate into Salesforce” describes the commands to export the message signing certificate.

If not, export the **portal-idp.crt** certificate from the keystore file in **JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/saml/sso/secure-keystore.jks**.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.8, “Configuring JBoss Portal as the SP”

Report a bug

19.3.8. Configuring JBoss Portal as the SP

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.7, “Configuring Salesforce as the IDP”

Configuring JBoss Portal as the Salesforce Service Provider

Make the required changes to the security policy and configuration to set JBoss Portal as the Salesforce Service Provider.

1. Import the certificate created by Salesforce into the JBoss Portal keystore located in **JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/saml/sso/secure-keystore.jks**.

   Use the command **keytool -import -file /tmp/salesforce_idp_cert.cer -keystore secure-keystore.jks -alias salesforce-idp** to import the certificate.

2. Open **JPP_SP_HOME/standalone/configuration/gatein/configuration.properties** and change the gatein.sso properties to values corresponding to the Salesforce domain, and the Portal Platform SP URL.

   ```
   gatein.sso.idp.url=https://[yourdomain].my.salesforce.com/idp/endpo
   int/HttpPost
   gatein.sso.sp.url=http://www.sp.com:8080/portal/dologin
   ```

3. Open **JPP_SP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/sso/saml/picketlink-sp.xml** and add the ValidatingAlias directive.

```
<ValidatingAlias Key="[yourdomain].my.salesforce.com"
Value="salesforce-idp" />
```

Note

Because the JBoss Portal Service Provider obtains role information from the Picketlink Identity Management database, specific roles-mapping configuration normally configured for delivery in the SAML response is not required.

Next Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.9, “Testing the IDP (Salesforce) and SP (JBoss Portal) Configuration”

Report a bug

19.3.9. Testing the IDP (Salesforce) and SP (JBoss Portal) Configuration

Prerequisites:

- Section 19.3.7, “Configuring Salesforce as the IDP”
- Section 19.3.8, “Configuring JBoss Portal as the SP”

Previous Step in Salesforce as the Identity Provider (IDP) and JBoss Portal as the Service Provider (SP)

- Section 19.3.8, “Configuring JBoss Portal as the SP”

Procedure 19.2. Testing the IDP (Salesforce) and SP (JBoss Portal) Configuration

1. Start JBoss Portal on the host (assuming the host set to *www.sp.com*).
2. Open http://www.sp.com:8080/portal, and click **`Sign in`**.

   JBoss Portal sends the SAML Request to Salesforce, and redirects to the Salesforce login screen.
3. Log onto Salesforce, using valid user name and password information.

   After providing correct credentials, you are redirected back to the portal home screen. You are authenticated with Salesforce login credentials because your portal user name is mapped to the Federated ID in Salesforce.

Report a bug

Chapter 20. OAuth - Authentication with Social Network accounts

OAuth is an authorization protocol, which allow third party applications to access user resources on behalf of another application (authorization server). The main purpose of OAuth is authorization, but in many web pages, OAuth is also used for authentication or registration of users. JBoss Portal uses OAuth for authentication and registration of users. Currently JBoss Portal acts as a third party application (OAuth client application) and it supports integration with OAuth providers like Facebook, Google+ and Twitter.

For details about specification, see the OAuth page. The current version of specification is OAuth 2.0 and the older version is OAuth 1.0a. Some vendors only support the older version.

Report a bug

20.1. Working of OAuth Protocol

You need to create and register new application on OAuth provider (Facebook, Google, Twitter). Registration is a vendor specific process. In this chapter, the registration process for each vendor is described in detail. The common result of registration for every vendor are:

- Client ID of your application
- Client Secret of your application
- Redirect URL, which will be used by OAuth provider to redirect back from authorization screen to JBoss Portal screen

Registration specific changes are required in the JBoss Portal configuration file **`configuration.properties`**.You need to enable particular OAuth provider and configure the values from registration. After startup of JBoss Portal, your users can login and register into JBoss Portal with OAuth provider.

Report a bug

20.2. OAuth Protocol User Interface

20.2.1. User Registration

Assumptions

- JBoss Portal is running on host **`server.local.network.com`**

We are using Facebook as an example to show the user registration process. This process is similar for other providers such as Google+ and Twitter.

Procedure 20.1. Registration of Portal with Facebook

1. JBoss Portal login page has options for logging with Facebook, Google or Twitter. It shows the providers, which you configured. See figure with enabled authentication for Facebook and Google+:

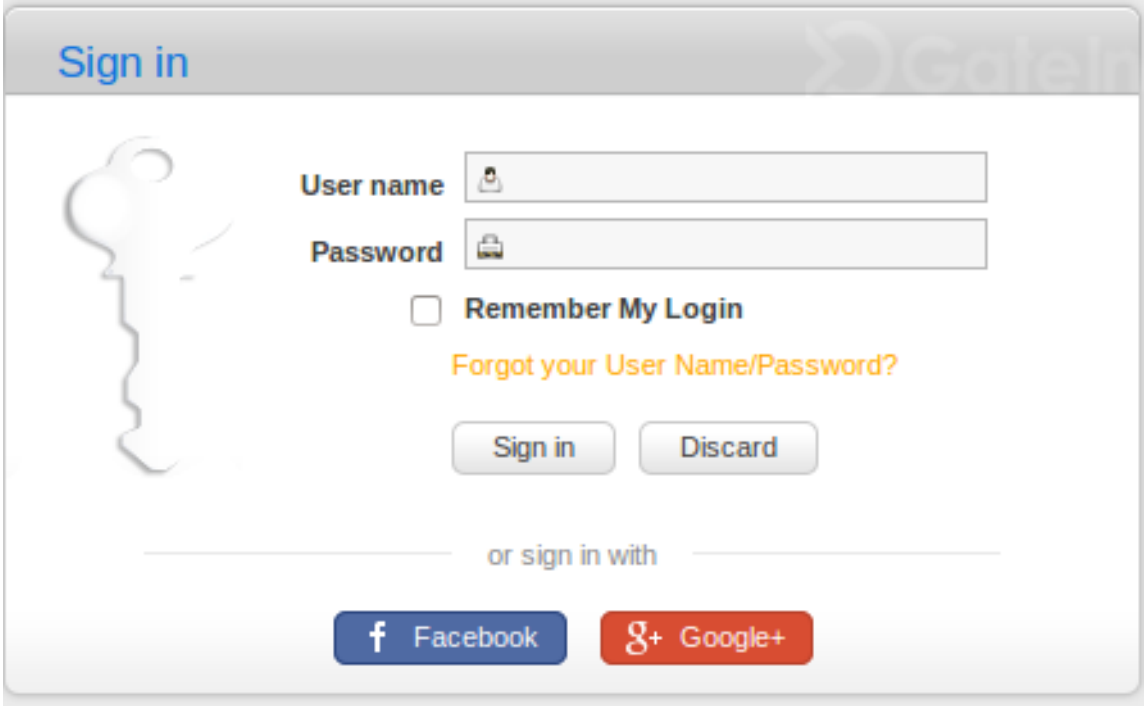


Figure 20.1. Sign in dialog box with Facebook and Google enabled

2. Click **`Sign in with Facebook`** to redirect the user to the Facebook site.
    a. The user has to authenticate Facebook if not already authenticated.
    b. Facebook displays an authorization screen.

Note

The required privileges (scopes) are configured in **`configuration.properties`**. The email and user profile information is sufficient. You can add more privileges but then you have less chance of your users confirming the authorization screen. (For example: It is not a good practice to request **`publish message on FB wall`** privilege, if you are using JBoss Portal only for registration/authentication of user)

3. User confirms the authorization screen and is redirected to JBoss Portal. The redirected URL is the one you configured as Redirect URL during registration process of your application. For example, the URL for Facebook is http://server.local.network.com:8080/portal/facebookAuth
4. JBoss Portal displays the registration dialog box with prepopulated values from Facebook.

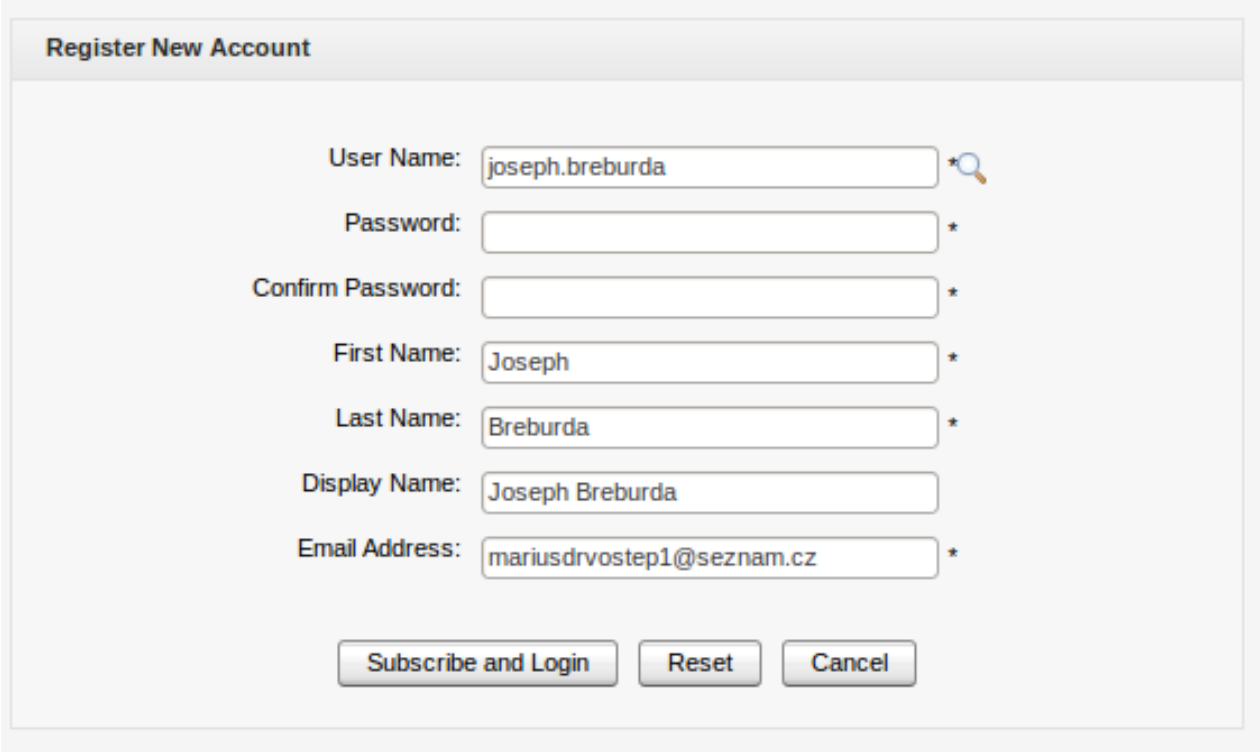


Figure 20.2. Registration dialog with prepopulated values from Facebook

5. The registration is complete and the user is signed into JBoss Portal.

Note

The user joseph.breburda is treated as normal user of JBoss Portal. So portal administrator can see him in list of users in JBoss Portal UI and can add him to more groups (By default the user is in **`/platform/users`** group). User registered through Facebook has two additional properties saved as part of his UserProfile:

User properties

Facebook username

It is used for mapping the JBoss Portal identity to Facebook identity. It can be same as JBoss Portal username, however it is not necessary. It provides information regarding mapping of user names.

For example, JBoss Portal user joseph is mapped to Facebook user joseph.breburda.

Note

JBoss Portal has a restriction that Facebook username needs to be unique, so we cannot have two JBoss Portal users with same Facebook username.

Facebook access token

This is needed for social portlets to use accessToken to retrieve required information from Facebook. For example, you can have portlet for displaying content of Facebook wall of

particular user.

For more details, see Developer guide, chapter *Using data from Social Networks in Portlets*

Important

Access token is saved in Picketlink IDM in encoded form using symmetric encryption.

Report a bug

20.2.2. Login Workflow

This workflow is used when JBoss Portal username is connected with Facebook username.

For example, JBoss Portal user **joseph** is connected with Facebook user **joseph.breburda**

Assumptions

- The user authenticated to Facebook as **joseph.breburda** is mapped to portal user **joseph**

Procedure 20.2. Logging process

1. The first three steps are same as Section 20.2.1, “User Registration”.
2. User is logged into JBoss Portal as joseph because JBoss Portal already knows that username **joseph** belongs to authenticated Facebook user **joseph.breburda**

Report a bug

20.3. Integrating OAuth with the Portal

Assumption

- The portal is running on the virtual host **server.local.network.com**.

Procedure 20.3. Integrating OAuth

1. Setup a virtual host by adding it to **/etc/hosts** (on linux):

```
192.168.1.88  server.local.network.com
```

2. In the **JPP_HOME/standalone/configuration/gatein/configuration.properties** file, change the property **gatein.oauth.portal.url** to the value of the host.
3. Set the value for **gatein.oauth.portal.url**.

```
gatein.oauth.portal.url=http://server.local.network.com:8080
```

Report a bug

20.4. Integration of OAuth with Facebook

Facebook is a popular social network, that supports OAuth 2.0 protocol and can act as OAuth provider (Resource server) for JBoss Portal.

Integration of OAuth with facebook consists of two main parts:

- Registration of JBoss Portal application on Facebook
- Configuring JBoss Portal for using OAuth Protocol with Facebook

Report a bug

20.4.1. Registration of Portal application on Facebook

Assumptions

- JBoss Portal host is **server.local.network.com** as described in Section 20.3, “Integrating OAuth with the Portal”.

Procedure 20.4. Registering Portal on Facebook

1. Register as Facebook developer on https://developers.facebook.com/apps.
2. Click **Create new application** and choose an application name such as var**gatein_app**.

Note

The application name is always unique.

3. Configure **App Domains** to contain value **local.network.com** and **Site URL for Facebook login** to contain value http://server.local.network.com:8080. You need to remember **Client ID** and **Client Secret**.

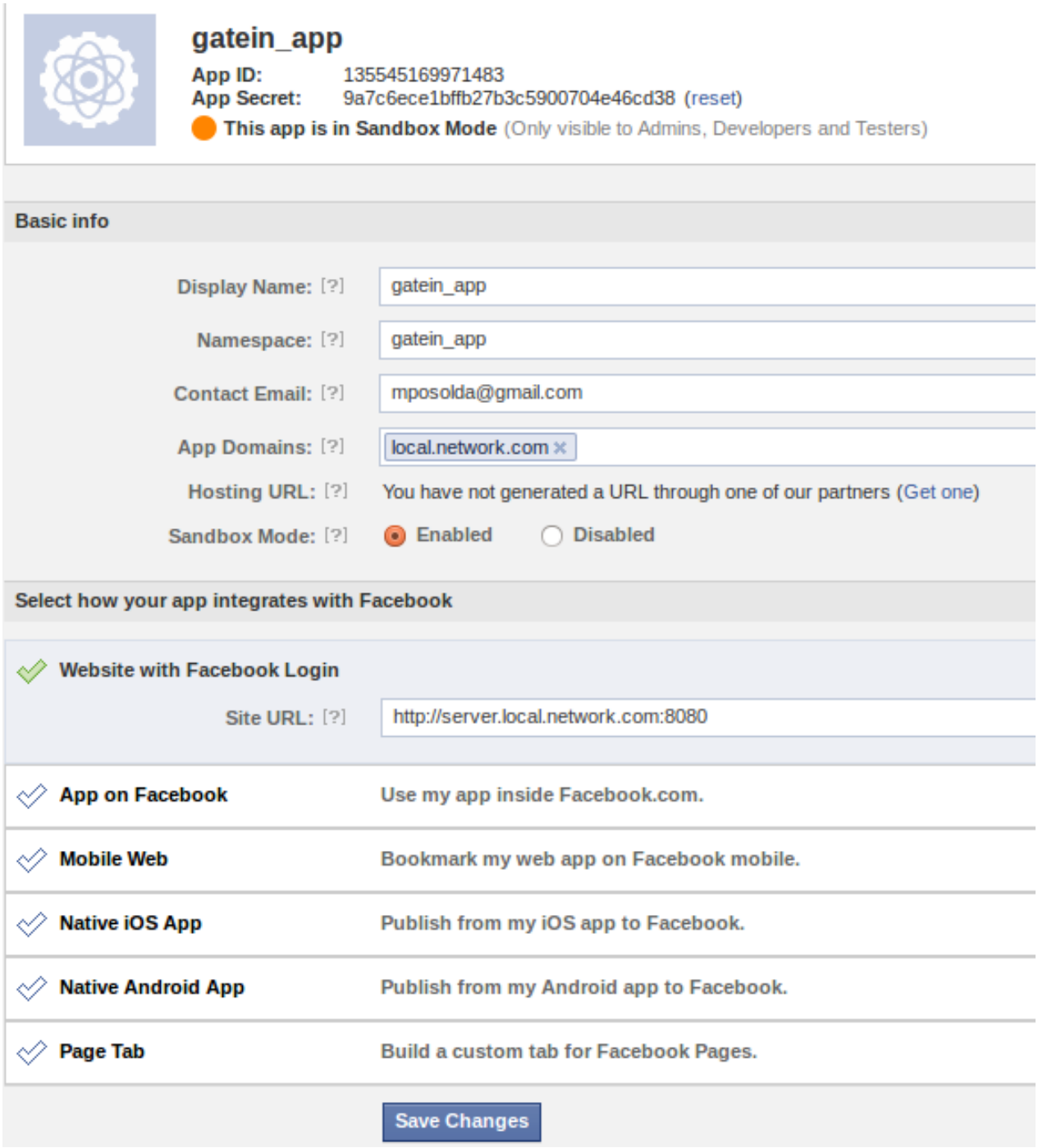


Figure 20.3. Registration of Portal application on Facebook

4. Save changes.

Important

Before initiation production, you have to disable sandbox mode in your Facebook application. With sandbox enabled, application is available only for you, your developers and, test users created for your application

Report a bug

20.4.2. Configuring JBoss Portal for using OAuth Protocol with Facebook

Procedure 20.5. Configure properties in Facebook section

- Change the file **configuration.properties** and configure properties in Facebook.

```
## Facebook
gatein.oauth.facebook.enabled=true
gatein.oauth.facebook.clientId=135545169971483
gatein.oauth.facebook.clientSecret=9a7c6ece1bffb27b3c5900704e46cd38
gatein.oauth.facebook.redirectURL=${gatein.oauth.portal.url}/@@portal.
container.name@@/facebookAuth
gatein.oauth.facebook.scope=email
```

Description of properties

gatein.oauth.facebook.enabled

It enable integration with Facebook. Users can login and register with Facebook.

gatein.oauth.facebook.clientId

Client ID of your application. This is obtained from registration on Facebook as described in Section 20.4.1, “Registration of Portal application on Facebook”

gatein.oauth.facebook.clientSecret

Client Secret of your application. This is obtained from registration on Facebook as described in Section 20.4.1, “Registration of Portal application on Facebook” .

Note

The client secret should not be shared among your portal users or other people.

gatein.oauth.facebook.redirectURL

The URL used by Facebook to redirect after user confirms Facebook authorization screen as shown in, step 3 of Section 20.2.1, “User Registration”. You do not need to change this value. But if you have changed property **gatein.oauth.portal.url** as described in Section 20.3, “Integrating OAuth with the Portal”, you have to set this value.

gatein.oauth.facebook.scope

Scopes represent permissions required from users on Facebook authorization screen. This list of scope is used during user registration or authentication to JBoss Portal (portlet applications may require more scopes if they need it). It is recommended not to change this value and keep value as email. Alternatively you can:

- Delete email and use empty value instead - In this case, JBoss Portal will ask only for basic set of permissions. However, JBoss Portal will not be able to access email address, so this field will remain empty during user's registration to JBoss Portal.
- Use more values - you can use more values described in Facebook documentation . However, it is likely that users will not allow your Portal application to access their informations. And having more values is only useful for quickstart portlet applications, but quickstart applications can ask for more scopes as per their requirement.

Note

Scopes from this configuration property are used during registration/login of users

After restart of JBoss Portal, your users should be able to register or login with their Facebook accounts as described in Section 20.2.1, “User Registration” .

Report a bug

20.5. Integration of OAuth with Google plus

Google+ is a popular social network. It supports OAuth 2.0 protocol. The integration of Google plus with JBoss Portal consists of two main parts:

- Registration of JBoss Portal application on Google plus
- Configuring JBoss Portal for using OAuth Protocol with Google plus

Report a bug

20.5.1. Registration of Portal application on Google

Assumption

- JBoss Portal is hosted on **`server.local.network.com`** as described in Section 20.3, “Integrating OAuth with the Portal” .

Procedure 20.6. Registering Portal on Google

1. Register as Google developer on https://code.google.com/apis/console/.
2. Click **`Create new application`** and choose an application name such as **`gatein_google_app`**.
3. Under **`services`**, turn on Google+ API.
4. Under **`>API Access`**, click **`Create new OAuth 2.0 client`**.
5. In the **`Create Client ID`** dialog box, enter name of product in **`Product name`**. Click Next.
6. Select **`Web application`**. Enter http://server.local.network.com:8080/portal/googleAuth in **`Authorized Redirects`**

Note

You can have multiple URIs, that can be used as returning URIs from Google plus application.

7. Click **`Create client ID`**

Result

- The **Client ID for web applications** dialog box shows the client information such as the Client ID, Email address, Client Secret, Redirect URI and so on. These values are used during JBoss Portal configuration.

Note

For more details about registration process, see Google+ documentation

Report a bug

20.5.2. Configuring JBoss Portal for using OAuth Protocol with Google plus

Procedure 20.7. Configure properties in Google plus

- Change the file **configuration.properties** for configuring Google plus.

```
## Google
gatein.oauth.google.enabled=true
gatein.oauth.google.clientId=397260572677.apps.googleusercontent.com
gatein.oauth.google.clientSecret=pOWDqdu57CjplhrMmkrgcia5
gatein.oauth.google.redirectURL=${gatein.oauth.portal.url}/@@portal.c
ontainer.name@@/googleAuth
gatein.oauth.google.scope=https://www.googleapis.com/auth/userinfo.ema
il https://www.googleapis.com/auth/userinfo.profile
gatein.oauth.google.accessType=offline
```

Properties description

gatein.oauth.google.enabled

It enables integration with Google plus. Users can login and register with Google plus.

gatein.oauth.google.clientId

Client ID of your application. This is obtained from registration with Google as described in Section 20.5.1, “Registration of Portal application on Google”

gatein.oauth.google.clientSecret

Client Secret of your application. This is obtained from registration with Google as described in Section 20.5.1, “Registration of Portal application on Google”.

gatein.oauth.google.redirectURL

The URL used by Google to redirect after user confirms Google plus authorization screen (Step 3 of Section 20.2.2, “Login Workflow”). Normally you do not need to change this value as long as you changed property **gatein.oauth.portal.url** as described in Section 20.3, “Integrating OAuth with the Portal”.

gatein.oauth.google.accessType

Possible values are **offline** or **online**. The default value is **offline**. In offline access, the token can be used even if you are not logged into Google+ . It can be refreshed automatically by JBoss Portal.

Note

For more info about access types is https://developers.google.com/accounts/docs/OAuth2WebServer#offline .

gatein.oauth.google.scope

Scopes represent permissions required from users on Google authorization screen. This list of scope is used during user registration or authentication to JBoss Portal (portlet applications may require more scopes if they need it). It is recommended not to change this value and keep value as https://www.googleapis.com/auth/userinfo.email and https://www.googleapis.com/auth/userinfo.profile. Alternatively you can:

- Either use https://www.googleapis.com/auth/userinfo.email or https://www.googleapis.com/auth/userinfo.profile. In both cases, user will be able to login into JBoss Portal, but some information will be obtained from Google.
- Use more values by adding value at https://www.googleapis.com/auth/plus.login. However, it is likely that users will not allow your Portal application to access their informations. And having more values is only useful for quickstart portlet applications, but quickstart applications can ask for more scopes as per their requirement.
- It is user friendly for some quickstart portlets such as GoogleActivitiesPortlet and GoogleFriendsPortlet. But to use Google+ only for login (not using portlet applications), the scope https://www.googleapis.com/auth/plus.login is not needed.

After restart of JBoss Portal, your users should be able to register or login with their Google accounts as described in Section 20.2.1, “User Registration”

Report a bug

20.6. Integration of OAuth with Twitter

Twitter is a social network, that can be integrated with JBoss Portal. It support OAuth 1.0a and not OAuth 2.0 protocol. Integration of Twitter with JBoss Portal consists of:

- Registration of JBoss Portal application on Twitter.
- Configuring JBoss Portal for using OAuth Portocol with Twitter.

Report a bug

20.6.1. Registration of Portal application on Twitter

Assumption

- JBoss Portal is hosted on **`server.local.network.com`** as described in Section 20.3, “Integrating OAuth with the Portal” .

Procedure 20.8. Registering on Twitter

1. Register as Twitter developer on https://dev.twitter.com/.
2. Register new application on https://dev.twitter.com/apps.

3. Click **Create new application**.
4. Enter name and description of application
5. Enter the value for website as http://server.local.network.com:8080.
6. Enter Callback URL value as http://server.local.network.com:8080/portal/twitterAuth

Callback URL is a mandatory field

The Callback URL is required to recognize the application type as web application, and twitter does not accept http://localhost:8080/portal as a valid URL.

7. Click **Create your Twitter application**.
8. Click **Settings**. Select **Allow this application to be used to Sign in with Twitter**.

Note

Remember the value of clientId and clientSecret as they are required in JBoss Portal configuration.

Report a bug

20.6.2. Configuring JBoss Portal for using OAuth Protocol with Twitter

Procedure 20.9. Configure properties in Twitter

- Change the file **configuration.properties** for configuring Twitter.

```
## Twitter
gatein.oauth.twitter.enabled=true
gatein.oauth.twitter.clientId=JBBAkYseVCzStLhhOQ
gatein.oauth.twitter.clientSecret=Ihi1txdiasrjMJoisRFvBaUjVq1K87gJNKW7V
CbY
gatein.oauth.twitter.redirectURL=${gatein.oauth.portal.url}/@@portal.
container.name@@/twitterAuth
```

Properties description

gatein.oauth.twitter.enabled

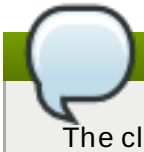
It enables integration with Twitter. Users can login and register with Twitter.

gatein.oauth.twitter.clientId

Client ID of your application. This is obtained from registration with Twitter as described in Section 20.6.1, “Registration of Portal application on Twitter”

gatein.oauth.twitter.clientSecret

Client Secret of your application. This is obtained from registration with Twitter as described in Section 20.6.1, “Registration of Portal application on Twitter”



Note

The client secret should not be shared among your portal users or other people.

gatein.oauth.twitter.redirectURL

The URL used by Twitter to redirect after user confirms Twitter authorization screen (Step 3 of Section 20.2.1, “User Registration”). Normally you do not need to change this value as long as you changed property **`gatein.oauth.portal.url`** as described in Section 20.3, “Integrating OAuth with the Portal”.

After restart of JBoss Portal, your users should be able to register or login with their Twitter accounts as described in Section 20.2.1, “User Registration” .

Report a bug

Chapter 21. Impersonation

It is useful for a portal administrator to have the option of temporarily logging onto the portal as another user, without knowing their password. For example, the admin user **root** wants to verify that user **mary** does not have permission to see page X or portlet Y on page Z. The Impersonate User feature of JBoss Portal allows portal administrators to interact with the portal in this way.

Report a bug

21.1. Using Impersonation

In **OrganizationManagementPortlet**, there is an action available in the user's list called Impersonate User.

Figure 21.1. Impersonate User

By clicking Impersonate User, the admin user initiating this feature is temporarily assigned the permissions of the selected user (for example, user mary). The impersonated user session starts, and the JBoss Portal UI will reflect the impersonated user's name and permission scheme.

There are two visual cues that Impersonate User mode is active:

- In the top right corner, the name of the impersonated user is followed by the username of the admin user in braces. For example: *Mary Kelly (root)*
- In the top left drop-down menu of the page **(UIStarToolbarPortlet)** a **Finish Impersonation** menu itemis shown instead of the default **Sign out** menu item. Click **Finish Impersonation** to terminate the impersonation session, and return the permission scheme back to the admin user. All portlets, and the JBoss Portal UI state is restored to the state before impersonation.

Because changes made to the portal will appear as though the impersonated user made the changes, only members of the **manager:/platform/administrators** group have permission to activate this feature. Who can access this feature is configurable through the **UserACL** component.

See Also:

- Section 25.2, “Overwrite Portal Default Permissions”

Report a bug

Chapter 22. Wildcard Membership Type

Wildcard (*) membership type is considered identical to *any* membership type.

If a user is assigned the wildcard membership type for a group, when a membership type for this group and the user is checked, then it is alway true *regardless of the membership type*.

Report a bug

22.1. Wildcard Membership Configuration and Initialization

The wildcard membership can not be created at runtime through the User Interface. It can only be initialized by configuring OrganizationDatabaseInitializer plug-in correctly.

Example 22.1. Wildcard Membership Initilization and Configuration

```
   <external-component-plugins>
    <target-
component>org.exoplatform.services.organization.OrganizationService</t
arget-component>
    <component-plugin>
      <name>init.service.listener</name>
      <set-method>addListenerPlugin</set-method>

<type>org.exoplatform.services.organization.OrganizationDatabaseInitial
izer</type>
      <description>this listener populate organization data for the
first launch</description>
      <init-params>
        …
        <object-param>
          <name>configuration</name>
          <description>description</description>
          <object
type="org.exoplatform.services.organization.OrganizationConfig">
            <field name="membershipType">
              <collection type="java.util.ArrayList">
                …
                <value>
                  <object
type="org.exoplatform.services.organization.OrganizationConfig$Membersh
ipType">
                    <field name="type">
                      <string>*</string>
                    </field>
                    <field name="description">
                      <string>any membership type</string>
                    </field>
                  </object>
                </value>
                …
              </collection>
            </field>
```

```
            ...
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
```

See Also:

- Chapter 11, *Predefined User Configuration*

Report a bug

22.2. Wildcard Membership API

The existing method behavior in the API is unchanged, to ensure backward compatibility is maintained. A new method in the **`GroupHandler`** interface is available to perform the wildcard membership type resolution.

The following method returns a collection of groups that the user has the specified membership type relationship, or the wildcard membership.

```
Collection resolveGroupByMembership(String userName, String
membershipType)
```

In the MembershipTypeHandler interface, the following method returns a collection of all membership types. In this particuar usage example, the wildcard membership would be the first element returned (if it exists).

```
Collection findMembershipTypes()
```

Report a bug

Part V. Mobile and Responsive Portal

Chapter 23. Mobile and Responsive Portal Site

A portal site optimized specifically for mobile and touch based devices is available. This site is designed with responsive principals to give the best user experience on desktops, tablets, and mobile phones.

This new site is accessed through http://localhost:8080/portal/mobile.

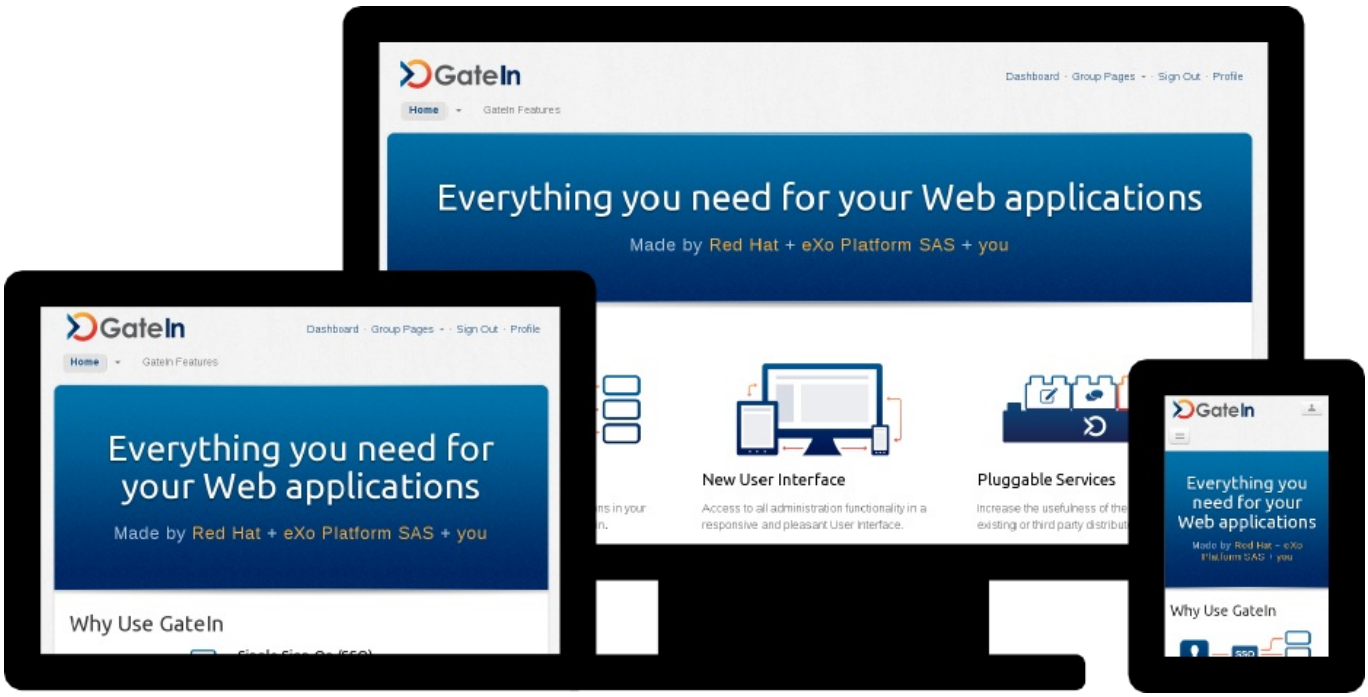


Figure 23.1. Responsive Site Comparisons

Feature Overview

Responsive Design

The site adapts to the width of the browser, and reported screen resolution of the device accessing the site. Unlike the classic portal site, the responsive site does not use fixed-width portlets.

Shared Layout (User Bar)

The Shared Layout bar (displayed at the top of the page for authenticated users) is now hidden to all users except Administrators.

The user bar can be configured to displayed for all users through the use of a portal property.

All User Bar functionality has been moved to a new responsive header portlet, except for page edit functionality and site navigation links.

Automatic Redirection

The redirection service has been configured by default to redirect mobile devices to the mobile site. If a user tries to access the site from a desktop, they should not be redirected and sent to the classic site.

Automatic Redirection is checked the first time the user accesses the site. The portal redirection service remembers which site the user was last using as their preferred site.

For more information about the redirect service and how to configure it, see *Site Redirection* in the Development guide.

Report a bug

23.1. Issues and Limitations

The following sections detail the known issues and limitations of the Mobile and Responsive Portal.

The Mobile and Responsive Portal is classed as a Technology Preview in Red Hat JBoss Portal 6.1, therefore there may be some features that require improvement. If you find an issue with the new Mobile and Responsive Portal, open a case through the Customer Portal by visiting https://access.redhat.com. Your feedback on how to improve the mobile experience is very welcome, and greatly appreciated.

Report a bug

23.1.1. Administration Functionality on Mobile Devices

The responsive site is designed to be presented to the portal user, as opposed to the portal administrator. Administration functions may be limited on touch and small-screen devices such as a smart phones.

- GTNPORTAL-3060: Support Node Management on Mobile Devices Node Management on iOS based devices will currently not work due to Safari for Mobile not supporting custom context menus. Some Android browsers support context menus by utilizing a long click. Node management on these browsers functions correctly.

Report a bug

23.1.2. Container Layouts and Page Configurations

The WebUI containers and layouts are designed for desktop browsers and large-screen page layouts only. As such they cannot be used to create responsive sites, except for the 'Row Page Layouts' and the 'Row Layout Containers'.

Only use the row layouts for responsive portlets and do not specify the width or height of the containers or portlets. The row layouts will give portlets the full width of the page and allow the portlet to handle how it should be rendered based on the browser's width.

If more complex layouts are required (such as column or mixed layouts) these will have to be done within the constraints of a single portlet and not through multiple portlets.

Report a bug

23.1.3. Group and User Sites

Group and user sites are shared between portal sites. There is currently no mechanism to customize or configure these sites to perform in one manner for the mobile site and another manner for the classic site.

Because the skin for these sites is set to the skin of the portal site that accesses it, these sites cannot be properly configured to work with the mobile site without affecting how the classic site interacts with it. While on the group and user sites, the shared layout/user bar portlets will be displayed to allow the user to navigate from one area to another.

Shared layout portlets may be enhanced in a future release so that they can be reused for both the responsive site and the classic site.

Note

The shared layout/user bar portlets are not optimal for mobile or touch based devices, but do work in a limited way. When navigating a menu, tapping once on a menu element will open the sub-menu for the element and tapping again will follow the link for the menu. While not ideal, this navigation style should allow shared layout user bar portlets to function on mobile.

Report a bug

23.1.4. Interchanging Mobile and Responsive Site Skins

The responsive skin has been designed for the responsive/mobile site and the classic skin has been designed for the mobile site. It is recommended to use the skin on the site it was designed for, using the skin on another site may result in some minor graphical issues or inconsistencies. It may be recommended to remove the ability for users to specify their skin preferences.

Report a bug

23.2. Configuring the Mobile Site

There are a few options for configuring the mobile site.

There are a few options available under site properties which are especially important for the mobile site.

Site Properties

Viewport Context

The viewport specifies things like the width, height and zoom level a browser should use when loading a mobile site. The value specified in this property will be directly added to the viewport meta tag for the portal. The default value used in the mobile site is "initial-scale-1.0" which will set the initial width to that of the browser and still allow the user to zoom into the page.

Shared Layout

This can be set to 'Admin Only' or 'All Users'. Since the default shared layout portlets are not necessarily optimized for mobile devices, we have by default disabled the shared layout portlets for all users except administrators.

Responsive Header Portlet Preferences

- enable.dashboard.link: this value specifies if the dashboard link should be displayed in the header portlet or not. By default the mobile site is configured to display the dashboard, but if required it can be hidden by setting this preference to 'false'.

- enable.grouppages.link: this value specifies if the group pages and menu should be displayed in the header portlet or not. By default the mobile site is configured to display the group pages, but if required it can be hidden by setting this preference to 'false'.
- level: this, like the navigation portlet, specifies how many levels of nodes should be fetched at a time when updating the menu via a ajax request. Accepts any integer value greater than 1.

For more information about the redirect service and how to configure it, see *Site Redirection* in the Development guide.

Report a bug

Chapter 24. Site Redirection

GateIn Portal includes a mechanism to redirect user to a different site within the portal based on the characteristics of their browser. This is especially useful for redirecting users to a mobile optimized site if you detect they are accessing the portal from a mobile device.

Redirections can be based on various criteria:

- The user agent string of the browser accessing the site
- Any properties which can be determined via javascript. This can include things like screen size, pixel density and if the device supports touch or not.

We also include options to map the nodes between one site and another when performing the redirect. This allows for greater control over what page within the site a user will end up on when a redirect is performed.

The following sections will briefly explain how to configure and setup the service using xml the configuration files.

For details on how to configure site redirects using the administration user interface, see topic *Manage Site Redirections* in *User Guide*.

Note

Redirects are only performed once for a user and stored in a cookie. The site redirect service will always use the value stored in the cookie regardless if the redirect service configuration has changed. This allows for users to specify their own preferences without being redirected to their non-preferred site when the redirection rules change. For costly browser based property detection, this also means that this check is only performed once for a user and not on each subsequent access.

Report a bug

24.1. Configuring Site Redirections in XML

All of the XML configuration options to add a redirectable, or alternative site is done in the **`portal.xml`** file for that particular site.

Note

By default, the **`portal.xml`** configuration file is only read the first time the portal is ever started. To make changes after the portal has already been started, follow the directions in the *Data Import Strategy* chapter of the *Development Guide*.

Report a bug

24.1.1. Adding a Redirectable Site

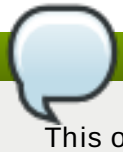
To add the site **`mobile`** as an alternative to the **`classic`** site you would modify the **`portal.xml`** file for the 'classic' site to include the following:

Example 24.1. Adding a Redirectable Site

```
<portal-redirects>
    <portal-redirect>
      <redirect-site>mobile</redirect-site> <!-- (1) -->
      <name>Mobile</name> <!-- (2) -->
      <enabled>true</enabled> <!-- (3) -->
    </portal-redirect>
</portal-redirects>
```

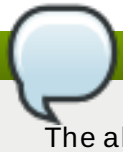
1. The name of the site to redirect to
2. The name given to the redirect
3. If the redirect should currently be enabled or not

Now when accessing the **classic** site you should notice a link at the bottom to the **mobile** site. If the user clicks this link, they will be redirected to that site and their preference for the alternative site will be stored. This means the next time the user tries to access the **classic** site they will automatically be redirected to the mobile site instead.



Note

This only creates a redirection link between already existing sites. You will still need to create the sites manually.



Note

The above steps only create a one way link. If the user also wants to make the **classic** site an alternative for the **mobile** site they will have to configure the **portal.xml** for the mobile site. **It is highly recommended to also configure the redirect on the other site to allow the user to return to the original size as well as to set their preference to the original site.**

Report a bug

24.2. Automatic Redirection Based on User Agent String

The example in Section 24.1.1, “Adding a Redirectable Site” describes how to add an option for a redirect, however user intervention is required to specify which site to display. The site redirection service can be configured to redirect a user automatically from one site to another. This can be based on either the user agent string of the clients browser or through a device detection page which can gather any data which can be gathered through javascript.

When a website is accessed, the browser normally sends a user agent string which describes in part the type and version of the browser used. Automatic redirects can be configured based on the values contained within this string, which enhances the user experience when viewing the portal site on mobile devices.

An example configuration which will preform a redirect if the user agent string contains the case-insensitive string **"foo"** or the case-sensitive string **"Bar"** and does not contain the case-sensitive string **"baz"** :

Example 24.2. Example Redirect Based on User Agent Strings

```
<portal-redirects>
    <portal-redirect>
      <redirect-site>test</redirect-site>
      <name>Test</name>
      <enabled>true</enabled>
      <conditions>
        <condition>
          <name>Test Condition</name> <!-- (1) -->
          <user-agent>
            <contains>(?i)foo</contains> <!-- (2) -->
            <contains>Bar</contains> <!-- (2) -->
            <does-not-contain>baz</does-not-contain> <!-- (3) -->
          </user-agent>
       </condition>
      </conditions>
    </portal-redirect>
  </portal-redirects>
```

1. The name of the redirect condition
2. A list of strings to check if they exist within the user agent
3. A string to make sure is not within the user agent string

The condition is considered to pass if any of the contains options is matched and if none of the does-not-contain options are matched.

Note

The format of the strings is the Java Regex Pattern format. This allows for powerful pattern matching capabilities. Please see the java regex documentations for more details.

Report a bug

24.3. Automatic Redirection Based on Device Properties

In some rare cases, using user agent strings may not be enough. You may need to determine some characteristics retrieve from the user's browser itself. The way we can do this is to send the user to another page first, run some javascript to gather some information and return this to the redirection service. This allows for some more powerful detection and redirection, but at the cost of extra complexity and performance degradation.

Note

It is strongly recommended to use user agent based detection whenever possible. Redirection based on device properties should be avoided if possible due to performance reasons.

Report a bug

24.3.1. The Device Detection Page

The jsp page which does the detection is located:

```
gatein.ear/portal.war/device/detection.jsp
```

By default this page only detects a couple of simple options like screen width and height. To add more options, this page can be modified and calls to the javascript method **`addParameter()`** will make new options and values available to the service.

A simple configuration to detect that the **`touch.enabled`** property specified in the **`detection.jsp`** file returned the value **`true`** .

Report a bug

24.3.2. Device Properties Based Redirection XML Configuration

Example 24.3. Example Redirection Based on Device Properties

```
<portal-redirects>
   <portal-redirect>
     <redirect-site>test</redirect-site>
     <name>Test</name>
     <enabled>true</enabled>
     <conditions>
       <condition>
         <name>Test Condition</name>
         <user-agent>
           <contains>.*</contains> <!-- (1) -->
         </user-agent>
         <device-properties>
          <device-property>
           <property-name>touch.enabled</property-name> <!-- (2) -->
             <equals>true</equals> <!-- (3) -->
           </device-property>
         </device-properties>
       </condition>
     </conditions>
   </portal-redirect>
 </portal-redirects>
```

1. Java Regex Pattern that specifies that any user agent string will work.

2. Specifies the name of the property retrieved from the detection page to check against.
3. Specifies the condition to check this property. The options include: **`equals`**, **`greater-than`**, **`less-than`**, and **`matches`** (which uses Java Pattern Regex format).

Multiple options can be used against a single device property. They must all match to satisfy the condition. This allows for detecting a value within a range by using both greater-than and less-than options.

Example 24.4. Multiple Property Options

This example shows how user agent string and device property detection can be used together to match any user agent string.

If multiple **`device-property`** options are used, then they must all match to satisfy the condition.

```
<condition>
  <name>Test Condition</name>
  <user-agent>
    <contains>.*</contains>
  </user-agent>
  <device-properties>
    <device-property>
      <property-name>screen.width</property-name>
      <greater-than>320</greater-than>
      <less-than>1024</less-than>
    </device-property>
  </device-properties>
</condition>
```

An alternative configuration to what is described in Example 24.4, “Multiple Property Options” is to restrict based on the user agent string. In this scenario, the service will only ever redirect the user to the device detection page if the user agent string values pass. Narrowing the user agent string is strongly advised to prevent unnecessary redirects to the device detection page.

Report a bug

24.3.3. Multiple Redirect Conditions

Multiple redirect conditions can be specified. The conditions are checked sequentially and perform the redirect when it encounters a condition which passes.

For example, the following will perform the redirect if the user agent string contains the string **`"foo"`** . If the user agent string does not contain **`"foo"`** it will check if it contains **`"bar"`** and the browser detection pare returns a **`touch.enabled`** property of **`true`** . If neither of these conditions pass, the redirect will not happen and the user's preference will be set to the original site.

Example 24.5. Multiple Redirect Conditions Example

```
<conditions>
  <condition>
    <name>Check Foo</name>
    <user-agent>
      <contains>foo</contains>
```

```
    </user-agent>
  </condition>
  <condition>
    <name>Check Touch</name>
    <user-agent>
      <contains>bar</contains>
    </user-agent>
    <device-properties>
      <device-property>
        <property-name>touch.enabled</property-name>
        <equals>true</equals>
      </device-property>
    </device-properties>
  </condition>
</conditions>
```

Report a bug

24.4. Mapping Page Nodes In Redirects

When redirecting between sites, the user is actually redirected from one page node on one site to a node on another site. There are a number of configuration options available to handle node redirects.

Report a bug

24.4.1. Explicit Node Mappings

The Administrator can configure explicit node mappings between the original site and the redirect site. Node mappings are always evaluated first when performing a redirect.

For example, if a redirect between **`classic`** and **`mobile`** sites is required, the **`http://localhost:8080/portal/classic/home/pageA`** URL could get redirected to the **`http://localhost:8080/portal/mobile/pageB`** URL.

Report a bug

24.4.2. Node Name Matching

Node name matching is enabled by default, but can be disabled if required.

If the node is not specified in the node mappings, then the redirect can be configured to try and match the nodes from the original site to a node of the same path on the redirect site.

If a redirect is configured between the **`classic`** and **`mobile`** sites and the user tries to access **`http://localhost:8080/portal/classic/home/pageXYZ"`** and the node **`/home/pageXYZ`** exists on the mobile site, the user is redirected to **`http://localhost:8080/portal/classic/mobile/pageXYZ"`**. If the node **`/home/pageXYZ`** does not exist on the redirect site, then the service will evaluate the unresolved node settings to determine where the redirect should go.

Report a bug

24.5. Resolving Unresolved Nodes

If the node is not listed in the node mappings, and the node name mappings could not resolve the node, then the node is considered currently *unresolved*. Using the **<unresolved-nodes>** XML element, the service can be configured to handle unresolved nodes in a variety of ways.

REDIRECT - Redirect Anyway

If the node is not resolved, perform the redirect and use the node path from the original site.

For example, if a redirect is to occur between **classic** and **mobile** and the user accesses **http://localhost:8080/portal/classic/home/pageABC** but **pageABC** does not exist on the redirect site, the user will still be redirected to **http://localhost:8080/portal/mobile/home/pageABC** . This will use the portal's node resolving configuration, which by default will resolve to the next available node in the path.

NO_REDIRECT - Abort the Redirect

If the node does not exist on the redirect site, the redirect will not take place and the user will continue on to the original URI.

ROOT - Redirect to the Sites Root

If the node does not exist on the redirect site, the user is redirected to the root of the redirect site.

For example, with a redirect between **classic** and **mobile** if the node **/home/PageA/PageB** does not exist on the redirect site, the user is redirected to http://localhost:8080/portal/mobile even if **/home/PageA** exists on the mobile site.

COMMON_ANCESTOR_NAME_MATCH - Redirect Based on a Common Ancestor Node

This option is similar to the REDIRECT option, and behaves the same way if the portal is using the default node resolver.

For example, if the node on the original site is **/home/PageA/PageB** this option will first check if **/home/PageA/PageB** exists on the redirect site, then **/home/PageA** , then **/home** and will finally resolve to the root of the redirect site if no match can be found.

Example 24.6. Node Mapping XML Example

```
<portal-redirect>
  <redirect-site>mobile</redirect-site>
  <name>Mobile</name>
  <enabled>true</enabled>
  <conditions>
    <!-- Irrelevant content removed for readability -->
  </conditions>
  <node-mapping>
    <node-map> <!-- (1) -->
      <origin-node>home</origin-node>
      <redirect-node>mobileHome</redirect-node>
    </node-map>
    <node-map> <!-- (2) -->
      <origin-node>foo</origin-node>
      <redirect-node>bar</redirect-node>
```

```
    </node-map>
    <use-node-name-mapping>true</user-node-name-mapping> <!-- (3) -->
    <unresolved-nodes>NO_REDIRECT</unresolved-nodes> <!-- (4) -->
  <node-mapping>
</portal-redirect>
```

1. **/home** on the original site will always be resolved to **/mobileHome** on the redirect site
2. **/foo** on the original site will always be resolved to **/bar** on the redirect site
3. Node name matching will be attempted.
4. If the origin node is not resolved from either the mapping or the name matching, then no redirect will occur.

Report a bug

24.6. Disabling Redirect Handler

The Site Redirection service can impact performance because it imposes additional overheads on the server when processing client requests. If this service is not required, it can be disabled using the following methods.

- Removing the redirect portlet from the site layout.
- Removing or commenting the following code from ***JPP_HOME*/standalone/configuration/gatein/controller.xml**

```
<!-- The portal redirect handler -->
<route-param qname="gtn:handler">
  <value>siteRedirect</value>
</route-param>
```

Report a bug

Part VI. Portal Configuration

Chapter 25. Portal Configuration

25.1. Configuring Permissions

The default permission configuration for the portal is defined through the **`org.exoplatform.portal.config.UserACL`** component configuration in the ***`JPP_HOME`*`/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal-configuration.xml`** file.

It defines nine permissions types:

`super.user`

The super user has all the rights on the platform, this user is referred to as *root*.

`portal.administrator.groups`

Any member of those groups are considered administrators. Default value is **`/platform/administrators`**.

`portal.administrator.mstype`

Any user with that membership type would be considered administrator or the associated group. Default value is **`manager`**.

`portal.creator.groups`

This list defines all groups that will be able to manage the different portals. Members of this group also have the permission to create new portals. The format is **`membership:/group/subgroup`**.

`user.impersonate.groups`

This list defines all groups that will be able to impersonate as other user. The format is **`membership:/group/subgroup`**.

`navigation.creator.membership.type`

Defines the membership type of group managers. The group managers have the permission to create and edit group pages and they can modify the group navigation.

`guests.group`

Any anonymous user automatically becomes a member of this group when they enter the public pages.

`mandatory.groups`

Groups that cannot be deleted.

`mandatory.mstypes`

Membership types that cannot be deleted.

Example 25.1. Permission configuration

```
<component>
  <key>org.exoplatform.portal.config.UserACL</key>
```

```
  <type>org.exoplatform.portal.config.UserACL</type>
  <init-params>

    <value-param>
      <name>super.user</name>
      <description>administrator</description>
      <value>root</value>
    </value-param>

    <value-param>
      <name>portal.creator.groups</name>
      <description>groups with membership type have permission to
manage portal</description>

<value>*:/platform/administrators,*:/organization/management/executive
-board</value>
    </value-param>

    <value-param>
      <name>navigation.creator.membership.type</name>
      <description>specific membership type have full permission with
group navigation</description>
      <value>manager</value>
    </value-param>

    <value-param>
      <name>guests.group</name>
      <description>guests group</description>
      <value>/platform/guests</value>
    </value-param>

    <value-param>
      <name>access.control.workspace</name>
      <description>groups with memberships that have the right to
access the User Control Workspace</description>

<value>*:/platform/administrators,*:/organization/management/executive
-board</value>
    </value-param>

  </init-params>
</component>
```

Report a bug

25.2. Overwrite Portal Default Permissions

When creating custom portals and portal extensions it is possible to override the default configuration by using **`org.exoplatform.portal.config.PortalACLPlugin`**, configuring it as an external component plug-in of **`org.exoplatform.portal.config.UserACL`** service:

```
<external-component-plugins>
   <target-component> org.exoplatform.portal.config.UserACL</target-
```

```
component>
    <component-plugin>
      <name> addPortalACLPlugin </name>
      <set-method> addPortalACLPlugin</set-method>
      <type>org.exoplatform.portal.config.PortalACLPlugin</type>
      <description>setting some permission for portal</description>
      <init-params>
        <values-param>
          <name>access.control.workspace.roles</name>
          <value>*:/platform/administrators</value>
          <value>*:/organization/management/executive-board</value>
        </values-param>
        <values-param>
          <name>portal.creation.roles</name>
          <value>*:/platform/administrators</value>
          <value>*:/organization/management/executive-board</value>
        </values-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
```

Report a bug

25.3. Portal Navigation

25.3.1. Configuring Portal Navigation

There are three types of navigation available to portal users:

- Portal_Navigation
- Group_Navigation
- User_Navigation

These navigators are configured using the standard XML syntax in **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal-configuration.xml**.

```
<component>
   <key>org.exoplatform.portal.config.UserPortalConfigService</key>
   <type>org.exoplatform.portal.config.UserPortalConfigService</type>
   <component-plugins>
     <component-plugin>
       <name>new.portal.config.user.listener</name>
       <set-method>initListener</set-method>
       <type>org.exoplatform.portal.config.NewPortalConfigListener</type>
       <description>this listener init the portal
configuration</description>
       <init-params>
          <value-param>
             <name>default.portal</name>
             <description>The default portal for checking db is empty or
```

```
not</description>
            <value>classic</value>
         </value-param>
         <value-param>
            <name>page.templates.location</name>
            <description>the path to the location that contains Page
templates</description>
            <value>war:/conf/portal/template/pages</value>
         </value-param>
         <value-param>
            <name>override</name>
            <description>The flag parameter to decide if portal
metadata is overriden on restarting server</description>
            <value>false</value>
         </value-param>
         <object-param>
            <name>site.templates.location</name>
            <description>description</description>
            <object
type="org.exoplatform.portal.config.SiteConfigTemplates">
               <field name="location">
                  <string>war:/conf/portal</string>
               </field>
               <field name="portalTemplates">
                  <collection type="java.util.HashSet">
                     <value><string>basic</string></value>
                     <value><string>classic</string></value>
                  </collection>
               </field>
               <field name="groupTemplates">
                  <collection type="java.util.HashSet">
                     <value><string>group</string></value>
                  </collection>
               </field>
               <field name="userTemplates">
                  <collection type="java.util.HashSet">
                     <value><string>user</string></value>
                  </collection>
               </field>
            </object>
         </object-param>
         <object-param>
            <name>portal.configuration</name>
            <description>description</description>
            <object
type="org.exoplatform.portal.config.NewPortalConfig">
               <field name="predefinedOwner">
                  <collection type="java.util.HashSet">
                     <value><string>classic</string></value>
                  </collection>
               </field>
               <field name="ownerType">
                  <string>portal</string>
               </field>
               <field name="templateLocation">
                  <string>war:/conf/portal/</string>
```

```
                </field>
                <field name="importMode">
                   <string>conserve</string>
                </field>
             </object>
          </object-param>
          <object-param>
             <name>group.configuration</name>
             <description>description</description>
             <object
type="org.exoplatform.portal.config.NewPortalConfig">
                <field name="predefinedOwner">
                   <collection type="java.util.HashSet">
                      <value><string>/platform/administrators</string>
</value>
                      <value><string>/platform/users</string></value>
                      <value><string>/platform/guests</string></value>
                      <value><string>/organization/management/executive-
board</string></value>
                   </collection>
                </field>
                <field name="ownerType">
                   <string>group</string>
                </field>
                <field name="templateLocation">
                   <string>war:/conf/portal</string>
                </field>
                <field name="importMode">
                   <string>conserve</string>
                </field>
             </object>
          </object-param>
          <object-param>
             <name>user.configuration</name>
             <description>description</description>
             <object
type="org.exoplatform.portal.config.NewPortalConfig">
                <field name="predefinedOwner">
                   <collection type="java.util.HashSet">
                      <value><string>root</string></value>
                      <value><string>john</string></value>
                      <value><string>mary</string></value>
                      <value><string>demo</string></value>
                      <value><string>user</string></value>
                   </collection>
                </field>
                <field name="ownerType">
                   <string>user</string>
                </field>
                <field name="templateLocation">
                   <string>war:/conf/portal</string>
                </field>
                <field name="importMode">
                   <string>conserve</string>
                </field>
             </object>
```

```
          </object-param>
       </init-params>
    </component-plugin>
  </component-plugins>
</component>
```

This XML configuration defines where in the portal's **WAR** to look for configuration settings, and which portals, groups, and user specific views to include in *portal/group/user* navigation.

The first time the portal is launched those files will be used to create an initial navigation. That information will then be stored in the JCR content repository and can be modified and managed from the portal UI.

Report a bug

25.3.2. Setting up Navigation

Each portal, groups and users navigation is indicated by a <object-params> configuration directive.

Example 25.2. Navigation Configuration Directives

```
<object-param>
   <name>portal.configuration</name>
   <description>description</description>
   <object type="org.exoplatform.portal.config.NewPortalConfig">
      <field name="predefinedOwner">
         <collection type="java.util.HashSet">
            <value><string>classic</string></value>
         </collection>
      </field>
      <field name="ownerType">
         <string>portal</string>
      </field>
      <field name="templateLocation">
         <string>war:/conf/portal/</string>
      </field>
      <field name="importMode">
         <string>conserve</string>
      </field>
   </object>
</object-param>
```

predefinedOwner defines the navigation owner, portal will look for the configuration files in folder with this name, if there is no suitable folder, a default portal will be created with name is this value.

ownerType defines the type of portal navigation. It may be a portal, group, or user.

templateLocation defines the classpath to all portal configuration files

importMode is the mode for navigation import. There are 4 types of import mode:

- ***conserve***: Import data when it does not exist, otherwise do nothing.
- ***insert***: Import data when it does not exist, otherwise performs a strategy that adds new data only.

- ***merge***: Import data when it does not exist, update data when it exists.
- ***rewrite***: Overwrite data whatsoever.

Based on these parameters, the portal looks for the configuration files, and creates the relevant portal navigation, pages, and data import strategy.

- Portal configuration files are stored in the folder structures **{templateLocation}/{ownerType}/{predefinedOwner}.**
- Portal navigations are defined in the **navigation.xml** file.
- Portal pages are defined in **pages.xml** file.
- Portal configuration is defined in **portal.xml** file.

With the above configuration, the portal will look for all configuration files contained within the **war:/conf/portal/portal/classic** directory.

Report a bug

25.3.3. Portal Navigation

The portal navigation incorporates the pages that can be accessed even when a user is not logged in (assuming the applicable permissions allow public access). For example; several portal navigations could be used when a company has multiple trademarks, and websites are set up for each of them.

The *Classic* portal is configured by three XML files in the ***JPP_HOME*/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal/classic** directory:

portal.xml

This file describes the layout and portlets that will be shown on all pages. Usually the layout contains the banner, footer, menu and breadcrumbs portlets. The portal is extremely configurable as every view element (even the banner and footer) is a portlet.

```
<portal-config
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_objects_1
_2 http://www.gatein.org/xml/ns/gatein_objects_1_2"
 xmlns="http://www.gatein.org/xml/ns/gatein_objects_1_2">
 <portal-name>classic</portal-name>
 <locale>en</locale>
 <access-permissions>Everyone</access-permissions>
 <edit-permission>*:/platform/administrators</edit-permission>
 <properties>
   <entry key="sessionAlive">onDemand</entry>
   <entry key="showPortletInfo">1</entry>
 </properties>

 <portal-layout>
   <portlet-application>
      <portlet>
```

```
         <application-ref>web</application-ref>
         <portlet-ref>BannerPortlet</portlet-ref>
         <preferences>
            <preference>
               <name>template</name>

<value>par:/groovy/groovy/webui/component/UIBannerPortlet.gtmpl</
value>
               <read-only>false</read-only>
            </preference>
         </preferences>
      </portlet>
      <access-permissions>Everyone</access-permissions>
      <show-info-bar>false</show-info-bar>
   </portlet-application>

   <portlet-application>
      <portlet>
         <application-ref>web</application-ref>
         <portlet-ref>NavigationPortlet</portlet-ref>
      </portlet>
      <access-permissions>Everyone</access-permissions>
      <show-info-bar>false</show-info-bar>
   </portlet-application>

   <portlet-application>
      <portlet>
         <application-ref>web</application-ref>
         <portlet-ref>BreadcumbsPortlet</portlet-ref>
      </portlet>
      <access-permissions>Everyone</access-permissions>
      <show-info-bar>false</show-info-bar>
   </portlet-application>

   <page-body> </page-body>

   <portlet-application>
      <portlet>
         <application-ref>web</application-ref>
         <portlet-ref>FooterPortlet</portlet-ref>
         <preferences>
            <preference>
               <name>template</name>

<value>par:/groovy/groovy/webui/component/UIFooterPortlet.gtmpl</
value>
               <read-only>false</read-only>
            </preference>
         </preferences>
      </portlet>
      <access-permissions>Everyone</access-permissions>
      <show-info-bar>false</show-info-bar>
   </portlet-application>
```

```
 </portal-layout>

</portal-config>
```

It is also possible to apply a nested container that can also contain portlets. Row, column or tab containers are then responsible for the layout of their child portlets.

Each application references a portlet using the id **portal#{portalName}:/{portletWarName}/{portletName}/{uniqueId}**.

Use the **page-body** tag to define where the portal should render the current page.

The defined *classic* portal is accessible to "Everyone" (at **/portal/classic**) but only members of the group **/platform/administrators** can edit it.

navigation.xml

This file defines all the navigation nodes the portal will have. The syntax is simple and uses nested node tags. Each node references a page defined in **pages.xml** file.

To create node labels for each language, use the label tag with the **xml:lang** attribute with the value of the relevant locale.

Otherwise, to have the node label localized by resource bundle files, use the **#{...}** syntax: the enclosed property name serves as a key that is automatically passed to internationalization mechanism which replaces the literal property name with a localized value from the associated properties file matching the current locale.

```
 <node-navigation
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_objects_1_
2 http://www.gatein.org/xml/ns/gatein_objects_1_2"
   xmlns="http://www.gatein.org/xml/ns/gatein_objects_1_2">
   <priority>1</priority>
   <page-nodes>
      <node>
         <name>home</name>
         <label xml:lang="en">Home</label>
         <page-reference>portal::classic::homepage</page-
reference>
      </node>
      <node>
         <name>sitemap</name>
         <label xml:lang="en">SiteMap</label>
         <visibility>DISPLAYED</visibility>
         <page-reference>portal::classic::sitemap</page-reference>
      </node>
<!-- Additional content removed for readability -->
   </page-nodes>
</node-navigation>
```

Warning

For top nodes, the **uri** and the **name** of your navigation nodes must have the *same* value. For other nodes the **uri** is a relative path.

For example; ***contentmanagement/fileexplorer*** where '**contentmanagement**' is the name of the parent node and '**fileexplorer**' is the name of the node (*<name>fileexplorer</name>*).

This navigation tree can be accessed from within portlets. This allows different presentations to be provided within portlets such as breadcrumb menus, sitemaps, or pop-up menus. Further examples of this can be found in the source tree of the portal itself.

The <priority> directive is an optional property that controls navigation ordering priority within the UI. The greater the value, the more weight the navigation has when the toolbar renders. If the directive is not declared, the element is allocated a default value of **-1**, which sets the navigation element to "undefined" when editing the navigation UI.

pages.xml

This configuration file structure is very similar to **portal.xml** and can also contain container elements. Further usage examples of container elements can be found in **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal/sharedlayout.xml**.

Each portlet application can decide whether to render the portlet border, the window state, the icons, or the portlet mode.

```
<page-set
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_objects_1_
2 http://www.gatein.org/xml/ns/gatein_objects_1_2"
   xmlns="http://www.gatein.org/xml/ns/gatein_objects_1_2">
   <page>
      <name>homepage</name>
      <title>Home Page</title>
      <access-permissions>Everyone</access-permissions>
      <edit-permission>*:/platform/administrators</edit-
permission>
      <portlet-application>
         <portlet>
            <application-ref>web</application-ref>
            <portlet-ref>HomePagePortlet</portlet-ref>
            <preferences>
               <preference>
                  <name>template</name>

<value>system:/templates/groovy/webui/component/UIHomePagePortlet
.gtmpl</value>
                  <read-only>false</read-only>
               </preference>
            </preferences>
```

```
         </portlet>
         <title>Home Page portlet</title>
         <access-permissions>Everyone</access-permissions>
         <show-info-bar>false</show-info-bar>
         <show-application-state>false</show-application-state>
         <show-application-mode>false</show-application-mode>
      </portlet-application>
   </page>

   <page>
      <name>sitemap</name>
      <title>Site Map</title>
      <access-permissions>Everyone</access-permissions>
      <edit-permission>*:/platform/administrators</edit-
permission>
      <portlet-application>
         <portlet>
            <application-ref>web</application-ref>
            <portlet-ref>SiteMapPortlet</portlet-ref>
         </portlet>
         <title>SiteMap</title>
         <access-permissions>Everyone</access-permissions>
         <show-info-bar>false</show-info-bar>
      </portlet-application>
   </page>
   ...
</page-set>
```

Report a bug

25.3.4. Group Navigation

Group navigations are dynamically added to the user navigation at login. This allows users to see the pages assigned to any groups they belong to in the menu.

The group navigation menu is configured in the **navigation.xml** and **pages.xml** files. The syntax used in these files is the same as those covered in the portal navigation directives covered in Section 25.3.3, “Portal Navigation”.

Group navigation directives are located in the **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/group/*group-name-path*/** directory. For example, **portal.war/WEB-INF/conf/portal/group/platform/administrators/** for the admin group.

Report a bug

25.3.5. User Navigation

User navigation is the set of nodes and pages that are owned by a user. They are part of the user's dashboard.

Two files configure the user navigation (**navigation.xml** and **pages.xml**). They are located in the directory**JPP_HOME/gatein.ear/portal.war/WEB-INF/conf/portal/user/{userName}**.

The example below shows a dashboard with all of the default gadgets included, as well as an extra currency converter gadget sourced from Google Gadgets.

```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/gadgets.xml" parse="text"/>
```

Report a bug

25.4. Default Configuration for JBoss Portal

25.4.1. Setting up Default Configuration for JBoss Portal

The default home page URL is **`http://{hostname}:{port}/portal/`**. There may be multiple independent portal containers deployed in parallel at any given time, each of which has its root context (**`http://{hostname}:{port}/sample-portal/`**, for example).

Each portal container is internally composed of one or more 'portals'. This is because there needs to be at least one such portal available. The default portal is called '*Classic*'. When accessing the default URL, you are automatically directed to the '*Classic*' portal.

The default portal performs another important task. When starting up the portal for the first time, its JCR database (where portal runtime-configurable settings are stored) will be empty . The default portal detects this and triggers automatic data initialization.

Report a bug

25.4.2. Configuring Classic Portal

The following example configuration can be found at: ***`JPP_HOME`*`/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal-configuration.xml`**.

```
<component>
  <key>org.exoplatform.portal.config.UserPortalConfigService</key>
  <type>org.exoplatform.portal.config.UserPortalConfigService</type>
  <component-plugins>
    <component-plugin>
      <name>new.portal.config.user.listener</name>
      <set-method>initListener</set-method>
      <type>org.exoplatform.portal.config.NewPortalConfigListener</type>
      <description>this listener init the portal
configuration</description>
      <init-params>
        <value-param>
          <name>default.portal</name>
          <description>The default portal for checking db is empty or
not</description>
          <value>classic</value>
        </value-param>
        ...
```

```
      </init-params>
    </component-plugin>
  </component-plugins>
</component>
```

In this example the *Classic* portal has been set as the default.

Notice that the **`NewPortalConfigListener`** ***`component-plug-in`*** is used to add configuration to **`UserPortalConfigService`**, which is designed in this way to allow other components to add configuration to it.

Components, *component-plugins*, and *init-params* are explained in *The eXo Kernel* in the Development Guide.

Report a bug

25.4.3. Using Component Plugins

In some cases, portal definitions and portal template definitions are defined but not used any more. To delete them, it is possible to add a component plug-in to ***`JPP_HOME`***`/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal-configuration.xml` which detects these definitions and cleans up the **`portal-configuration.xml`** file automatically.

Example 25.3. Using Component Plugins

```
<external-component-plugins>
  <target-
component>org.exoplatform.portal.config.UserPortalConfigService</target
-component>
  <component-plugin>
    <name>new.portal.config.user.listener</name>
    <set-method>deleteListenerElements</set-method>
    <type>org.exoplatform.portal.config.NewPortalConfigListener</type>
    <description>this listener delete some predefined portal and
templates configuration</description>
    <init-params>
      <object-param>
        <name>site.templates.location</name>
        <description>description</description>
        <object
type="org.exoplatform.portal.config.SiteConfigTemplates">
          <field name="portalTemplates">
            <collection type="java.util.HashSet">
              <value>
                <string>basic</string>
              </value>
              <value>
                <string>classic</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
      <object-param>
```

```
        <name>portal.configuration</name>
        <description>description</description>
        <object type="org.exoplatform.portal.config.NewPortalConfig">
          <field  name="predefinedOwner">
            <collection type="java.util.HashSet">
              <value><string>classic</string></value>
            </collection>
          </field>
          <field name="ownerType"><string>portal</string></field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

Report a bug

25.4.4. Setting up Information Bar

You can set the info bar shown by default for portlets of a portal by adding an <entry> directive to the <properties> configuration of the **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/portal/classic/portal.xml`** file.

```
<properties>
  <entry key="showPortletInfo">1</entry>
</properties>
```

There are two values for "showPortletInfo": 0 and 1. If the value is 1, the info bar of portlets is shown by default. If the value is 0, it is hidden.

Report a bug

25.4.5. Disabling Portal Container

Once you have created a custom portal container that meets requirements, consider disabling a default portal container that is no longer required.

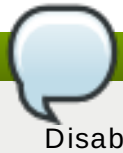
Procedure 25.1. Disable an Unused Portal Container

1. Add the following configuration to the **`configuration.xml`** of the custom extension in order to disable a portal container:

```
<!-- Comment 1 -->
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_0.xsd
http://www.exoplaform.org/xml/ns/kernel_1_1.xsd"
  xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd">
  <external-component-plugins>
<!-- The full qualified name of the PortalContainerConfig -->
    <target-
```

```
component>org.exoplatform.container.definition.PortalContainerConfi
g
    </target-component>
    <component-plugin>
<!-- The name of the plug-in -->
        <name>Add PortalContainer Definitions</name>
<!-- (existing configuration for new portal container) -->
    </component-plugin>
    <component-plugin>
<!-- The name of the plug-in -->
          <name>Disable a PortalContainer</name>
<!-- The name of the method to call on the PortalContainerConfig in
order to register the changes on the PortalContainerDefinitions -->
          <set-method>registerDisablePlugin</set-method>
<!-- The full qualified name of the
PortalContainerDefinitionDisablePlugin -->

<type>org.exoplatform.container.definition.PortalContainerDefinitio
nDisablePlugin</type>
          <init-params>
<!-- The list of the name of the portal containers to disable -->
        <values-param>
          <name>names</name>
          <value>$PORTAL_NAME</value>
        </values-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</configuration>
```

2. The portal name declared in the <values-param> directive will no longer be available at **`http://{hostname}:{port}/portal`**.



Note

Disabling the default portal container is possible, but not recommended. Some functions, such as WSRP or Services Management, depend on the default portal container to be deployed. These functions will no longer work if the default portal container is disabled.

Report a bug

25.5. Internationalization Configuration

The portal is fully configurable for internationalization, however users should have a general knowledge of Internationalization in Java products before attempting these configurations. Oracle hosts a comprehensive guide to internationalizing Java products at http://docs.oracle.com/javase/tutorial/i18n/TOC.html.

All portal applications contain property files for various languages. They are packaged with the portlets applications in a **`WEB-INF/classes/locale/`** directory.

These files are located in the **`classes`** folder of the **`WEB-INF`** directory, so as to be loaded by the **`ClassLoader`**.

All resource files are in a sub-folder named **locale**.

For instance; the translations for the *NavigationPortlet* are located in **web.war/WEB-INF/classes/locale/portlet/portal**

```
 NavigationPortlet_de.properties
NavigationPortlet_en.properties
NavigationPortlet_es.properties
NavigationPortlet_fr.properties
NavigationPortlet_nl.properties
NavigationPortlet_ru.properties
NavigationPortlet_uk.properties
NavigationPortlet_ar.xml
```

Inside those file are typical **key=value** Java EE properties. For example; in the **Spanish** file:

```
javax.portlet.title=Portlet de navegaci\u00f3n
```

There are also properties files in the portal itself. They form the **portal resource bundle**.

From a portlet you can then access translations from the portlet itself or shared at the portal level, both are aggregated when you need them.

Note

It is also possible to use a proprietary XML format to define translations. This is a more convenient way to translate a document for some languages such as Japanese, Arabic or Russian.

Property files must be ISO 8859-1 encoded, while the XML file can define its encoding. As a result it is easier for a person to read a translation in XML instead of having to decode and encode the property file.

For more information see *XML Resources Bundles* in the Development Guide.

Report a bug

25.5.1. Configuring Locales

Various languages are available in the portal package. The configuration below will define which languages are shown in the "**Change Language**" section and made available to users.

The **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/common/common-configuration.xml** file of your installation contains the following section:

```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/default149.xml" parse="text"/>
```

This configuration points to the locale configuration file.

The locale configuration file (**portal.war:/WEB-INF/conf/common/locales-config.xml**) contains the following code:

Example 25.4. The locales-config.xml File Explanation

```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
href="extras/default150.xml" parse="text"/>
```

locale: The locale has to be defined as per the codes defined here. In this example **ar** is Arabic.

output-encoding: deals with character encoding. It is recommended that UTF-8 be used.

input-encoding: In the Java implementation, the encoding parameters will be used for the request response stream. The input-encoding parameter will be used for request **setCharacterEncoding(..)**.

description: A description of the language.

orientation: Although the default orientation of text and images is Left-To-Right, the portal supports Right-To-Left orientation. Modifying text orientation is explained in *Right To Left RTL Framework* in Development Guide.

Report a bug

25.5.2. Configuring Resource Bundle Service

The resource bundle service is configured in: **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/conf/common/common-configuration.xml**:

Example 25.5. The common-configuration.xml file explained

```
<component>
  <key>org.exoplatform.services.resources.ResourceBundleService</key>

<type>org.exoplatform.services.resources.impl.SimpleResourceBundleServi
ce</type>
  <init-params>
  <!-- Comment #1 -->
    <values-param>
      <name>classpath.resources</name>
      <description>The resources that start with the following package
name should be loaded from the file system</description>
      <value>locale.portlet</value>
    </values-param>
    <!-- Comment #2 -->
    <values-param>
      <name>init.resources</name>
      <description>Initiate the following resources during the first
launch</description>
      <value>locale.portal.expression</value>
      <value>locale.portal.services</value>
      <value>locale.portal.webui</value>
      <value>locale.portal.custom</value>
```

```
      <value>locale.navigation.portal.classic</value>
      <value>locale.navigation.group.platform.administrators</value>
      <value>locale.navigation.group.platform.users</value>
      <value>locale.navigation.group.platform.guests</value>

<value>locale.navigation.group.organization.management.executive-
board</value>
    </values-param>
    <!-- Comment #3 -->
    <values-param>
      <name>portal.resource.names</name>
      <description>The properties files of  the portal ,  those file
will be merged
        into one ResoruceBundle properties </description>
      <value>locale.portal.expression</value>
      <value>locale.portal.services</value>
      <value>locale.portal.webui</value>
      <value>locale.portal.custom</value>
    </values-param>
  </init-params>
</component>
```

Comment #1

Resources whose package name starts with the name specified in the *classpath.resources* parameter are loaded from the file system. Detailed information can be found in a later section of this chapter.

Comment #2

Resources related to portal, group, and user reference bundles

Comment #3

The *portal.resource.names* parameter defines all resources that belong to the *Portal Resource Bundle*. These resources are merged to a single resource bundle which is accessible from anywhere in the portal. All these keys are located in the same bundle, which is separated from the navigation resource bundles.

Report a bug

25.5.3. Configuring Navigation Resource Bundles

There is a resource bundle for each navigation. A navigation can exist for user, groups, and portal.

Section 25.5.2, “Configuring Resource Bundle Service” shows bundle definitions for the navigation of the classic portal and of four different groups. Each of these resource bundles occupies a different sphere, they are independent of each other and they are not included in the ***portal.resource.names*** parameter.

The properties for a group must be in the **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/locale/navigation/group/** folder. For example, **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/locale/navigation/group/organization/management/executive-board_en.properties**.

The folder and file names must correspond to the group hierarchy. The group name "***executive-board***" is followed by the ISO 639 code.

Each language defined in ***LocalesConfig*** must have a resource file defined. If the name of a group is changed the name of the folder and/or files of the correspondent navigation resource bundles must also be changed.

Content of **executive-board_en.properties**:

```
organization.title=Organization
organization.newstaff=New Staff
organization.management=Management
```

This resource bundle is only accessible for the navigation of the ***organization.management.executive-board*** group.

Report a bug

25.6. Portlets

25.6.1. Configuring Portlets

Portlets are independent applications and deliver their own resource files. All shipped portlet resources are located in the **locale/portlet** sub-folder. The ResourceBundleService parameter **classpath.resources** defines this sub-folder.

Procedure 25.2. Add Spanish Translation to the GadgetPortlet

1. Create the file **GadgetPortlet_es.properties** in **JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/locale/portlet/gadget/GadgetPortlet**.
2. In **portlet.xml**, add ***Spanish*** as a **supported-locale** ('**es**' is the two letter code for Spanish). The **resource-bundle** is already declared and is the same for all languages :

```
<supported-locale>en</supported-locale>
<supported-locale>es</supported-locale>
<resource-bundle>locale.portlet.gadget.GadgetPortlet</resource-bundle>
```

See the portlet specification for more details about portlet internationalization.

Report a bug

25.6.2. Standard Portlet Resource Keys

The portlet specification defines three standard keys: *Title*, *Short Title* and *Keywords*. Keywords is formatted as a comma-separated list of tags.

```
javax.portlet.title=Breadcrumbs Portlet
 javax.portlet.short.title=Breadcrumbs
 javax.portlet.keywords=Breadcrumbs, Breadcrumb
```

Report a bug

25.6.3. Debugging Resource Bundle Usage

When translating an application it can sometimes be difficult to find the right key for a given property. You can start the Portal in *debug mode* and use the **Magic locale** from the list of available portal languages to assist in finding the correct translated key value. This special locale translates a key to the same value. For example, the translated value for the "organization.title" key is the value "organization.title". Selecting Magic locale allows use of the portal and its applications with all the keys visible. This makes it easier to discover the correct key for a given label in the portal page.

Procedure 25.3. Accessing the Magic Locale

1. Start the portal in debug mode by executing the following command-line argument:

```
[<replaceable>USER</replaceable>@<replaceable>HOST</replaceable>
jboss-jpp-6.1;]$ ./bin/standalone.sh -Dexo.product.developing=true
```

2. Open http://localhost:8080/portal/classic to display the Portal Platform landing page.
3. Click **Change Language**.
4. Select **ma** from the list of available languages to activate the Magic locale.

Report a bug

25.6.4. Translating the Language Selection Form

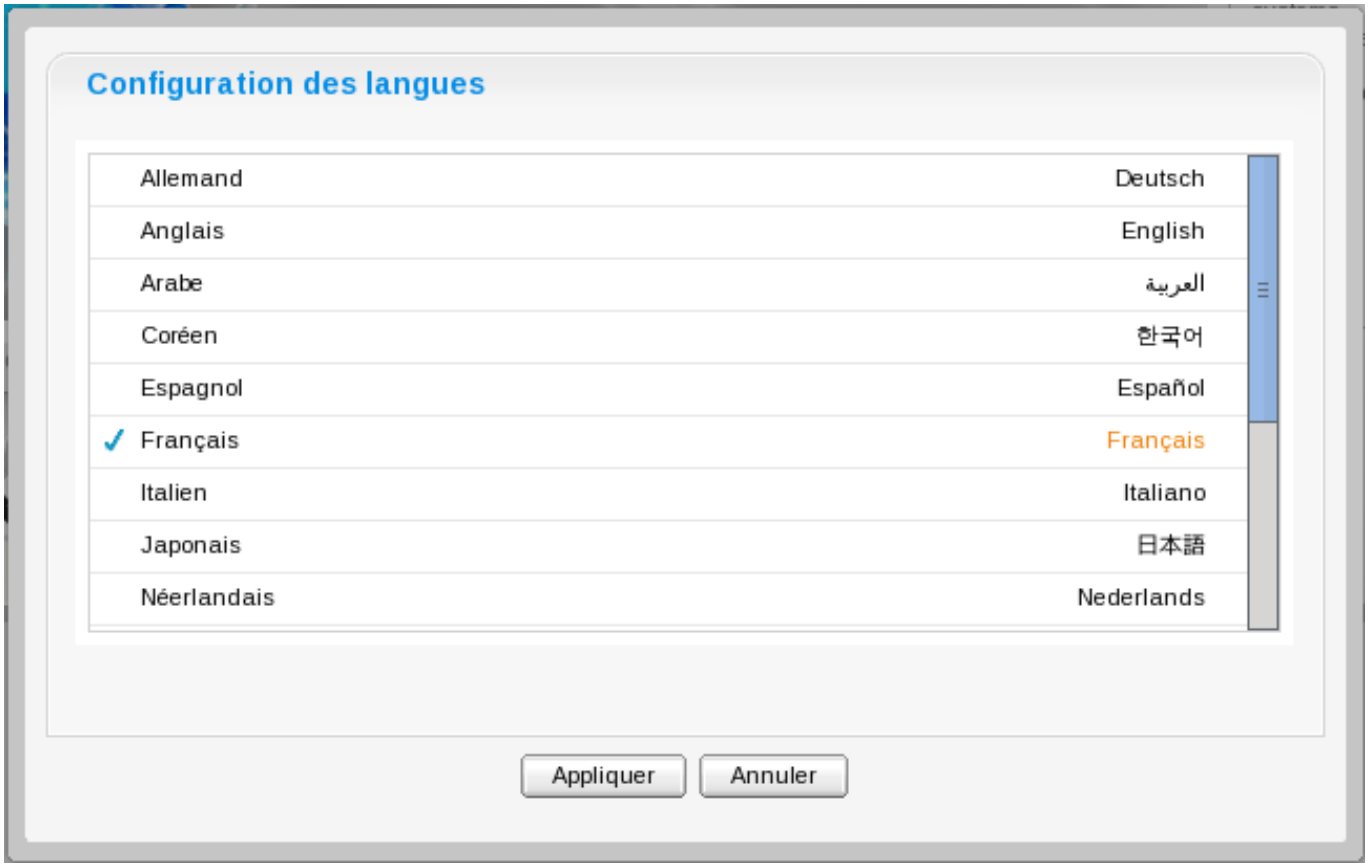


Figure 25.1. Language Selection Screen

When choosing a language from the Language Select screen, the user is presented with a list of languages on the left side in the current chosen language. On the right side, the same language is translated into its own language.

The local language values are obtained from the JDK API **`java.util.Locale.getDisplayedLanguage()`** and **`java.util.Locale.getDisplayedCountry()`** (if needed). Not all languages may be translated, and the languages available can also depend on the JVM currently used.

It is possible to override these values by editing the **`locale.portal.webui`** resource bundle.

Report a bug

25.6.5. Overriding Default JDK API Language Values

Procedure 25.4. Overriding Default JDK API Language Values

1. Edit the **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/locale/portal/webui_xx_yy.properties`** file where *xx_yy* represents the country code of the language you wish to translate.
2. In that file, add or modify a key such as **`Locale.xx_yy`** with the *xx_yy* value being the translated string.
3. Edit **`JPP_HOME/gatein/gatein.ear/portal.war/WEB-INF/classes/locale/portal/webui_fr.properties`** where fr is the country code for French, and add the following key into it:

```
Locale.zh_TW=Chinois traditionnel
```

When the portal is next restarted, the localized language will be updated in the user interface when a user tries to change the current language.

Report a bug

Chapter 26. Localization Configuration

26.1. Pluggable Locale policy

Every request processed by every portlet is invoked within a context of the current **Locale**. The current **Locale** can be retrieved by calling the **getLocale()** method of **javax.portlet.PortletRequest** interface. The exact algorithm for determining the current **Locale** is not specified by Portlet Specification. Portlet containers implement this the way they deem most appropriate. Each portal instance has a default language which can be used to present content for new users.

Another option is to use each user's browser language preference, provided it matches one of the available localizations that the portal supports, and only fallback to the portal's default language if no match is found. Every user, while visiting a portal, has an option to change the language of the interface by using a Language chooser. The choice can be remembered for the duration of the session, or it can be remembered for a longer period using a browser cookie. For registered and logged-in users, it can be saved into the user's profile.

As there is more than one way to determine the **Locale** to be used for displaying a portal page, the mechanism for determining the current **Locale** of the request is pluggable in the portal, and the exact algorithm can be customized.

Report a bug

26.1.1. Locale Policy API

Customization is achieved by using LocalePolicy API, which is a simple API consisting of one interface, and one class:

- **org.exoplatform.services.resources.LocalePolicy** interface
- **org.exoplatform.services.resources.LocaleContextInfo** class

The **LocalePolicy** interface defines a single method that is invoked on the installed **LocalePolicy** service implementation:

```
public interface LocalePolicy
   {
      public Locale determineLocale(LocaleContextInfo localeContext);
   }
```

The **Locale** returned by determineLocale() method is the **Locale** that will be returned to portlets when they call **javax.portlet.PortletRequest.getLocale()** method.

The returned **Locale** has to be one of the locales supported by portal, otherwise it will fall back to the portal default **Locale**.

The supported locales are listed in ***JPP_HOME*/gatein/gatein.ear/portal.war/WEB-INF/conf/common/locales-config.xml** file.

The **determineLocale()** method takes a parameter of type **LocaleContextInfo**, which represents a compilation of preferred locales from different sources; user's profile, portal default, browser language settings, current session, browser cookie.

All these different sources of **Locale** configuration or preference are used as input to **LocalePolicy** implementation that decides which **Locale** should be used.

Report a bug

26.1.2. Default Locale Policy

By default, **org.exoplatform.portal.application.localization.DefaultLocalePolicyService**, an implementation of **LocalePolicy**, is installed to provide the default behavior. This, however, can easily be extended and overridden. A completely new implementation can also be written from scratch.

DefaultLocalePolicyService treats logged-in users slightly differently than anonymous users. Logged-in users have a profile that can contain language preference, while anonymous users do not.

Here is an algorithm used for anonymous users.

Procedure 26.1. An algorithm for anonymous users

1. Iterate over **LocaleContextInfo** properties in the following order:
   - **cookieLocales**
   - **sessionLocale**
   - **browserLocales**
   - **portalLocale**
2. Get each property's value. If it's a collection, get the first value.
3. If value is one of the supported locales return it as a result.
4. If the value is not in the supported locales set, try to remove country information and verify whether a language matching the locale is in the list of supported locales. If so, return it as a result.
5. Otherwise, continue with the next property.

If no supported locale is found the return locale eventually defaults to **portalLocale**.

The algorithm for authenticated users is virtually the same except that the first **Locale** source checked is user's profile.

Procedure 26.2. An algorithm for authenticated users

1. Iterate over **LocaleContextInfo** properties in the following order:
   - **userProfile**
   - **cookieLocales**
   - **sessionLocale**
   - **browserLocales**
   - **portalLocale**
2. Perform the rest of the steps in An algorithm for anonymous users.

Report a bug

26.1.3. Customize LocalePolicy

The easiest way to customize the **LocalePolicy** is to extend **DefaultLocalePolicyService**. A study of the source code is required. JavaDocs provide thorough information on this.

Most customizations will involve simply overriding one or more of its protected methods.

An example of a customization is an already provided **NoBrowserLocalePolicyService**. By overriding just one method, it skips any use of browser language preference.

```
public class NoBrowserLocalePolicyService extends
DefaultLocalePolicyService
   {
      /**
       * Override super method with no-op.
       *
       * @param context locale context info available to implementations
in order to determine appropriate Locale
       * @return null
       */
      @Override
      protected Locale getLocaleConfigFromBrowser(LocaleContextInfo
context)
      {
         return null;
      }
   }
```

Report a bug

26.1.4. Configuring Locale Policy

The **LocalePolicy** framework is enabled for portlets by configuring **LocalizationLifecycle** class in portal's webui configuration file: ***JPP_HOME*/gatein/gatein.ear/portal.war/WEB-INF/webui-configuration.xml**:

```
<application-life-cycle-listeners>
   ...

<listener>org.exoplatform.portal.application.localization.LocalizationLif
ecycle</listener>
</application-life-cycle-listeners>
```

The default **LocalePolicy** implementation is installed as an eXo Kernel portal service via ***JPP_HOME*/gatein/gatein.ear/portal.war/WEB-INF/conf/portal/web-configuration.xml**.

The following excerpt is responsible for installing the service:

```
<component>
    <key>org.exoplatform.services.resources.LocalePolicy</key>

<type>org.exoplatform.portal.application.localization.DefaultLocalePolic
yService</type>
</component>
```

Besides implementing **`LocalePolicy`**, the service class also needs to implement **`org.picocontainer.Startable`** interface in order to get installed.

Report a bug

26.2. Bridged and Nonbridged Resources

All the resources in portals that are not portlets themselves, but are accessed through portlets - reading data through **`PortletRequest`**, and writing to **`PortletResponse`** - are referred to as 'bridged'. Any resources that are accessed directly, bypassing portal filters and servlets, are referred to as *non-bridged*.

Non-bridged servlets, and .jsps have no access to **`PortalRequest`**. They do not use **`PortletRequest.getLocale()`** to determine current **`Locale`**. Instead, they use **`ServletRequest.getLocale()`** which is subject to precise semantics defined by Servlet specification - it reflects browser's language preference.

In other words, non-bridged resources do not have a notion of current **`Locale`** in the same sense that portlets do. The result is that when mixing portlets and non-bridged resources there may be a localization mismatch, an inconsistency in the language used by different resources composing your portal page.

This problem is addressed by **`LocalizationFilter`**. This is a filter that changes the behavior of **`ServletRequest.getLocale()`** method so that it behaves the same way as **`PortletRequest.getLocale()`**.

That way even localization of servlets, and .jsps accessed in a non-bridged manner can stay in sync with portlet localization.

Report a bug

26.2.1. Installing LocalizationFilter

All the resources in portals that are not portlets themselves, but are accessed through portlets - reading data through **`PortletRequest`**, and writing to **`PortletResponse`** - are referred to as 'bridged'. Any resources that are accessed directly, bypassing portal filters and servlets, are referred to as *non-bridged*.

Non-bridged servlets, and .jsps have no access to **`PortalRequest`**. They do not use **`PortletRequest.getLocale()`** to determine current **`Locale`**. Instead, they use **`ServletRequest.getLocale()`** which is subject to precise semantics defined by Servlet specification - it reflects browser's language preference.

In other words, non-bridged resources do not have a notion of current **`Locale`** in the same sense that portlets do. The result is that when mixing portlets and non-bridged resources there may be a localization mismatch, an inconsistency in the language used by different resources composing your portal page.

This problem is addressed by **`LocalizationFilter`**. This is a filter that changes the behavior of **`ServletRequest.getLocale()`** method so that it behaves the same way as **`PortletRequest.getLocale()`**.

That way even localization of servlets, and .jsps accessed in a non-bridged manner can stay in sync with portlet localization.

Report a bug

Part VII. Gadget Configuration

Chapter 27. Gadget Importer Tool

The gadget importer allows you to add gadgets to the application registry. The tool uses a WebApp folder which contains the gadgets configuration. WebApp has a standard structure that is processed by the Gadget importer.

The WebApp folder consist of **web.xml** and **gadget.xml** configuration files.

Local gadget resources

For local gadgets the source and resources must be located in the same folder. The Gadget importer tool searches for the folder containing the gadget source followed by searching the resources in the folder. The Gadget importer tool stores the gadget resources in JCR. So, placing the gadget resources in a different folder other than the gadget source can cause many unwanted files to be stored as the resource files of the gadget.

Report a bug

27.1. Importing Gadgets

27.1.1. Importing Gadgets

This section explains the structure and configuration of a WebApp. WebApp is processed by the Gadget importer tool.

For more information on using user interface to import gadgets, see *User guide*.

Report a bug

27.1.2. Creating a Standard WebApp Folder to Import Gadgets

Red Hat JBoss Portal allows you to create a WebApp folder to contain the configuration of the list of gadgets which are automatically added to the Application Registry.

This section explains the structure and configuration of a WebApp. WebApp is processed by the Gadget importer.

Report a bug

27.1.3. Configuring the WebApp Files

The WebApp structure consists of **web.xml** and **gadget.xml** files.

To register WebApp to RedHat JBoss Portal, add the parameter **org.gatein.wci.api.GateInServlet** in **web.xml** file.

```
<web-app>
  <display-name>eXoGadgets</display-name>

  <servlet>
    <servlet-name>GateInServlet</servlet-name>
    <servlet-class>org.gatein.wci.api.GateInServlet</servlet-class>
```

```
    <load-on-startup>0</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>GateInServlet</servlet-name>
    <url-pattern>/gateinservlet</url-pattern>
  </servlet-mapping>

</web-app>
```

The **gadget.xml** file locates the configuration of gadgets that are added to the Application Registry.

```
 <gadgets
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_objects_1_0
http://www.gatein.org/xml/ns/gadgets_1_0"
    xmlns="http://www.gatein.org/xml/ns/gadgets_1_0">

 <gadget name="To-do">
  <path>/gadgets/Todo/Todo.xml</path>
 </gadget>

 <gadget name="Calendar">
  <path>/gadgets/Calendar/Calendar.xml</path>
 </gadget>

 <gadget name="Calculator">
  <path>/gadgets/Calculator/Calculator.xml</path>
 </gadget>

 <gadget name="rssAggregator">
  <path>/gadgets/rssAggregator/rssAggregator.xml</path>
 </gadget>

  <gadget name="Currency">
    <url>http://www.donalobrien.net/apps/google/currency.xml</url>
  </gadget>

  <gadget name="ServicesManagement">
    <path>/gadgets/ServicesManagement/ServicesManagement.xml</path>
  </gadget>
</gadgets>
```

The **gadget.xml** has three important tags, gadget,path, and url.

- **<gadget>**: A gadget is configured in the **<gadget>** tag. The **name** attribute is the gadget name. The gadget name is managed by the Application Registry.
- **<path>**: This tag is the child of the **<gadget>** tag. It contains the path to the gadget and helps the Gadget service to find the location of the gadget source and resources. It is known as the local path of the server.
- **<url>**: This tag is configured to indicate the URL of the gadget incase the source of gadget resides on an external server.

Note

The Gadget importer searches for a folder that contains the gadget source, then finds resources in the folder and store them into JCR as the resources of the gadget. Therefore, putting the gadget resources in the folder different from the gadget source can cause unnecessary files to be stored as the resource files of the gadget.

Report a bug

27.2. Virtual Servers for Gadgets

27.2.1. Setting up Virtual Servers for Gadget Rendering

Red Hat JBoss Portal recommends using two virtual hosts for security. If the gadget is running on a different domain than the container (the website that contains the application), it is unable to interfere with the portal by modifying code or cookies.

An example, where the portal is hosted on `http://www.sample.com` and the gadgets on `http://www.samplemodules.com` .

To configure virtual server for gadgets, set the **`gadgets.hostName`** parameter **`value`** to **`path/to/gadgetServer`** in **`GadgetRegisteryService`**.

Example 27.1. Setting the value variable for the gadget hostname

```
<component>
  <key>org.exoplatform.application.gadget.GadgetRegistryService</key>

<type>org.exoplatform.application.gadget.jcr.GadgetRegistryServiceImpl
</type>
  <init-params>
    <value-param>
      <name>gadgets.hostName</name>
      <description>Gadget server url</description>
      <value>http://localhost:8080/GateInGadgetServer/gadgets/</value>
    </value-param>
  </init-params>
</component>
```

Portal supports multiple rendering servers to balance the load across multiple servers.

Note

When deploying on the same server, the gadget must be initiated before handling any calls requests. For example, if the webapp **`GateInGadgets`** uses **`org.exoplatform.application.gadget.GadgetRegister`** so the webapp must be initiated before it is called .

Report a bug

27.2.2. Configuring Gadget Server

The gadget container uses three security files for authenticating and authorizing gadgets.

- **key.txt**
- **oauthkey.pem**
- **oauthkey_pub.pem**

The location of the security keys for different servers are listed below.

- JBoss: **$JBOSS_HOME/standalone/configuration/gatein/gadgets** . This folder is configured by system variables in **$JBOSS_HOME/standalone/configuration/gatein/configuration.properties** .

  Example 27.2. Configuring Security key paths

  ```
  gatein.gadgets.securitytokenkeyfile=$\
  {gatein.conf.dir\}/gadgets/key.txt
  gatein.gadgets.signingkeyfile=$\
  {gatein.conf.dir\}/gadgets/oauthkey.pem
  ```

  If you have other files update the path variables to point to your files.

The **key.txt** file consist of a secret key, which is used to encrypt the security token for user authentication. At portal startup, this file is read through the **gatein.gadgets.securitytokenkeyfile** path.

If the **key.txt** file is not found, JBoss Portal generates a new **key.txt**and stores it in the **gatein.gadgets.securitytokenkeyfile** path.

The **oauthkey.pem** and **oauthkey_pub.pem** are a key pair of RSA cryptography standard. **oauthkey.pem** is the private key and **oauthkey_pub.pem** is the public key. These keys are the default keys of the gadget container which is used by OAuth gadget to authorize with external service providers.

Report a bug

27.2.3. Configuring Gadget Proxy and Concat

The proxy and concat server must be on the same domain as the gadget server. You can configure the container in using **eXoGadgetServer:/WEB-INF/classes/containers/default/container.js** .

Example 27.3. Setting proxy and concat server url

```
"gadgets.content-rewrite" : {
  "include-urls": ".*",
  "exclude-urls": "",
  "include-tags": ["link", "script", "embed", "img", "style"],
  "expires": "86400",
  "proxy-url": "http://localhost:8080/eXoGadgetServer/gadgets/proxy?
```

```
url=",
  "concat-url":
"http://localhost:8080/eXoGadgetServer/gadgets/concat?"
},
```

Add the following code to the beginning of the JVM to allow external gadgets to obtain a connection to a server behind a firewall.

```
-Dhttp.proxyHost=proxyhostURL -Dhttp.proxyPort=proxyPortNumber -
Dhttp.proxyUser=someUserName -Dhttp.proxyPassword=somePassword
```

Report a bug

27.3. Shindig Server

27.3.1. Configuring Shindig Container

Red Hat JBoss Portal uses a custom Shindig OpenSocial container for handling gadgets. The Shindig configuration file is located at: **`$JBOSS_HOME/JPP_HOME/standalone/configuration/gatein/gadgets/shindig.properties`**

Note

For Shindig configuration options, see the Apache Shindig documentation.

Report a bug

27.3.2. Configuring Shindig Container for Offline access

The default features available in the current version of shindig require online access. If the server starts without internet access, there is a minor delay of usually 1 sec at startup when the server tries to access third party services.

Due to specific server settings, the delay at startup can be longer. If you are using Red Hat JBoss Portal offline and the gadget server has long time outs during startup, you can configure the shindig server to exclude online features.

To configure shindig server in offline mode you need to modify the shindig configuration file located at, **`$JBOSS_HOME/standalone/configuration/gatein/gadgets/shindig.properties`**

The location of feature manifest is changed from online to offline by excluding the **`online-features.txt`** file reference.

Example 27.4. Online Mode setting

```
# Location of feature manifests (comma separated)
shindig.features.default=res://features/default-
features.txt,res://features/online-features.txt
```

Example 27.5. Offline Mode setting

```
# Location of feature manifests (comma separated)
shindig.features.default=res://features/default-features.txt
```

Important

In offline mode the analytics is not available for the gadgets.

Report a bug

Part VIII. Web Services for Remote Portlets

Chapter 28. Web Services for Remote Portlets

The Web Services for Remote Portlets specification defines a web service interface for accessing and interacting with interactive presentation-oriented web services. It has been produced through the efforts of the Web Services for Remote Portlets (WSRP) OASIS Technical Committee. It is based on the requirements gathered and on the concrete proposals made to the committee.

Scenarios that motivate WSRP functionality include:

- Content hosts, such as portal servers, providing Portlets as presentation-oriented web services that can be used by aggregation engines.
- Aggregating frameworks, including portal servers, consuming presentation-oriented web services offered by content providers and integrating them into the framework.

More information on WSRP can be found on the official website for WSRP. Further suggested reading is the primer for a good, albeit technical, overview of WSRP.

Report a bug

28.1. WSRP Support

The WSRP Technical Committee defined WSRP Use Profiles to help with WSRP interoperability. See http://www.oasis-open.org/committees/download.php/3073 for more information. This section references terms defined in this document.

The portal provides a Simple level of support for the WSRP Producer except that out-of-band registration is not currently handled. In-band registration and persistent local state (which are defined at the Complex level) are supported.

On the Consumer side, the portal provides a Medium level of support for WSRP, except that the consumer only handles HTML markup (because the portal itself does not handle other markup types). The portal does support explicit portlet cloning and it fully supports the PortletManagement interface.

As far as caching goes, the component has a Level 1 Producer and Consumer. Cookie handling is supported properly on the Consumer and the Producer requires initialization of cookies (it has been found that it improves interoperability with some consumers). The component does not support custom window states or modes, because the portal does not. The component does, however, support CSS on both the Producer (though it is more a function of the portlets than inherent Producer capability) and Consumer.

While a complete implementation of WSRP 1.0 is provided, the community developers do need to review the Conformance statements declared at http://www.oasis-open.org/committees/download.php/6018, and perform more interoperability testing.

The portal supports WSRP 2.0 with a complete implementation of the mandatory features with the exception of support for lifetimes, and leasing support.

Note

As of version 6.1 of the portal, WSRP is only activated and supported using the portal deployed on Red Hat JBoss Enterprise Application Platform 6.

Report a bug

28.2. Deploying Services

28.2.1. Deploying Web Services for Remote Portlets services

The portal provides complete support for WSRP 1.0 and 2.0 standard interfaces, and offers both consumer and producer services. Starting with version 2.1.0-GA of the component, WSRP is packaged as a portal extension, and is contained in a package named ***JPP_HOME*/gatein/extensions/gatein-wsrp-integration.ear** .

The only files of interest from a user perspective are located in the ***JPP_HOME*/standalone/configuration/gatein/wsrp** directory.

- **gatein-wsse-consumer.xml**, which allows you to configure WS-Security support for the consumer.
- **gatein-wsse-producer.xml** which allows you to configure WS-Security support for the producer.

The extension itself is composed of the following components:

- **META-INF** contains files necessary for EAR packaging.
- The **extension-component-6.1.0.jar**, which contains the components needed to integrate the WSRP component into the portal. It also includes the default configuration files for the WSRP producer and the default WSRP consumers.
- The **extension-config-6.1.0.jar**, which contains the configuration file needed by the GateIn extension mechanism to properly register this EAR as an extension.
- The **extension-war-6.1.0.war**, which contains the configuration files needed by the portal extension mechanism to setup the WSRP service. It includes **wsrp-configuration.xml** which, configures options for the **WSRPServiceIntegration** component of the WSRP integration in the portal.

  The file **$RHJP_HOME/gatein/extensions/gatein-wsrp-integration.ear/extension-war.war/WEB-INF/conf/wsrp/wsrp-configuration.xml** contains the location of file **wsrp-consumers-config.xml** defined under the configuration key **consumersConfigLocation**.
- The **lib** directory, which contains the different libraries needed by the WSRP service.
- The **wsrp-admin-gui-2.2.2.Final.war**, which contains the WSRP Configuration portlet with which you can configure consumers to access remote servers and how the WSRP producer is configured.
- The **wsrp-producer-jb5wsss-2.2.2.Final.war**, which contains the producer-side support for WS-Security.

If WSRP is not required, it will not adversely affect your installation to leave it as is. Otherwise, remove the **gatein-wsrp-integration.ear** file from your AS deploy directory.

See Also:

- Section 30.1, “About Web Services Security Configuration”

Report a bug

28.2.2. Considerations to use WSRP

The web service stack that the portal uses is based on JBoss WS. It updates the port and host name used in WSDL. For more information, see the *Web Services* chapter in the Red Hat JBoss Enterprise Application Platform 6 *Administration and Configuration User Guide*.

If you have modified the host name and port on which your server runs, you will need to update the configuration for the consumer used to consume the platform's producer.

Report a bug

28.3. Remote Portlets

28.3.1. Making a Remote Portlet

The portal does not expose local portlets for consumption by remote WSRP consumers by default. In order to make a portlet remotely available, it must be made *remotable* by marking it as such in the associated **`portlet.xml`**. This is accomplished by using a specific **`org.gatein.pc.remotable`** container-runtime-option.

Setting its value to **`true`** makes the portlet available for remote consumption, while setting its value to **`false`** will not publish it remotely. As specifying the remotable status for a portlet is optional, you do not need to do anything if you do not need your portlet to be available remotely.

Important

Only JSR-286 (Portlet 2.0) portlets can be made remotable as the mechanism to expose a portlet to WSRP relies on a JSR-286-only functionality.

Report a bug

28.3.2. Making a Single Remote Portlet

Example 28.1. Single Portlet Remotable Example

The "BasicPortlet" portlet is specified as being remotable.

```
<?xml version="1.0" standalone="yes"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-
app_2_0.xsd"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-
app_2_0.xsd http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
             version="2.0">
<portlet-app>
   <portlet>
      <portlet-name>BasicPortlet</portlet-name>

      ...

      <container-runtime-option>
```

```
          <name>org.gatein.pc.remotable</name>
          <value>true</value>
       </container-runtime-option>
    </portlet>
 </portlet-app>
```

It is also possible to specify that all the portlets declared within a given portlet application to be remotable by default. This is done by specifying the **container-runtime-option** at the **portlet-app** element level. Individual portlets can override that value to not be remotely exposed.

Report a bug

28.3.3. Making Multiple Remote Portlets

The example defines two portlets. The **org.gatein.pc.remotable container-runtime-option** being set to **true** at the **portlet-app** level, all portlets defined in this particular portlet application are exposed remotely by the WSRP producer.

```
<?xml version="1.0" standalone="yes"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-
app_2_0.xsd"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-
app_2_0.xsd http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
             version="2.0">
<portlet-app>

   <portlet>
      <portlet-name>RemotelyExposedPortlet</portlet-name>
      ...
   </portlet>
   <portlet>
      <portlet-name>NotRemotelyExposedPortlet</portlet-name>
      ...
      <container-runtime-option>
         <name>org.gatein.pc.remotable</name>
         <value>false</value>
      </container-runtime-option>
   </portlet>

   <container-runtime-option>
      <name>org.gatein.pc.remotable</name>
      <value>true</value>
   </container-runtime-option>
</portlet-app>
```

It is possible to override the default behavior by specifying a value for the **org.gatein.pc.remotable container-runtime-option** at the **portlet** level.

The **RemotelyExposedPortlet** inherits the remotable status defined at the **portlet-app** level since it does not specify a value for the **org.gatein.pc.remotable container-runtime-option**. The **NotRemotelyExposedPortlet**, however, overrides the default behavior and is not remotely exposed. Note that in the absence of a top-level **org.gatein.pc.remotable container-runtime-option** value set to **true**, portlets are not remotely exposed.

Report a bug

28.3.4. Make portlets aware of WSRP requests

A portlet can check if a request is coming from WSRP by referring to the WSRP attributes. WSRP sets the **`org.gatein.invocation.fromWSRP`** (defined by the **`org.gatein.wsrp.WSRPConstants.FROM_WSRP_ATTRIBUTE_NAME`** constant) request attribute to **`Boolean.TRUE`** when the request is initiated from WSRP.

Example 28.2. Checking if a request is issued from WSRP inside a portlet

```
if
(!Boolean.TRUE.equals(request.getAttribute(WSRPConstants.FROM_WSRP_ATTR
IBUTE_NAME)))
{
  // do something
}
```

Report a bug

28.3.5. Using WSRP Portlets from a Remote Consumer

WSRP Producers vary greatly from a configuration perspective. Most of them require that you specify the URL for the Producer's WSDL definition. See the remote producer's documentation for specific instructions. For instructions on how to configure the WSRP producer in the portal, see the Consumer Configuration.

The portal's Producer is automatically set up when you deploy a portal instance with the WSRP service. You can access the WSDL file at **`http://{hostname}:{port}/wsrp-producer/v2/MarkupService?wsdl`**. To use only the WSRP 1 compliant version of the producer, use the WSDL file found at **`http://{hostname}:{port}/wsrp-producer/v1/MarkupService?wsdl`**. The default hostname is **`localhost`** and the default port is 8080.

Report a bug

Chapter 29. Securing Web Services for Remote Portlets

There are two main ways to secure the communication between a producer and consumer:

1. Securing the Transport Layer This requires using SSL and a HTTPS endpoint. By using this, the communication between the consumer and producer will be encrypted.

2. Securing the Contents of the SOAP message This option requires using ws-security to handle parts of the SOAP message. With this option you can specify things like encryption, signing, timestamps, etc as well as passing across user credentials to perform a login on the producer side. WS-Security is more powerful and has more options, but is requires more complex configurations.

Depending on requirements, an HTTPs endpoint and/or ws-security can be used.

Report a bug

29.1. Web Services for Remote Portlets over SSL with HTTP endpoints

It is possible to use WSRP over SSL for a secure exchange of data. The portal does not come initially configured for HTTPS connectors, therefore the producer's must be configured for the server. This is a global configuration change, and will affect more than the portal and WSRP. See the Red Hat JBoss Enterprise Application Platform 6 *Administration and Configuration Guide* for instructions relating to configuring HTTPS connectors for the server.

Once the producer is configured for HTTPS connections, modify the URL for the WSRP endpoint on the consumer to point to the new HTTPS based URL. This will require either manually updating the value in the WSRP administration application, or by specifying it using the **`wsrp-consumers-config.xml`** configuration file before the server is first started.

Report a bug

29.1.1. Configuration For Enabling SSL With WSRP

The following procedures are provided as an example of configuring HTTPS/SSL with WSRP.

- Section 29.1.2, “Configuring the Producer to Use HTTPS”
- Section 29.1.3, “Configuring the Consumer to Access the WSRP Endpoint over HTTPS”

Warning

These examples are not the best practices for configuring HTTPS with the platform, and does things which should not be used in a production server (such as self-signed certificates). See JBoss Enterprise Application Platform 6 product documentation for detailed, best practice configuration guidelines.

Report a bug

29.1.2. Configuring the Producer to Use HTTPS

Configure the producer's server to use HTTPS. This is handled in the same manner that you would configure any JBoss Enterprise Application server for HTTPS.

1. Generate the keystore for the producer by executing the following command.

```
keytool -genkey -alias tomcat -keyalg RSA -keystore
producerhttps.keystore -dname "cn=localhost" -keypass changeme -
storepass changeme
```

2. Configure the server to add an HTTPS connection. This requires modifying the standalone/configuration/standalone.xml file with the following content in bold:

```
                <subsystem xmlns="urn:jboss:domain:web:1.1" default-
virtual-server="default-host" native="false">

                <connector name="http" protocol="HTTP/1.1"
scheme="http" socket-binding="http"/>

                <connector name="https" protocol="HTTP/1.1"
scheme="https" socket-binding="https" secure="true">

                <ssl certificate-key-
file="/path/to/producerhttps.keystore" password="changeme"/>

                </connector>

                <virtual-server name="default-host" enable-welcome-
root="true">

                <alias name="localhost"/>

                <alias name="example.com"/>

                </virtual-server>

                ...
```

3. Start the server and verify that https://localhost:8443/portal is accessible. Note that since you are using a self-signed certificate that your browser will give a warning that the certificate cannot be trusted.

Note

In this example case we are accessing the portal using 'localhost' hence why we are using "cn=localhost" in the keytool command. If you are using this across another domain, you will need to make the necessary changes.

Report a bug

29.1.3. Configuring the Consumer to Access the WSRP Endpoint over HTTPS

1. Export the producer's public key from the producer's keystore

```
keytool -export -alias tomcat -file producerkey.rsa -keystore
producerhttps.keystore -storepass changeme
```

2. Import the producer's public key into a new keystore for the consumer

```
keytool -import -alias tomcat -file producerkey.rsa -keystore
consumerhttps.keystore -storepass changeme -noprompt
```

3. Configure the bin/standalone.conf file to add the following line at the end of the file:

```
JAVA_OPTS="$JAVA_OPTS -
Djavax.net.ssl.trustStore=/path/to/consumerhttps.keystore -
Djavax.net.ssl.trustStorePassword=changeme"
```

4. Start the consumer and change the selfv2 producer url to https://localhost:8443/wsrp-producer/v2/MarkupService?wsdl and verify that the consumer can access the producer.

Note

It is possible to modify the wsrp-consumers-config.xml configuration file to change the URL instead of modifying it in the administration GUI.

It is possible to use WSRP over SSL for secure exchange of data. Configure your server appropriately as described in the *HTTPS Configuration* section of the *Installation Guide*.

Report a bug

29.2. Web Services for Remote Portlets and Web Services Security

Portlets may present different data or options depending on the currently authenticated user. For remote portlets, this means having to propagate the user credentials from the consumer back to the producer in a safe and secure manner. The WSRP specification does not directly specify how this should be accomplished, but delegates this work to the existing WS-Security standards. The WS-Security standards can also be used to secure the soap message, such as encryption and signing the message.

Encryption is strongly recommended

Encrypt the credentials being sent between the consumer and producer, otherwise they will be sent in plain text and could be easily intercepted. Configure WS-Security to encrypt and sign the SOAP messages being sent, or secure the transport layer by using an HTTPS endpoint. Failure to encrypt the soap message or transport layer will result in the username and password being sent in plain text.

Web Container Compatibility

WSRP and WS-Security is only supported on the portal when running on JBoss Enterprise Application Platform 6.

Report a bug

Chapter 30. Credentials for Web Services Security

When the consumer sends the user credentials to the producer, it is sending the credentials for the currently authenticated user in the consumer. This makes signing in to remote portlets transparent to end users, but also requires that the producer and consumer use the same credentials. This means that the user name and password must be the same and valid on both servers.

The recommended approach for this situation would be to use a common LDAP configuration. See the *LDAP Integration* chapter in the *Administration and Configuration Guide* to correctly configure LDAP on the portal.

Report a bug

30.1. About Web Services Security Configuration

JBoss Enterprise Application Platform 6 uses a different web service implementation than the previous versions: it is now uses the JBossWS CXF Stack instead of the JBossWS Native Stack. Due to these changes, the way we configure WS-Security for WSRP with the portal on JBoss Enterprise Application Platform 6 has changed.

CXF uses interceptors to extend and configure its behavior. There are two main types of interceptors: *inInterceptors* and *outInterceptors*. InInterceptors are invoked for communication coming into the client or server, while outInterceptors are invoked when the client or server sends a message.

So for the WSRP case, the communication from the consumer to the producer is governed by the consumer's OutInterceptor and the producer's InInterceptors. The communication from the producer to the consumer is governed by the producer's OutInterceptor and the consumer's InInterceptor. This may mean having to configure 4 Interceptors.

When dealing with user propagation, only the consumer sends the user credentials to the producer. So Username Tokens only need to be configured for the consumer's OutInterceptor and the producer's InInterceptor.

Consider encrypting the message from the consumer to the producer and also the message from the producer to the consumer. This means that encryption properties must be configured for all 4 interceptors.

See the CXF Documentation for more details on interceptors and their types: http://cxf.apache.org/docs/interceptors.html

To support ws-security, the portal uses CXF's WSS4J Interceptors which handle all ws-security related tasks. See the CXF Documentation for more details: http://cxf.apache.org/docs/ws-security.html

Note

We only support one ws-security configuration option for the producer. All consumers accessing the producer will have to conform to this security constraint. This means if the producer requires encryption, all consumers will be required to encrypt their messages when accessing the producer.

We only support one ws-security configuration option to be used by all the consumers. A consumer has the option to enable or disable ws-security, which allows for one or more consumers to use ws-security while the others do not.

Report a bug

30.2. WSS4J Interceptors and WSRP

The WSS4J Interceptors are configured using property files. WSRP looks for specific property files to know whether or not in/out interceptors must be added and configured for either consumers or producer.

Theses files are located in the **`standalone/configuration/jpp/wsrp/cxf/ws-security`** directory.

Consumer-specific files are in the consumer subdirectory while producer-specific files are located in the producer subdirectory. To add and configure a WSS4J interceptor, add the proper configuration file in the proper directory. If no configuration file is found for a specific interceptor type, then no such interceptor will be added.

“In” interceptors are configured using WSS4JInInterceptor.properties files while “out” interceptors are configured using WSS4JOutInterceptor.properties files.

Table 30.1. Files needed to configure interceptor for WSRP

Side	Interceptor Type	Configuration File
Consumer	IN	**`standalone/configuration/gatein/wsrp/cxf/ws-security/consumer/WSS4JInInterceptor.properties`**
	OUT	**`standalone/configuration/gatein/wsrp/cxf/ws-security/consumer/WSS4JOutInterceptor.properties`**
Producer	IN	**`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/WSS4JInInterceptor.properties`**
	OUT	**`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/WSS4JOutInterceptor.properties`**

See the CXF or WSS4J documentation for instructions and options available for each type of

interceptor.

Report a bug

30.2.1. User Propagation

User propagation can be configured to be used over WSRP with ws-security. What this means is that a user logged into a consumer can have their credentials propagated over to the producer. This allows the producer to authenticate the user and any portlet on the producer (a remote portlet from the consumer's perspective) will view the user as being properly authenticated. This allows for remote portlets to access things like user information.

Note

This only works if the user's credentials on the producer and consumer are the same. This may require using a common authentication mechanism, such as LDAP.

This requires some special options when configuring the producer and server.

Report a bug

30.3. WS-Security Consumer Configuration

In order to configure ws-security on the consumer side, you will have to configure the WSS4J Interceptors as seen above. This will require having to configure the WSS4JInInterceptor and/or WSS4JOutInterceptor. You will also need to check the 'Enable WS-Security' checkbox on the WSRP Admin Portlet for the consumer configuration to take effect.

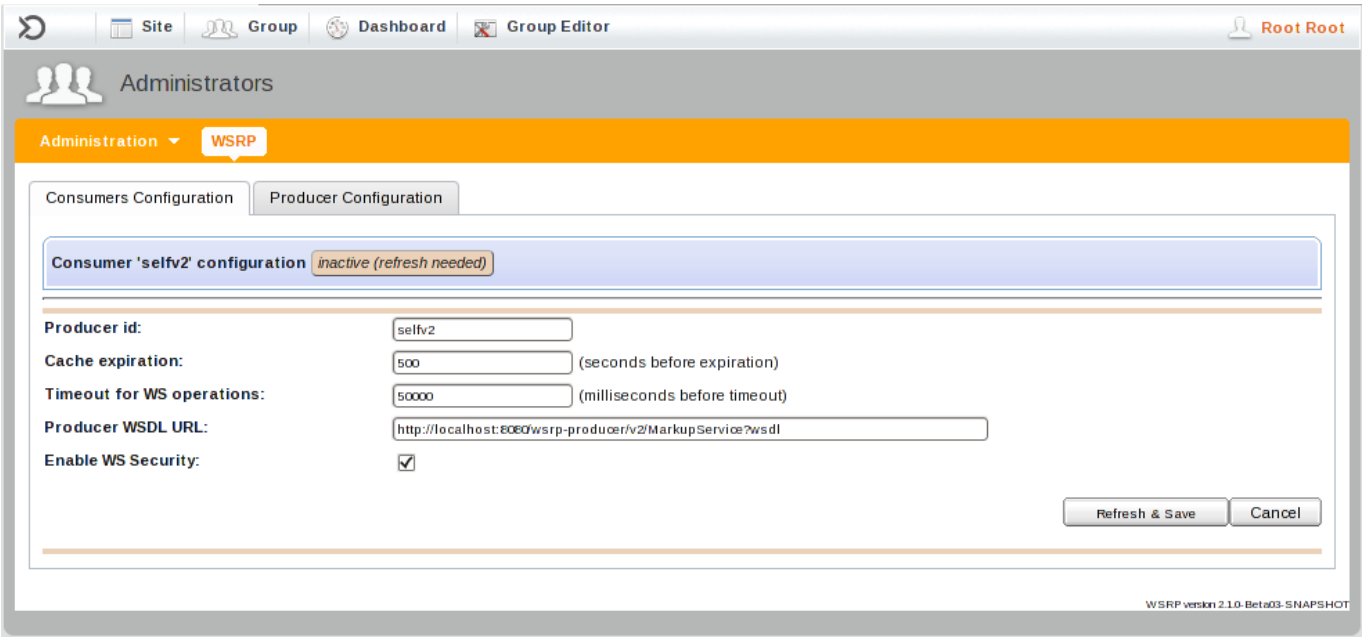


Figure 30.1. WSRP Consumers Configuration

Report a bug

30.3.1. Portal-specific Configuration Options for User Propagation

In order to handle portal user propagation across ws-security, a number of configuration options have been created, which are recommended for the consumer's WSS4JOutInterceptor.

Custom user option

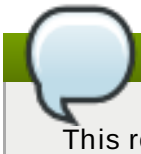
```
user=gtn.current.user
```

This option sets the user property to the currently authenticated user on the consumer.

Custom action option

```
action=gtn.UsernameToken.ifCurrentUserAuthenticated
```

If a user is currently authenticated, it will replace the gtn.UsernameToken.ifCurrentUserAuthenticated with UsernameToken. If the current user is an unauthenticated user, gtn.UsernameToken.ifCurrentUserAuthenticated will be removed from the action list. If no other actions are specified, then the WSS4J interceptor will not be added to the consumer. This allows you to only use ws-security when dealing with authenticated users, and not for anonymous users.



Note

This requires that the user option is set to 'gtn.current.user'

Custom PasswordCallbackClass

```
action=gtn.UsernameToken.ifCurrentUserAuthenticated
```

To set the password for the username token, we need to specify the password in a callback class. See the cxf ws-security documentation for more details http://cxf.apache.org/docs/ws-security.html

A special callback class has already been created which handles this for you: CurrentUserPasswordCallback. This class will retrieve the currently authenticated user's password and set this as the password in the callback object.

```
passwordCallbackClass=org.gatein.wsrp.wss.cxf.consumer.CurrentUser
PasswordCallback
```

Report a bug

30.4. Producer Configuration

The configuration of the producer is similar to that of the consumer. It also requires having to configure the WSS4JInInterceptor and/or WSS4JOutInterceptor.

Report a bug

30.4.1. Special Configuration Options for User Propagation

To correctly propagate user information on the producer-side, use GTNSubjectCreatingInterceptor instead of a regular WSS4JInInterceptor. This portal-specific "in" interceptor is an extension of the traditional WSS4JInInterceptor and therefore can be configured similarly and accept the same configuration properties. To use the GTNSubjectCreatingInterceptor, create a property file at **`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/GTNSubjectCreatingInterceptor.properties`** instead of the regular WSS4JInInterceptor.properties file.

This Interceptor will handle the ws-security headers and retrieve the users credentials. It will then use these credentials to perform a login on the producer site, thus authenticating the user on the producer and makes the user available to remote portlets.

Note

This class also extends org.jboss.wsf.stack.cxf.security.authentication.SubjectCreatingInterceptor and can accept the same properties this class normally accepts. See the JBossWS documentation for options and more information.

Report a bug

30.4.2. Custom 'action' option

```
action=gtn.UsernameToken.ifAvailable
```

When this option is activated, the interceptor will set the action to 'UsernameToken' when the received SOAP message contains ws-security headers. If no ws-security header is included in the message, then no action is taken and the interceptor is not run. This is useful for dealing with authenticated and unauthenticated users trying to access the producer.

Report a bug

30.5. Configuring WSRP using the User name Token and User Propagation

30.5.1. Producer Setup

1. create the following file: **`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/GTNSubjectCreatingInterceptor.properties`**
2. set the content of **`GTNSubjectCreatingInterceptor.properties`** created in step 1 to:

   ```
   action=gtn.UsernameToken.ifAvailable
   ```

3. start the producer server

Warning

This example configuration does not encrypt the message. This means the username and password will be sent between the producer and consumer in plain text. This is a security concern and is only being shown as a simple example. It is up to administrators to properly configure the WSS4J Interceptors to encrypt messages or to only use https communication between the producer and consumer.

Report a bug

30.5.2. Consumer Setup

Procedure 30.1. Configuring a Consumer

1. Create the following file: **`standalone/configuration/gatein/wsrp/cxf/ws-security/consumer/WSS4JOutInterceptor.properties`**
2. Set the content of the **`WSS4JOutInterceptor.properties`** to the following parameters:

```
passwordType=PasswordText
user=gtn.current.user
action=gtn.UsernameToken.ifCurrentUserAuthenticated
passwordCallbackClass=org.gatein.wsrp.wss.cxf.consumer.CurrentUserPasswordCallback
```

3. Start the consumer server.
4. In the WSRP admin portlet, click the 'enable ws-security' checkbox
5. Access a remote portlet (for example, the user identity portlet included as an example portlet) and verify that the authenticated user is the same as the one on the consumer.

Report a bug

30.6. Securing WSRP Endpoints using Encryption and Signing

30.6.1. Sample Configuration for securing the Endpoints using Encryption and Signing

The following steps outline how to configure the producer and consumer to encrypt and sign SOAP messages passed between the producer and consumer. This example only deals with SOAP messages being sent between the producer and consumer, and not with user propagation.

Note

Some of the configuration options specified here are based on the content at http://cxf.apache.org/docs/ws-security.html and http://www.jroller.com/gmazza/entry/cxf_x509_profile More information may be available at these sites.

Report a bug

30.6.2. Password Callback Class

WSS4J uses a Java class to specify the password when performing any security related actions. For the purpose of these encryption and signing examples, we will use the same password for the producer's and consumer's keystore (wsrpAliasPassword). This simplifies configuration because one password callback class can be used for both the producer and consumer.

Example 30.1. Example test.TestCallbackHandler Class

```
package test;
 
import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
 
import org.apache.ws.security.WSPasswordCallback;
import org.gatein.wsrp.wss.cxf.consumer.CurrentUserPasswordCallback;
 
public class TestCallbackHandler implements CallbackHandler
{
 
    @Override
    public void handle(Callback[] callbacks) throws
IOException,
           
UnsupportedCallbackException
    {
 
        //First check if we have
any user name token call backs to add.
        //NOTE: only needed if
using username tokens, and you want the currently authenticated users
password added
        CurrentUserPasswordCallback
currentUserPasswordCallback = new CurrentUserPasswordCallback();
       
currentUserPasswordCallback.handle(callbacks);
 
        for (Callback callback:
callbacks)
        {
            if
(callback instanceof WSPasswordCallback)
            {
           &#16
0;    WSPasswordCallback wsPWCallback =
(WSPasswordCallback)callback;
           &#16
0;    // since the CurrentUserPasswordCallback already
handles the USERNAME_TOKEN case, we don&apos;t want to set it in this
case
           &#16
```

```
0;    if (wsPWCallback.getUsage() !=
WSPasswordCallback.USERNAME_TOKEN)
           &#16
0;    {
           &#16
0;       
wsPWCallback.setPassword(&quot;wsrpAliasPassword&quot;);
           &#16
0;    }
            }
        }
    }
}
```

Note

CallbackHandler implementations are provided to the portal using the standard Java ServiceLoader infrastructure. CallbackHandler implementations need to be bundled in a jar containing a file **`META-INF/services/javax.security.auth.callback.CallbackHandler`** specifying the fully qualified name of the CallbackHandler implementation class. This jar then needs to be put in the **`gatein/extensions`** directory of the portal installation.

A working example of a CallbackHandler implementation is available from https://github.com/gatein/gatein-wsrp/tree/master/examples/wss-callback

Report a bug

30.6.3. Configuring the Keystores

Note

In this example we are making it a bit easier by specifying the same keystore password for both the producer and consumer, as they can use the same password callback class.

1. Generate the producer's private encryption keys

```
 keytool -genkey -alias producerAlias -keypass wsrpAliasPassword -
keystore producer.jks -storepass keyStorePassword -dname
&quot;cn=producerAlias&quot; -keyalg RSA
```

2. Export the producer's public key

```
 keytool -export -alias producerAlias -file producerkey.rsa -
keystore producer.jks -storepass keyStorePassword
```

3. Generate the consumer's private encryption keys

```
 keytool -genkey -alias consumerAlias -keypass wsrpAliasPassword -
keystore consumer.jks -storepass keyStorePassword -dname
&amp;quot;cn=consumerAlias&amp;quot; -keyalg RSA
```

4. Export the consumer's public key

```
 keytool -export -alias consumerAlias -file consumerkey.rsa -
keystore consumer.jks -storepass keyStorePassword
```

5. Import the consumer's public key into the producer's keystore

```
 keytool -import -alias consumerAlias&amp;#160; -file consumerkey.rsa -
keystore producer.jks -storepass keyStorePassword -noprompt
```

6. Import the producer's public key into the consumer's keystore

```
 keytool -import -alias producerAlias&amp;#160; -file producerkey.rsa -
keystore consumer.jks -storepass keyStorePassword -noprompt
```

7. Copy the **`producer.jks`** file to the **`standalone/configuration/gatein/wsrp/cxf/ws-security/producer`** directory on the producer

8. Copy the **`consumer.jks`** file to the **`standalone/configuration/gatein/wsrp/cxf/ws-security/consumer`** directory on the consumer

Report a bug

30.6.4. Configuring the Producer

1. Create **`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/WSS4JInInterceptor.properties`** with the following content. This will configure the incoming message between the producer and the consumer

```
action=Signature Encrypt Timestamp
signaturePropFile=producer-security.properties
decryptionPropFile=producer-security.properties
passwordCallbackClass=test.TestCallbackHandler
```

2. Create **`standalone/configuration/gatein/wsrp/cxf/ws-security/producer/WSS4JOutInterceptor.properties`** with the following content. This will configure the outgoing message between the producer and the consumer

```
action=Signature Encrypt Timestamp
signaturePropFile=producer-security.properties
encryptionPropFile=producer-security.properties
passwordCallbackClass=test.TestCallbackHandler
user=producerAlias
encryptionUser=consumerAlias
signatureUser=producerAlias
```

3. Create **standalone/configuration/gatein/wsrp/cxf/ws-security/producer/producer-security.properties** with the following content:

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.compo
nents.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.password=keyStorePassw
ord
org.apache.ws.security.crypto.merlin.file=producer.jks
```

4. The **passwordCallbackClass** property in these configuration files needs to match the fully qualified name of your CallbackHandler implementation class. In our case, it is **test.TestCallbackHandler** .

Report a bug

30.6.5. Configuring the Consumer

1. Create standalone/ **configuration/gatein/wsrp/cxf/ws-security/consumer/WSS4JOutInterceptor.properties** with the following content. This will configure the outgoing message between the consumer and the producer

```
action=Signature Encrypt Timestamp
signaturePropFile=consumer-security.properties
encryptionPropFile=consumer-security.properties
passwordCallbackClass=test.TestCallbackHandler
user=consumerAlias
encryptionUser=producerAlias
signatureUser=consumerAlias
```

2. Create standalone/ **configuration/gatein/wsrp/cxf/ws-security/consumer/WSS4JInInterceptor.properties** with the following content. This will configure the incoming message between the consumer and the producer

```
action=Signature Encrypt Timestamp
signaturePropFile=consumer-security.properties
decryptionPropFile=consumer-security.properties
passwordCallbackClass=test.TestCallbackHandler
```

3. Create standalone/configuration/gatein/wsrp/cxf/ws-security/consumer/consumer-security.properties with the following content:

```
org.apache.ws.security.crypto.provider=org.apache.ws.security.compo
nents.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=jks
org.apache.ws.security.crypto.merlin.keystore.password=keyStorePassw
ord
org.apache.ws.security.crypto.merlin.file=consumer.jks
```

4. The **passwordCallbackClass** property in these configuration files needs to match the fully qualified name of your CallbackHandler implementation class. In our case, it is **test.TestCallbackHandler** .

Report a bug

30.7. Configuring WSRP using User name Token, Encryption and Signing with User Propagation

30.7.1. Sample Configuration using User name Token, Encryption and Signing with User Propagation

This section outline how to configure the producer and consumer to encrypt and sign the soap message as well as use user propagation between the producer and consumer.

Report a bug

30.7.2. Configure the Producer

Follow the steps outlined in the Section 30.6.1, “Sample Configuration for securing the Endpoints using Encryption and Signing” but make the following changes:

1. rename the **`WSS4JInInterceptor.properties`** file to **`GTNSubjectCreatingInterceptor.properties`**
2. set the action property in **`GTNSubjectCreatingInterceptor.properties`** as:

```
action= gtn.UsernameToken.ifAvailable Signature Encrypt Timestamp
```

3. set the passwordType in **`GTNSubjectCreatingInterceptor.properties`** as:

```
passwordType=PasswordText
```

Report a bug

30.7.3. Configure the Consumer

Follow the steps outlined in the Section 30.6.1, “Sample Configuration for securing the Endpoints using Encryption and Signing” section but make the following changes:

1. set the action property in **`WSS4JOutInterceptor.properties`** as:

```
action=gtn.UsernameToken.ifCurrentUserAuthenticated Signature
Encrypt Timestamp
```

2. set the user in the **`WSS4JOutInterceptor.properties`** as:

```
user=gtn.current.user
```

3. set the passwordType in the **`WSS4JOutInterceptor.properties`** as:

```
passwordType=PasswordText
```

Report a bug

Chapter 31. Using Remote WSRP Portlets

To be able to consume WSRP portlets exposed by a remote producer, the portal's WSRP consumer must know how to access that remote producer. You can configure access to a remote producer using the provided configuration portlet. Alternatively, it is also possible to configure access to remote producers using an XML descriptor, though it is recommended (and easier) to do so via the configuration portlet.

Once a remote producer has been configured, the portlets that it exposes are then available in the Application Registry to be added to categories and then to pages.

Report a bug

31.1. Configuring a Remote Producer using the Configuration Portlet

This section will cover the steps of defining access to a remote producer using the configuration portlet so that its portlets can be consumed within the portal. We will configure access to NetUnity's public WSRP producer.

Some WSRP producers do not support chunked encoding that is activated by default by JBoss WS. If your producer does not support chunked encoding, your consumer will not be able to properly connect to the producer. This will manifest itself with the following error: **`Caused by: org.jboss.ws.WSException: Invalid HTTP server response [503] - Service Unavailable`**. See http://community.jboss.org/wiki/Workaroundwhenchunkedencodingisnotsupported for more details.

The portal provides a portlet to configure access (among other functions) to remote WSRP Producers graphically. Starting with 6.1.0, the WSRP configuration portlet is installed by default. You can find it at http://localhost:8080/portal/login?initialURI=%2Fportal%2Fprivate%2Fclassic%2FwsrpConfiguration&username=root&password=gtn

You should see a screen similar to:

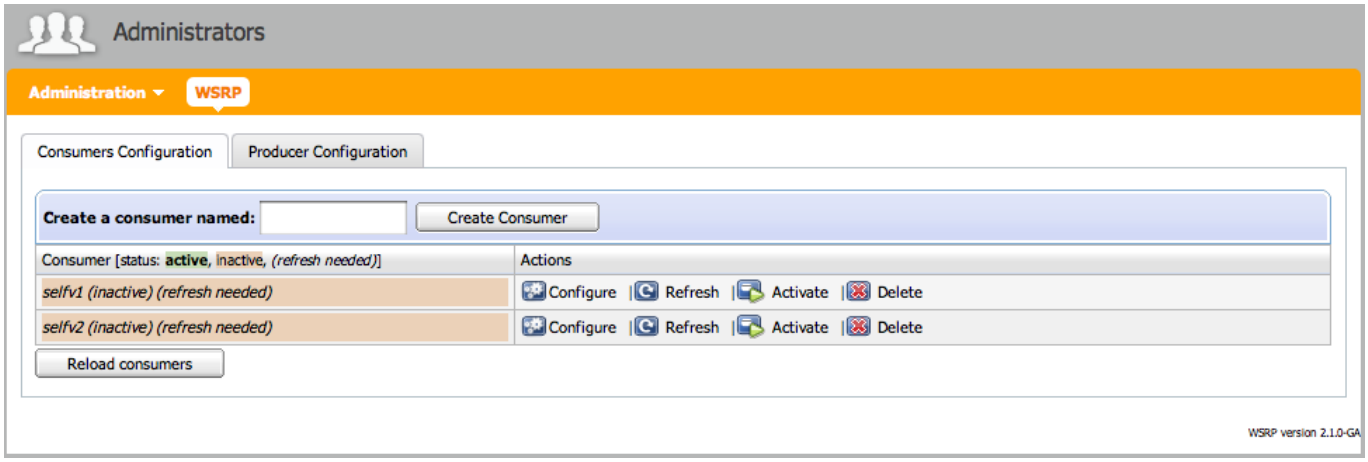


Figure 31.1. Consumers Configuration Screen

A Consumer can be active or inactive. Activating a Consumer means that it is ready to act as a portlet provider. Note also that a Consumer can be marked as requiring refresh meaning that the information held about it might not be up to date and refreshing it from the remote Producer might be a good idea. This can happen for several reasons: the service description for that remote Producer has not

been fetched yet, the cached version has expired or modifications have been made to the configuration that could potentially invalidate it, thus requiring re-validation of the information.

Report a bug

31.2. Access Remote Producers

31.2.1. Configuring Access to Remote Producers using XML

While it is recommended to use the WSRP Configuration portlet to configure Consumers, the component provides an alternative way to configure consumers by adding an XML file called **wsrp-consumers-config.xml** in the ***JPP_DIST*/standalone/configuration/gatein/wsrp/** directory.

Note

An XML Schema defining which elements are available to configure Consumers through XML can be found in ***JPP_DIST*/modules/org/gatein/wsrp/main/wsrp-integration-api-2.2.2.Final.jar/xsd/gatein_wsrp_consumer_1_0.xsd**

Important

Once the XML configuration file for consumers has been read upon the WSRP service first start, the associated information is put under control of JCR (Java Content Repository). Subsequent launches of the WSRP service will use the JCR-stored information and ignore the content of the XML configuration file.

This section covers what information is required to configure access to a remote producer.

Provide an identifier for the producer you are configuring so that you can reference it afterwards using the mandatory **id** attribute of the <wsrp-producer> element.

The portal also must know about the remote producer's endpoints to be able to connect to the remote web services and perform WSRP invocations. This is accomplished by specifying the URL for the WSDL description for the remote WSRP service, using the <endpoint-wsdl-url> element.

Both the **id** attribute and <endpoint-wsdl-url> elements are required for a functional remote producer configuration.

Report a bug

31.2.2. Additional Configuration to Remote Producers

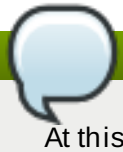
It is also possible to provide additional configuration, which, in some cases, might be important to establish a proper connection to the remote producer.

One such optional configuration concerns caching. To prevent useless round-trips between the local consumer and the remote producer, it is possible to cache some of the information sent by the producer (such as the list of offered portlets) for a given duration. The rate at which the information is refreshed is defined by the **expiration-cache** attribute of the **<wsrp-producer>** element which specifies the refreshing period in seconds. For example, providing a value of 120 for expiration-

cache means that the producer information will not be refreshed for 2 minutes after it has been somehow accessed. If no value is provided, the portal will always access the remote producer regardless of whether the remote information has changed or not. Since, in most instances, the information provided by the producer does not change often, it is recommended that you use this caching facility to minimize bandwidth usage.

It is also possible to define a timeout after which WS operations are considered as failed. This is helpful to avoid blocking the WSRP service, waiting on the service that does not answer. Use the **ws-timeout** attribute of the **<wsrp-producer>** element to specify how many milliseconds the WSRP service will wait for a response from the remote producer before timing out and giving up.

Additionally, some producers require consumers to register with them before authorizing them to access their offered portlets. If you know that information beforehand, you can provide the required registration information in the producer configuration so that the consumer can register with the remote producer when required.



Note

At this time, though, only simple String properties are supported and it is not possible to configure complex registration data. This should, however, be sufficient for most cases.

Registration configuration is done via the **<registration-data>** element. If the remote producer does not require any registration properties, you only need to provide an empty **<registration-data>** element. Values for the registration properties required by the remote producer can be provided through **<property>** elements. Additionally, you can override the default consumer name automatically provided by the portal through the <consumer-name> element. If you choose to provide a consumer name, remember that this must uniquely identify your consumer.

Report a bug

31.3. Configuration Examples

31.3.1. Consumer Configuration

The **extension-component-2.2.2.Final.jar** extension and the **selfv1** and **selfv2** files are fictitious files used to demonstrate the configuration required for Consumers. These files are not present in the default configuration file **wsrp-consumers-config.xml**.

Example 31.1. selfv1 and selfv2 defined in wsrp-consumers-config.xml

To correctly configure the selfv1 and selfv2 consumers, declare the configuration in the***JPP_HOME*/gatein/extensions/gatein-wsrp-integration.ear/lib/extension-component-2.2.2.Final.jar/conf/wsrp-consumers-config.xml** file.

For example, this file shows the configuration for **selfv1** and **selfv2** consumers consisting of a cache, expiring every 500 seconds with a 50 second timeout for web service operations. Use this example as an indication of the configuration required.

```
<?xml version='1.0' encoding='UTF-8' ?>
<deployments
xmlns="http://www.gatein.org/xml/ns/gatein_wsrp_consumer_1_0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_wsrp_consumer_1
_0 http://www.jboss.org/portal/xsd/gatein_wsrp_consumer_1_0.xsd">
   <deployment>
      <wsrp-producer id="selfv1" expiration-cache="500" ws-
timeout="50000">
         <endpoint-wsdl-url>http://localhost:8080/wsrp-
producer/v1/MarkupService?wsdl</endpoint-wsdl-url>
         <registration-data/>
      </wsrp-producer>
   </deployment>
   <deployment>
      <wsrp-producer id="selfv2" expiration-cache="500" ws-
timeout="50000">
         <endpoint-wsdl-url>http://localhost:8080/wsrp-
producer/v2/MarkupService?wsdl</endpoint-wsdl-url>
         <registration-data/>
      </wsrp-producer>
   </deployment>
</deployments>
```

When customizing a file of this type, it is recommended to make a copy of the file as a roll-back point.

Report a bug

31.3.2. Example 2: Registration Data and Cache Expiry

Here is the configuration of the **selfv1** and **selfv2** consumers as found in ***JPP_HOME*/gatein/extensions/gatein-wsrp-integration.ear/lib/extension-component-2.2.2.Final.jar/conf/wsrp-consumers-config.xml** with a cache expiring every 500 seconds and with a 50 second timeout for web service operations.

Note

This file contains the default configuration suitable for most installations. If changes are required, it is recommended a copy of the file is taken before modifying.

Example 31.2. Registration Data and Cache Expiry Example

This example shows a WSRP descriptor with registration data and cache expiring every minute:

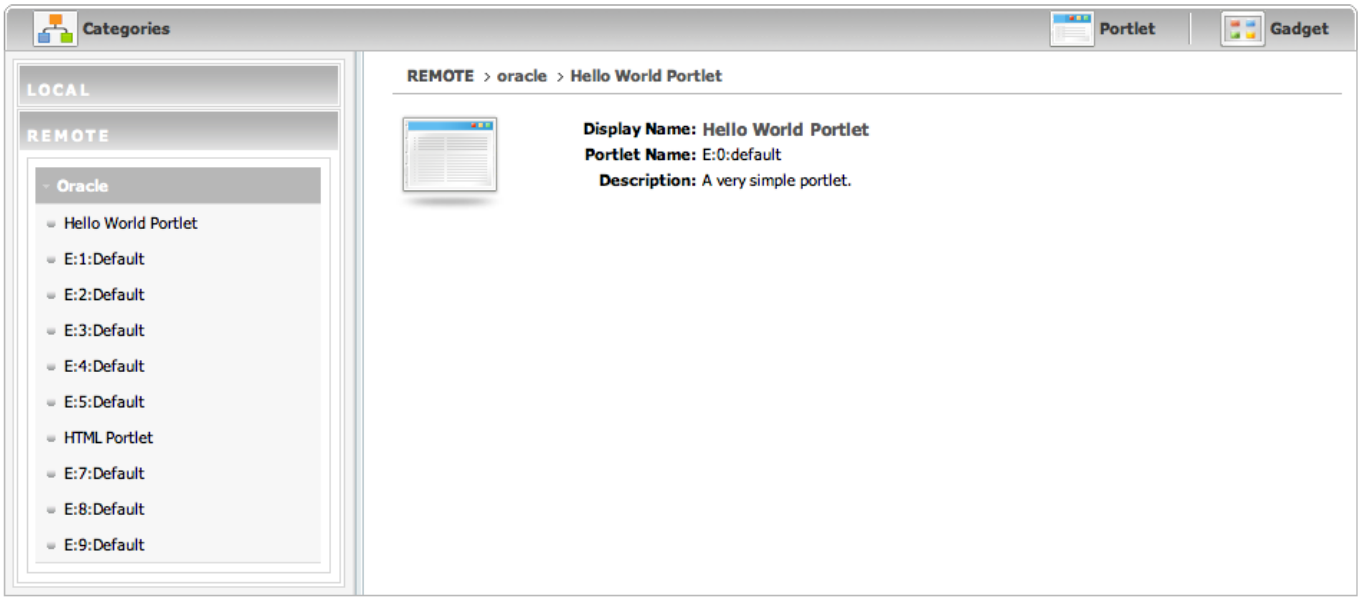
```
<?xml version='1.0' encoding='UTF-8' ?>
<deployments
xmlns="http://www.gatein.org/xml/ns/gatein_wsrp_consumer_1_0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.gatein.org/xml/ns/gatein_wsrp_consumer_1
_0 http://www.jboss.org/portal/xsd/gatein_wsrp_consumer_1_0.xsd">
<deployments>
   <deployment>
```

```
      <wsrp-producer id="AnotherProducer" expiration-cache="60">
         <endpoint-wsdl-url>http://example.com/producer/producer?
WSDL</endpoint-wsdl-url>
         <registration-data>
            <property>
               <name>property name</name>
               <lang>en</lang>
               <value>property value</value>
            </property>
         </registration-data>
      </wsrp-producer>
   </deployment>
</deployments>
```

Report a bug

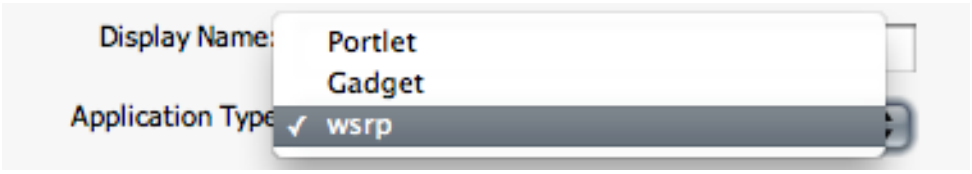
31.4. Adding remote portlets to categories

If you go to the Application Registry and examine the available portlets by clicking on the Portlet link, you will now be able to see remote portlets if you click on the REMOTE tab in the left column:



These portlets are, of course, available to be used such as regular portlets: they can be used in categories and added to pages. If you use the Import Applications functionality, they will also be automatically imported in categories based on the keywords they define.

More specifically, if you want to add a WSRP portlet to a category, you can access these portlets by selecting **`wsrp`** in the Application Type drop-down menu:



Report a bug

31.5. Adding remote portlets to pages

Since remote portlets can be manipulated like regular portlets, you can add them to pages like a regular portlet.

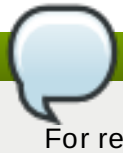
It is possible to add a remote portlet to a **pages.xml** configuration file. This is accomplished using the <wsrp> element instead of the <portlet> element in the **pages.xml** document.

While <portlet> references a local portlet using the name of the application in which the portlet is contained and the portlet name itself to identify which portlet to use, <wsrp> references a remote portlet using a combination of the consumer identifier for the producer publishing the portlet and the portlet handle identifying the portlet within the context of the producer.

The format for such a reference to a remote portlet takes the following format:

- The identifier of the consumer that accesses the remote producer publishing the remote portlet.
- A separator, currently a period (**.**).
- The portlet handle for that portlet, which is a string provided by the producer to identify the portlet.

Since there is currently is no easy way to determine the correct portlet handle, it is recommended that the graphical user interface is used to add remote portlets to pages instead of using **pages.xml**.



Note

For remote portlets published by the portal's WSRP producer, the portlet handles follow the **/<portlet application name>.<portlet name>** format.

Report a bug

31.5.1. Example: Adding Portlets

Example 31.3. Adding Portlets

Two portlets are defined for a page named **Test** in the **pages.xml** configuration. They reference the same portlet: one accessed locally and the other accessed through the **selfv2** consumer, which consumes the portal WSRP producer.

The first portlet is local (the **<portlet-application>** with the '**Added locally**' title) and follows the usual declaration. The second portlet (the one with the '**Added from selfv2 consumer**' title) comes from the **selfv2** consumer and uses the **<wsrp>** element instead of **<portlet>** element and follows the format for portlets coming from the portal WSRP producer.

```
<page>
   <name>Test</name>

   ...

   <portlet-application>
      <portlet>
         <application-ref>richFacesPortlet</application-ref>
         <portlet-ref>richFacesPortlet</portlet-ref>
      </portlet>
      <title>Added locally</title>
```

```
      ...

   </portlet-application>

   <portlet-application>
      <wsrp>selfv2./richFacesPortlet.richFacesPortlet</wsrp>
      <title>Added from selfv2 consumer</title>

      ...

   </portlet-application>
</page>
```

Report a bug

Chapter 32. Maintaining Consumers

32.1. Modifying a registration

32.1.1. Registration Modification for Service Upgrade

Producers often offer several levels of service depending on consumers' subscription levels (for example). This is implemented at the WSRP level with the registration concept: producers can assert which level of service to provide to consumers based on the values of given registration properties.

There might also be cases where you just want to update the registration information because it has changed. For example, the producer required you to provide a valid email and the previously email address is not valid anymore and needs to be updated.

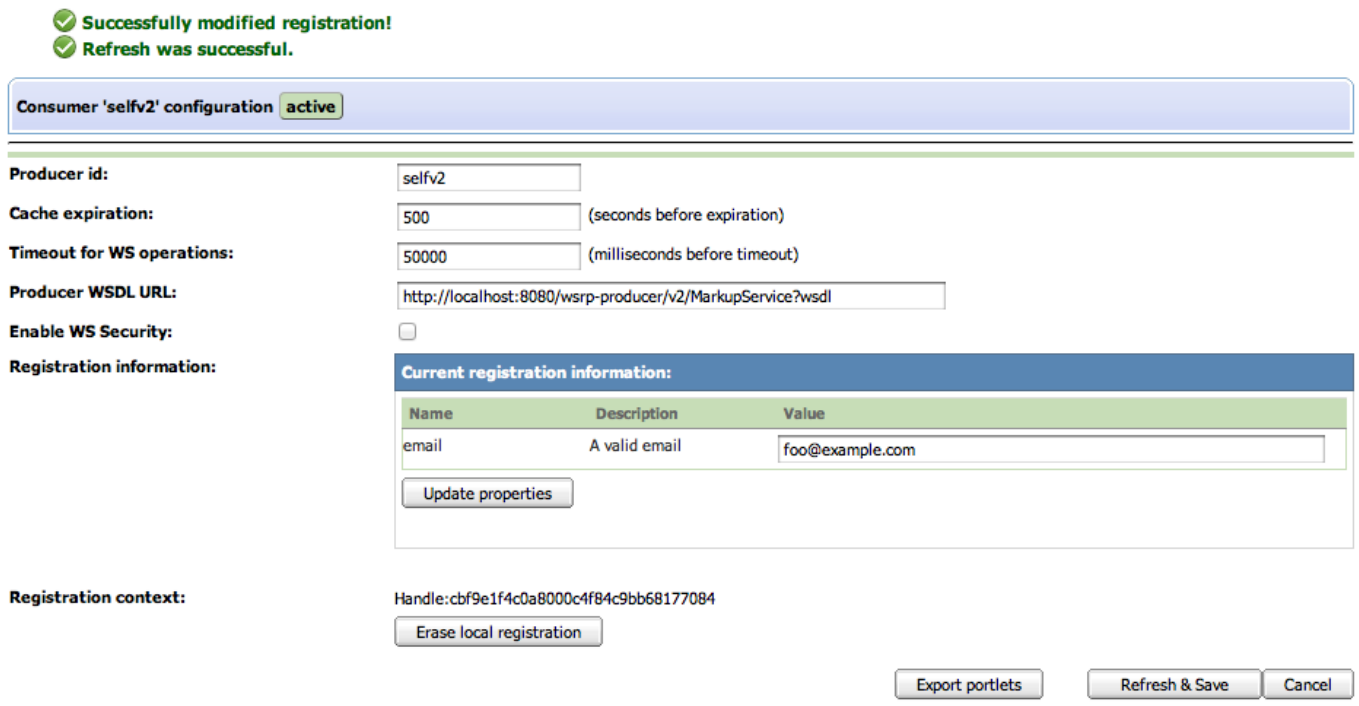
It is therefore sometimes necessary to modify the registration that concretizes the service agreement between a consumer and a producer. The following is an example of a producer requiring a valid email (via an **`email`** registration property) as part of its required information that consumers need to provide to be properly registered.

Suppose now that you would like to update the email address that you provided to the remote producer when you first registered. You will need to tell the producer that the registration data has been modified.

Select the consumer for the remote producer in the available consumers list to display its configuration. Assuming you want to change the email you registered with to **`foo@example.com`**, change its value in the field for the **`email`** registration property:

Now click on "Update properties" to save the change. A "Modify registration" button should now appear to let you send this new data to the remote producer:

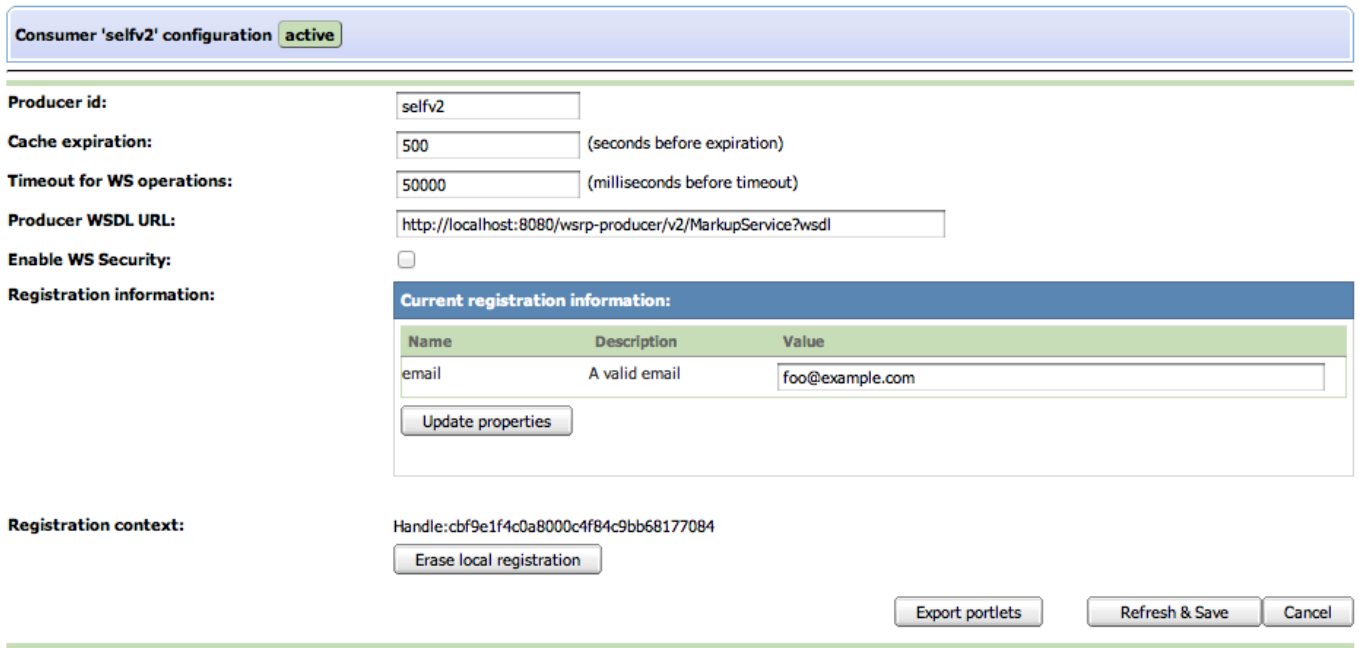
Click on this new button and, if everything went well and your updated registration has been accepted by the remote producer, you should see something similar to:



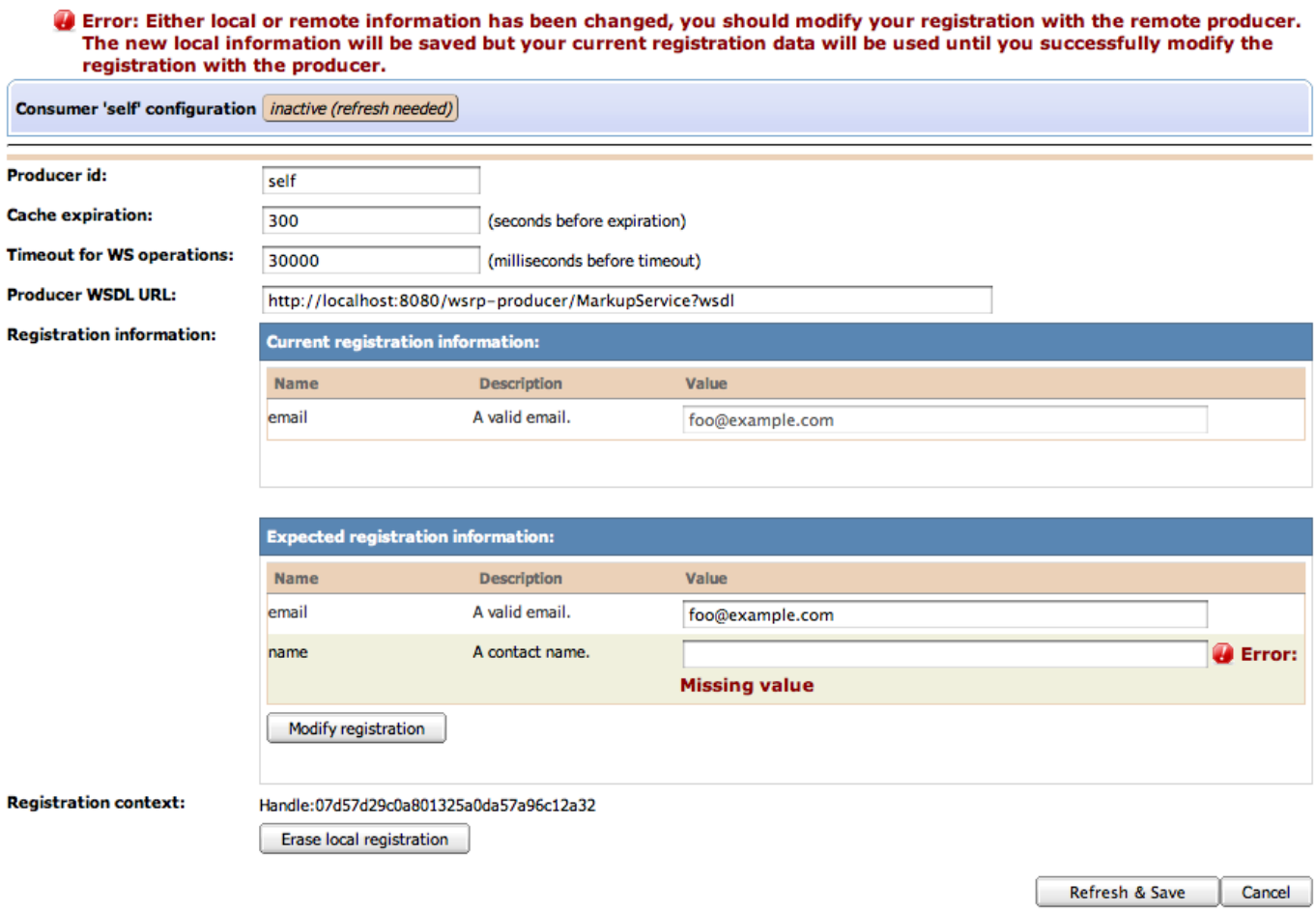
Report a bug

32.1.2. Registration modification on producer error

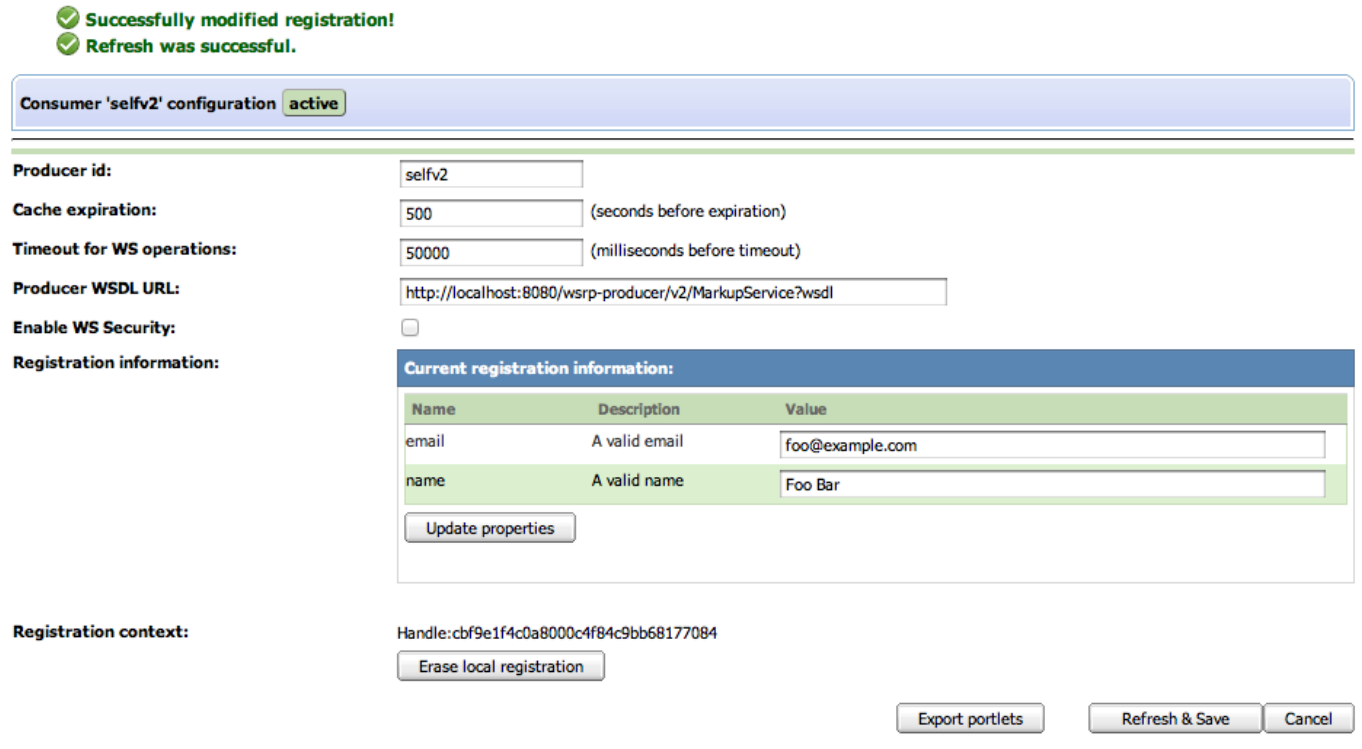
It can also happen that a producer administrator decided to change its requirement for registered consumers. The portal will attempt to help you in this situation. This section will walk through an example using the **`selfv2`** consumer (assuming that registration is requiring a valid value for an **`email`** registration property). If you go to the configuration screen for this consumer, you should see:



Now suppose that the administrator of the producer now additionally requires a value to be provided for a **`name`** registration property. Operations with this producer will now fail. If you suspect that a registration modification is required, you should go to the configuration screen for this remote producer and refresh the information held by the consumer by clicking "Refresh & Save":



As you can see, the configuration screen now shows the currently held registration information and the expected information from the producer. Enter a value for the **name** property and then click on "Modify registration". If all went well and the producer accepted your new registration data, you should see something similar to:



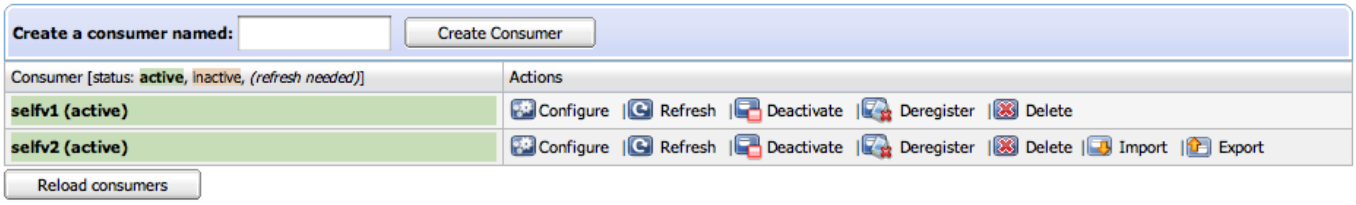
Note

WSRP 1 makes it rather difficult to ascertain for sure what caused an **`OperationFailedFault`** as it is the generic exception returned by producers if something did not quite happen as expected during a method invocation. This means that **`OperationFailedFault`** can be caused by several different reasons, one of them being a request to modify the registration data. Please take a look at the log files to see if you can gather more information as to what happened. WSRP 2 introduces an exception that is specific to a request to modify registrations thus reducing the ambiguity that exists when using WSRP 1.

Report a bug

32.2. Consumer Operations

Several operations are available from the consumer list view of the WSRP configuration portlet:



The available operations are:

- Configure: displays the consumer details and allows user to edit them
- Refresh: forces the consumer to retrieve the service description from the remote producer to refresh the local information (offered portlets, registration information, etc.)
- Activate/Deactivate: activates/deactivates a consumer, governing whether it will be available to provide portlets and receive portlet invocations
- Register/Deregister: registers/deregisters a consumer based on whether registration is required and/or acquired
- Delete: destroys the consumer, after deregistering it if it was registered
- Export: exports some or all of the consumer's portlets to be able to later import them in a different context
- Import: imports some or all of previously exported portlets

Note

Import/Export functionality is only available to WSRP 2 consumers of producers that support this optional functionality. Import functionality is only available if portlets had previously been exported.

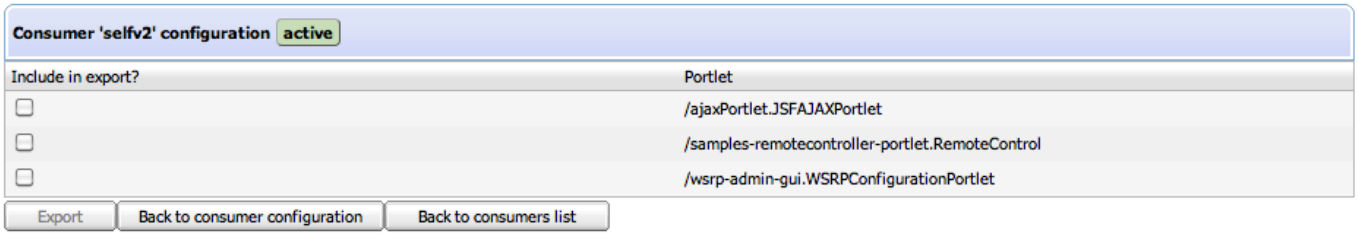
Report a bug

32.3. Importing and Exporting Portlets

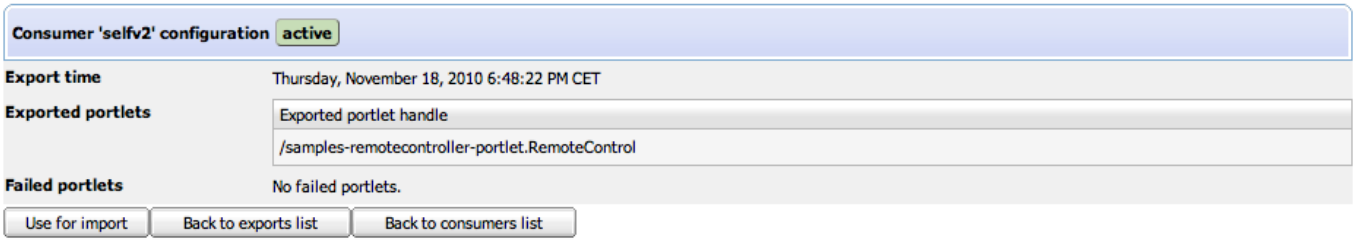
Import and export are new functionality added in WSRP 2. Exporting a portlet allows a consumer to get an opaque representation of the portlet which can then be use by the corresponding import operation to reconstitute it. It is mostly used in migration scenarios during batch operations. Since the portal does not currently support automated migration of portal data, the functionality that is provided as part of WSRP 2 is necessarily less complete than it could be with full portal support.

The import/export implementation in the portal allows users to export portlets from a given consumer. These portlets can then be used to replace existing content on pages. This is accomplished by assigning previously exported portlets to replace the content displayed by windows on the portal's pages. Below is an example to make things clearer.

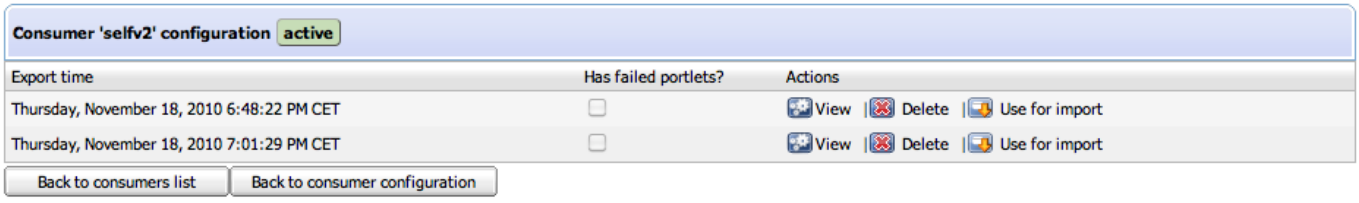
Clicking on the "Export" action for a given consumer will display the list of portlets currently made available by this specific consumer. An example of such a list is shown below:



Once portlets have been selected, they can be exported by clicking on the "Export" button thus making them available for later import:



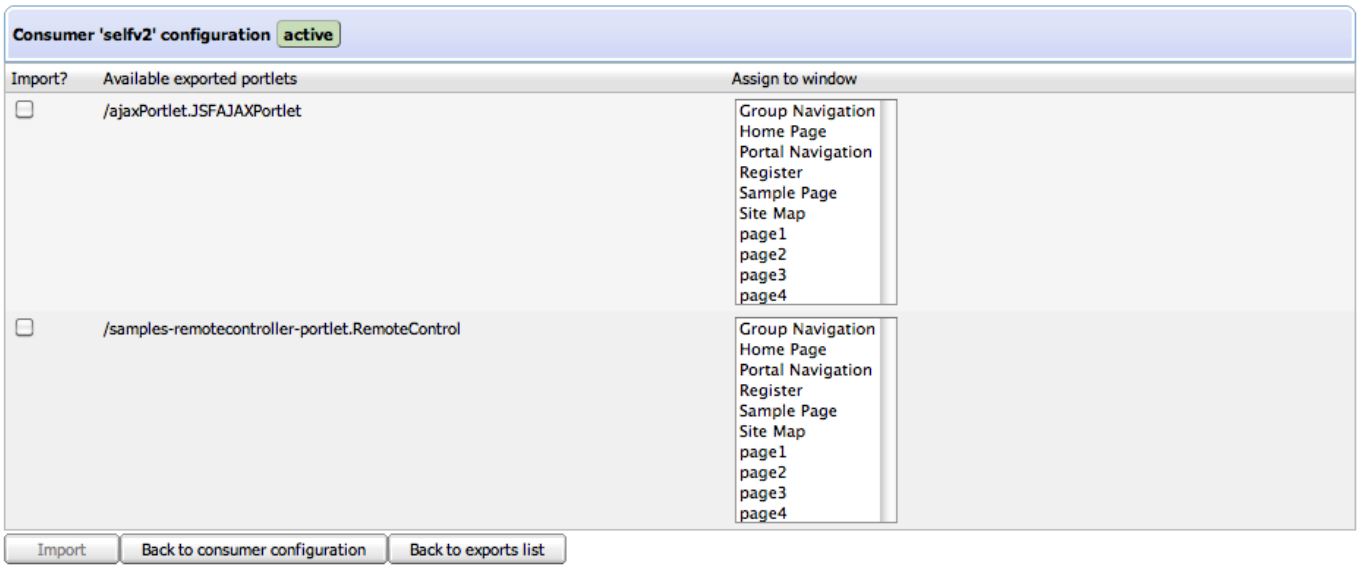
You can re-import the portlets directly by pressing the "Use for import" button or, on the Consumers list page, using the "Import" action for a given consumer. This assumes that you used that second option and that you currently have several available sets of previously exported portlets to import from. After clicking the action link, you should see a screen similar to the one below:



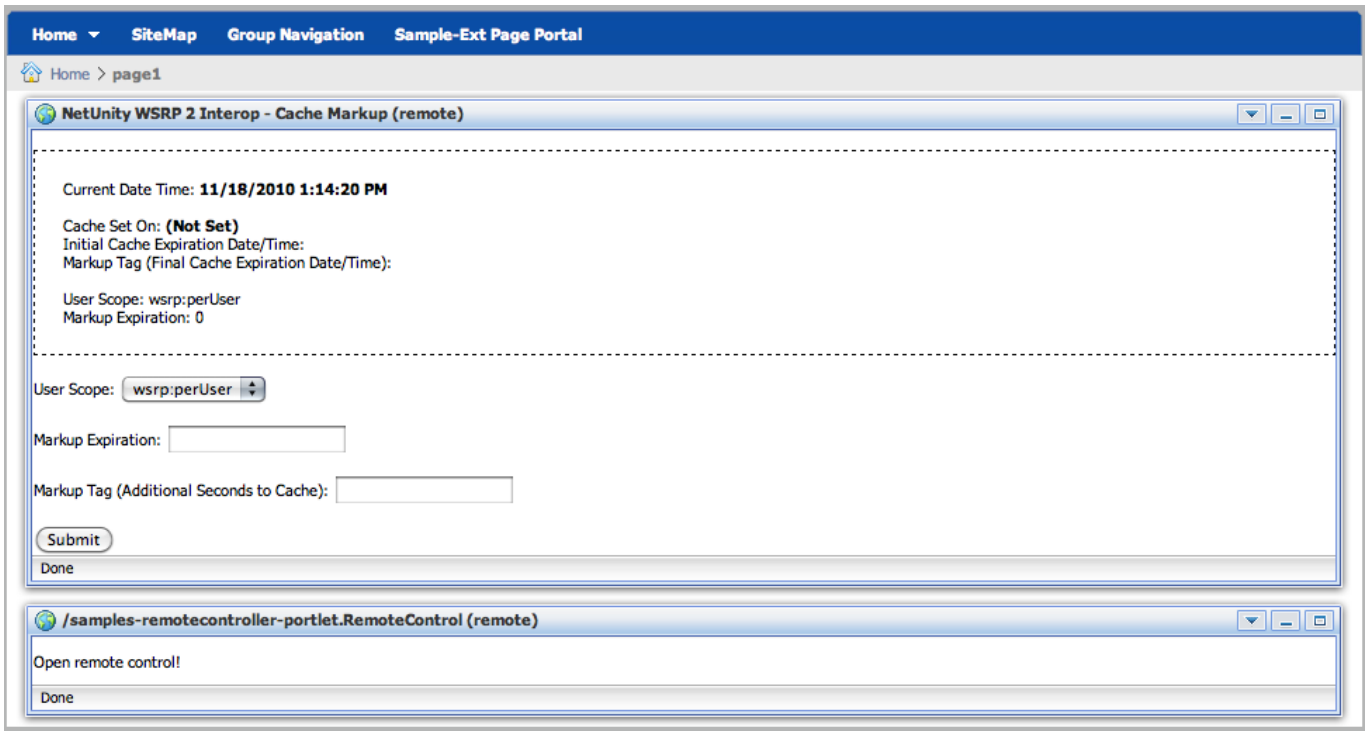
As you can see this screen presents the list of available exports with available operations for each:

- View: displays the export details as previously seen when the export was first performed
- Delete: deletes the selected export, asking you for confirmation first
- Use for import: selects the export to import portlets from

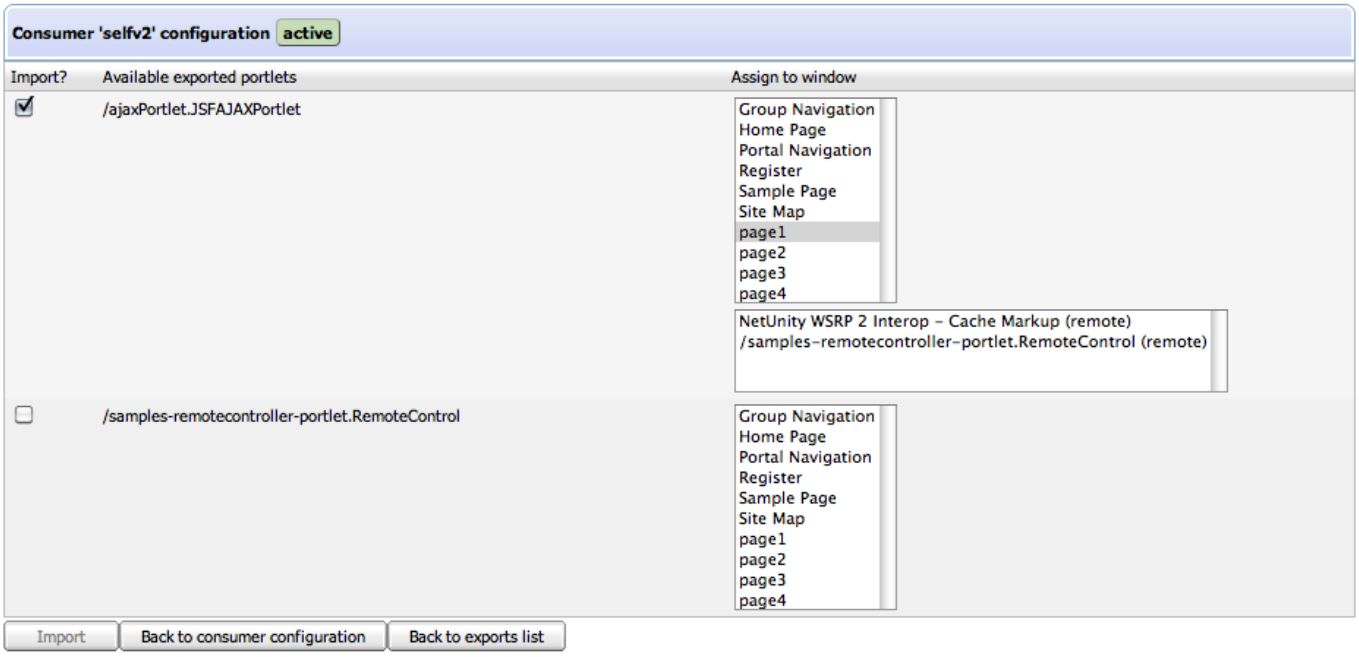
Once you have selected an export to import from, you will see a screen similar to the one below:



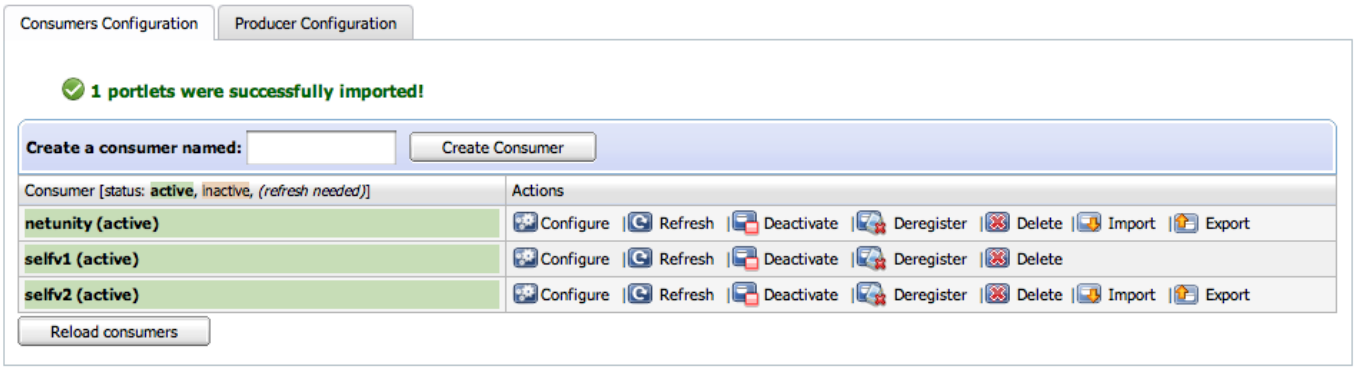
The screen displays the list of available exported portlets for the previously selected export. You can select which portlet you want to import by checking the checkbox next to its name. Next, you need to select the content of which window the imported portlet will replace. This process is done in three steps. This example assumes that you have the following page called **`page1`** and containing two windows called **`NetUnity WSRP 2 Interop - Cache Markup (remote)`** and **`/samples-remotecontroller-portlet.RemoteControl (remote)`** as shown below:



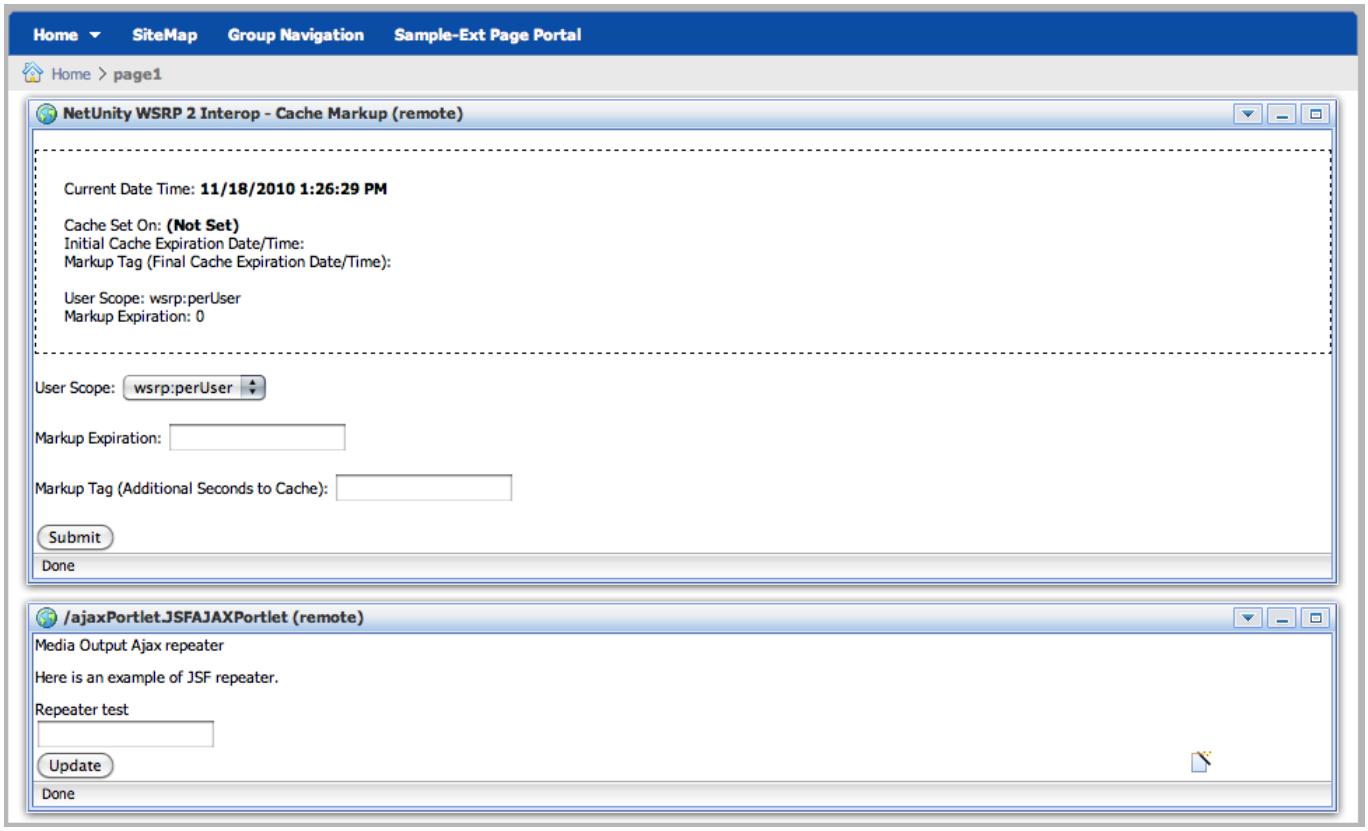
In this example, you want to replace the content of the **`/samples-remotecontroller-portlet.RemoteControl (remote)`** by the content of the **`/ajaxPortlet.JSFAJAXPortlet`** portlet that you previously exported. To do so, you will check the checkbox next to the **`/ajaxPortlet.JSFAJAXPortlet`** portlet name to indicate that you want to import its data and then select the **`page1`** in the list of available pages. The screen will then refresh to display the list of available windows on that page, similar to the one seen below:



Note that, at this point, you still need to select the window which content you want to replace before being able to complete the import operation. Select the **`/samples-remotecontroller-portlet.RemoteControl (remote)`** window, at which point the "Import" button will become enabled, indicating that you now have all the necessary data to perform the import. If all goes well, pressing that button should result in a screen similar to the one below:



If you now take a look at the **`page1`** page, you should now see that the content **`/samples-remotecontroller-portlet.RemoteControl (remote)`** window has been replaced by the content of the **`/ajaxPortlet.JSFAJAXPortlet`** imported portlet and the window renamed appropriately:



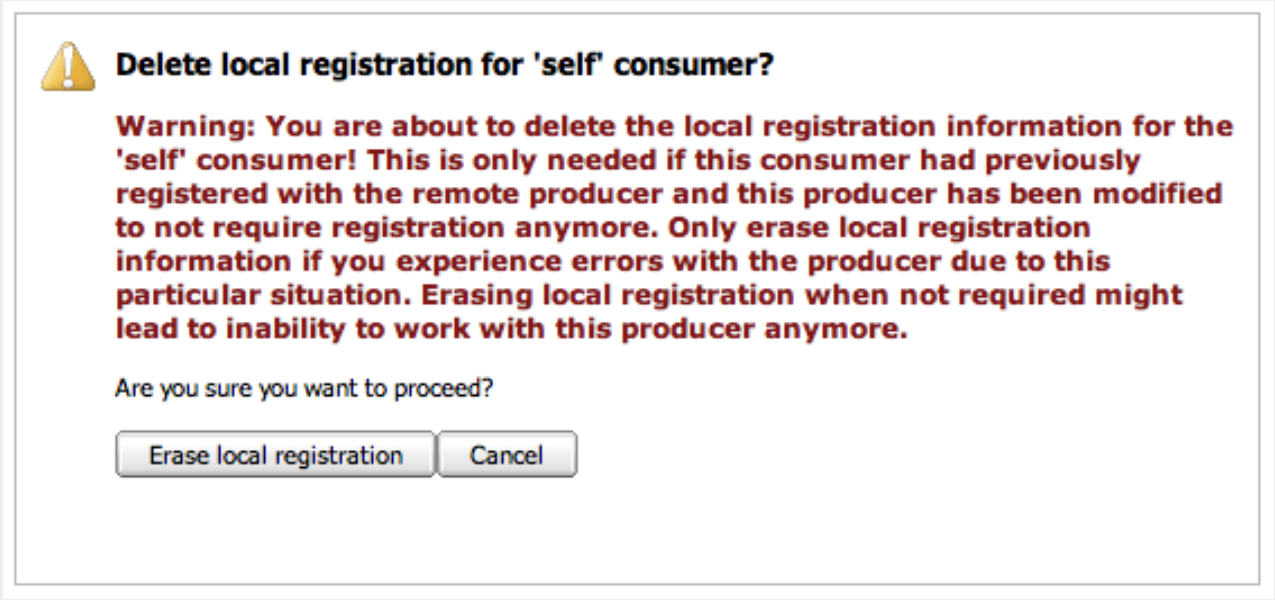
Report a bug

32.4. Erasing Local Registration Data

There are rare cases where it might be required to erase the local information without being able to de-register first. This is the case when a consumer is registered with a producer that has been modified by its administrator to not require registration anymore. If that ever was to happen (most likely, it will not), you can erase the local registration information from the consumer so that it can resume interacting with the remote producer. To do so, click on "Erase local registration" button next to the registration context information on the consumer configuration screen:

Warning:

This operation is dangerous as it can result in inability to interact with the remote producer if invoked when not required. A warning screen will be displayed to give you a chance to change your mind:



Report a bug

Chapter 33. Working with WSRP Extensions

The WSRP specifications allows for implementations to extend the protocol using Extensions . The portal provides a way for client code (for example, portlets) to interact with such extensions in the form of several classes and interfaces gathered within the **`org.gatein.wsrp.api.extensions`** package , the most important ones being **`InvocationHandlerDelegate`** , **`ConsumerExtensionAccessor`** and **`ProducerExtensionAccessor`** .

To be able to use this API, include the **`wsrp-integration-api-2.2.2.Final.jar`** (or higher) file to your project, where **`$WSRP_VERSION`** is the version of the portal WSRP implementation to use. This can be done by adding the following dependency to your maven project:

```
<dependency>
  <groupId>org.gatein.wsrp</groupId>
  <artifactId>wsrp-integration-api</artifactId>
  <version>$WSRP_VERSION</version>
</dependency>
```

Report a bug

33.1. Using WSRP Extensions

33.1.1. Infrastructure for InvocationHandlerDelegate

Using the **`InvocationHandlerDelegate`** infrastructure, custom behavior can be inserted on either consumer or producer sides to enrich WSRP applications before and after portlet requests and responses. For more information on the interface, see the Javadoc for **`org.gatein.wsrp.api.extensions.InvocationHandlerDelegate`**, which is available from https://github.com/gatein/gatein-wsrp/blob/master/api/src/main/java/org/gatein/wsrp/api/extensions/InvocationHandlerDelegate.java.

Warning

Because **`InvocationHandlerDelegate`** is a very generic interface, it could potentially be used for more than working with WSRP extensions alone. Because this interface has access to internal portal classes, it is important to treat access to these internal classes as read-only to prevent any unintentional side effects.

Report a bug

33.1.2. Injecting InvocationHandlerDelegate implementations

Implementations of **`InvocationHandlerDelegatemust`** follow the same constraints as **`RegistrationPolicy`** implementations as detailed in Customization of Registration handling behavior section of the Configuring GateIn's WSRP Producer chapter, in essence, they **must** follow the Java **`ServiceLoader`** architectural pattern and be deployed in the appropriate **`JPP_HOME/gatein/extensions`** directory.

You can specify only one **`InvocationHandlerDelegate`** implementation per side: one implementation for the consumer and another one for the producer. This is accomplished by passing the fully classified class name to the appropriate system property when the portal is started:

Table 33.1. System Property Classname

WSRP Side	System property
consumer	org.gatein.wsrp.consumer.handlers.delegate
producer	org.gatein.wsrp.producer.handlers.delegate

Example 33.1. Example

```
./standalone.sh -
Dorg.gatein.wsrp.consumer.handlers.delegate=com.example.FooInvocationH
andlerDelegate
```

will inject the **`com.example.FooInvocationHandlerDelegate`** class on the consumer side, assuming that class implements the **`org.gatein.wsrp.api.extensions.InvocationHandlerDelegate`** interface and is packaged and deployed appropriately as explained above.

Report a bug

33.1.3. Accessing extensions from client code

You can access extensions from client code using **`ConsumerExtensionAccessor`** and **`ProducerExtensionAccessor`** on the consumer and producer, respectively. Each interface provides several methods but you should only have to ever call two of them on each, as shown in the following table:

Table 33.2. Table Title

Interface	Relevant methods
`ConsumerExtensionAccessor`	**`public void addRequestExtension(Class targetClass, Object extension)`** **`public List<UnmarshalledExtension> getResponseExtensionsFrom(Class responseClass)`**
`ProducerExtensionAccessor`	**`List<UnmarshalledExtension> getRequestExtensionsFor(Class targetClass)`** **`void addResponseExtension(Class wsrpResponseClass, Object extension)`**

See the Javadoc for these classes for more details.

Adding and accessing extensions is supported from a core subset of WSRP classes pertaining to markup, interaction, resource or event requests and responses.

Important

Use **org.w3c.dom.Element** values as extensions to ensure interoperability.

Table 33.3. Supported Request and Response WSRP Classes

Request classes	Response classes
org.oasis.wsrp.v2.InteractionParams	**org.oasis.wsrp.v2.MarkupResponse**
org.oasis.wsrp.v2.EventParams	**org.oasis.wsrp.v2.BlockingInteractionResponse**
org.oasis.wsrp.v2.MarkupParams	**org.oasis.wsrp.v2.HandleEventsResponse**
org.oasis.wsrp.v2.ResourceParams	**org.oasis.wsrp.v2.ResourceResponse**

Report a bug

33.2. WSRP Implementation Example

We also provide a complete, end-to-end example to get you started, which you can find at https://github.com/gatein/gatein-wsrp/tree/master/examples/invocation-handler-delegate . This example shows how **ExampleConsumerInvocationHandlerDelegate** , a consumer-side **InvocationHandlerDelegate** implementation, can add information extracted from the consumer and pass it along to the producer, working in conjunction with **ExampleProducerInvocationHandlerDelegate** , the associated producer-side **InvocationHandlerDelegate** , to establish a round-trip communication channel outside of the standard WSRP protocol, implementing the following scenario:

- **ExampleConsumerInvocationHandlerDelegate** attaches to the consumer to add the current session id as an extension to render requests sent to the producer.
- **ExampleProducerInvocationHandlerDelegate** provides the counterpart of **ExampleConsumerInvocationHandlerDelegate** on the producer. It checks incoming render requests for potential extensions matching what **ExampleConsumerInvocationHandlerDelegate** sends and adds an extension of its own to the render response so that the consumer-side delegate can know that the information it passed was properly processed.

To activate the InvocationHandlerDelegates on both the consumer and producer, start the portal instance as described in Chapter 34, *Configuring the WSRP Producer*.

Report a bug

Chapter 34. Configuring the WSRP Producer

The preferred way to configure the behavior of the portal's WSRP Producer is through the WSRP configuration portlet. Alternatively, it is possible to add an XML file called **wsrp-producer-config.xml** in the ***JPP_HOME*/standalone/configuration/gatein/wsrp** directory. Several aspects can be modified with respect to whether registration is required for consumers to access the Producer's services.

Important

Once the XML configuration file for the producer has been read upon the WSRP service first start, the associated information is put under control of the JCR (Java Content Repository). Subsequent launches of the WSRP service will use the JCR-stored information and ignore the content of the XML configuration file.

Note

An XML Schema defining which elements are available to configure Consumers via XML can be found in ***JPP_HOME*/modules/org/gatein/wsrp/main/wsrp-integration-api-2.2.2.Final.jar/xsd/gatein_wsrp_producer_1_0.xsd**

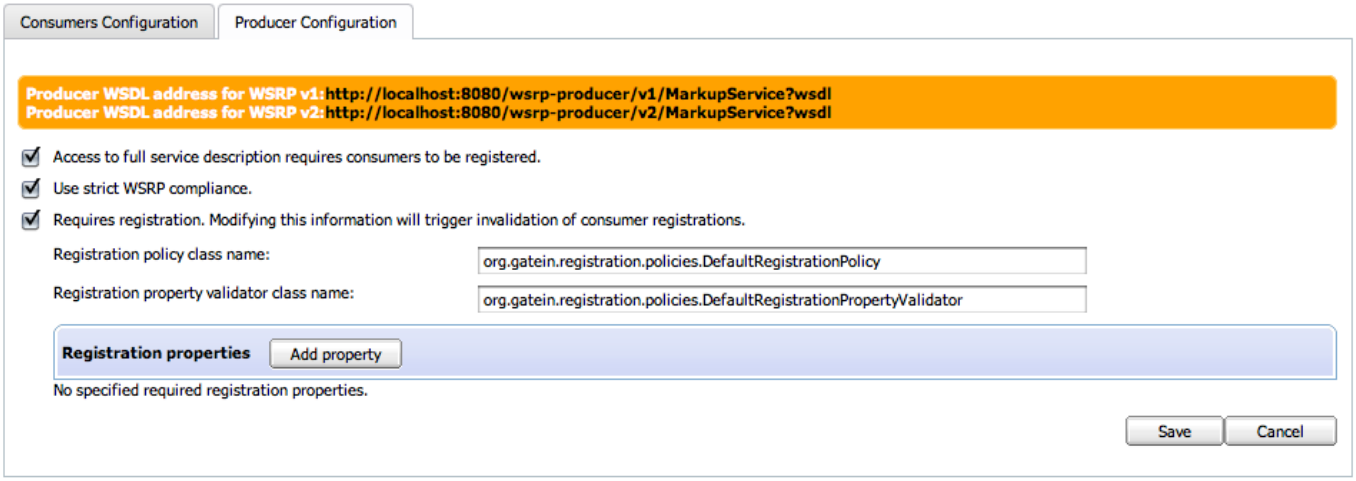
Report a bug

34.1. Default Producer Configuration

The default producer configuration requires that consumers register with it before providing access its services, but does not require any specific registration properties (apart from what is mandated by the WSRP standard). It does, however, require consumers to be registered before sending them a full service description. This means that the WSRP producer will not provide the list of offered portlets and other capabilities to unregistered consumers.

The producer also uses the default **RegistrationPolicy** paired with the default **RegistrationPropertyValidator**. This allows users to customize how the Portal's WSRP Producer determines whether a given registration property is valid or not.

The portal provides a web interface to configure the producer's behavior. It is accessed by clicking on the "Producer Configuration" tab of the "WSRP" page of the "admin" portal. The image below is what you should see with the default configuration:



You can specify whether or not the producer will send the full service description to unregistered consumers, and, if it requires registration, which **`RegistrationPolicy`** to use (and, if needed, which **`RegistrationPropertyValidator`**), along with required registration property description for which consumers must provide acceptable values to successfully register.

The WSDL URLs to access the Portal's WSRP producer are displayed either in WSRP 1 or WSRP 2 mode.

See Also:

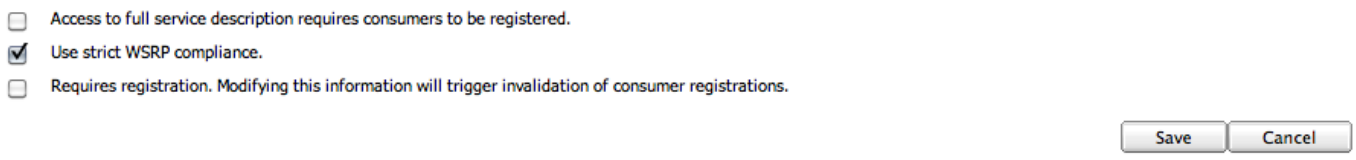
- Section 34.2, “Registration Configuration”

Report a bug

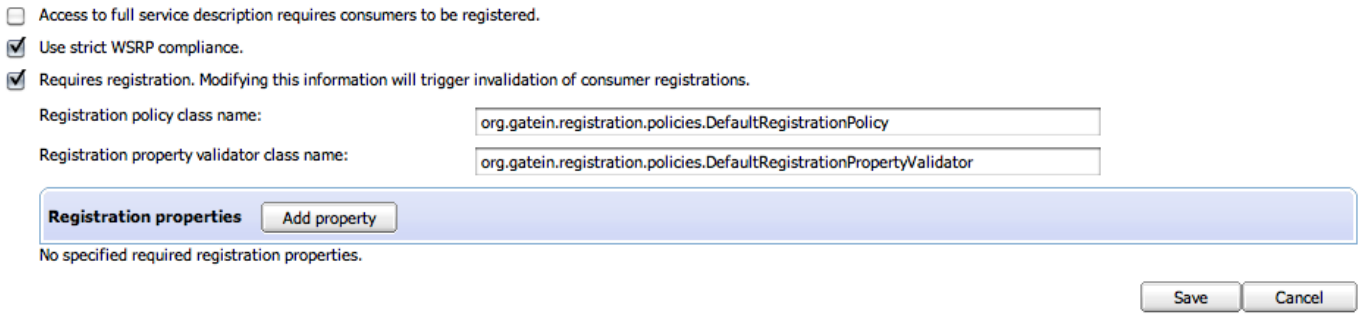
34.2. Registration Configuration

In order to require consumers to register with Portal's producer before interacting with it, you need to configure Portal's behavior with respect to registration. Registration is optional, as are registration properties. The producer can require registration without requiring consumers to pass any registration properties as is the case in the default configuration.

To configure the producer starting with a blank state:

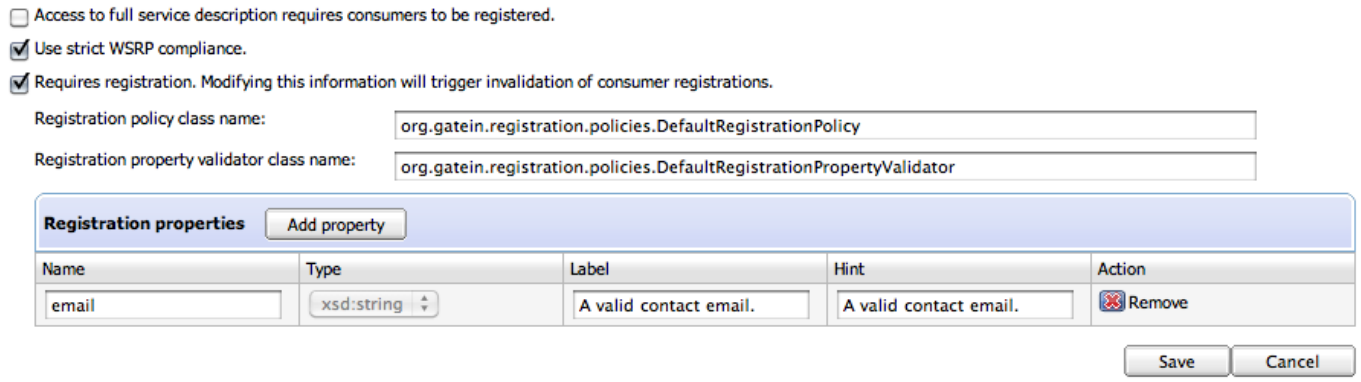


You will allow unregistered consumers to see the list of offered portlets so you leave the first checkbox ("Access to full service description requires consumers to be registered.") unchecked. You will, however, specify that consumers will need to be registered to be able to interact with the producer. Check the second checkbox ("Requires registration. Modifying this information will trigger invalidation of consumer registrations."). The screen should now refresh and display:



You can specify the fully-qualified name for your **`RegistrationPolicy`** and **`RegistrationPropertyValidator`** there. Keep the default value.

Add a registration property called **`email`**. Click "Add property" and enter the appropriate information in the fields, providing a description for the registration property that can be used by consumers to figure out its purpose:



Click "Save" to record the modifications.

Note

At this time, only String (xsd:string) properties are supported. If your application requires more complex properties, please supply feedback.

Note

If consumers are already registered with the producer, modifying the configuration of required registration information will trigger the invalidation of held registrations, requiring consumers to modify their registration before being able to access the producer again.

Report a bug

34.2.1. Customization of Registration handling behavior

Registration handling behavior can be customized by users to suit their Producer needs. This is accomplished by providing an implementation of the **`RegistrationPolicy`** interface. This interface defines methods that are called by Portal's Registration service so that decisions can be made appropriately. A default registration policy that provides basic behavior is provided and should be enough for most user needs.

While the default registration policy provides default behavior for most registration-related aspects,

there is still one aspect that requires configuration: whether a given value for a registration property is acceptable by the WSRP Producer. This is accomplished by plugging a **`RegistrationPropertyValidator`** in the default registration policy. This allows users to define their own validation mechanism for registration properties that are passed by a consumer to a producer.

See the Javadoc™ for https://github.com/gatein/gatein-wsrp/blob/master/producer/src/main/java/org/gatein/registration/RegistrationPolicy.java and https://github.com/gatein/gatein-wsrp/blob/master/producer/src/main/java/org/gatein/registration/policies/RegistrationPropertyValidator.java for more details on what is expected of each method.

Defining a registration policy is required for the producer to be correctly configured. If you don't provide one, the **`DefaultRegistrationPolicy`** associated to the **`DefaultRegistrationPropertyBehavior`** is used.

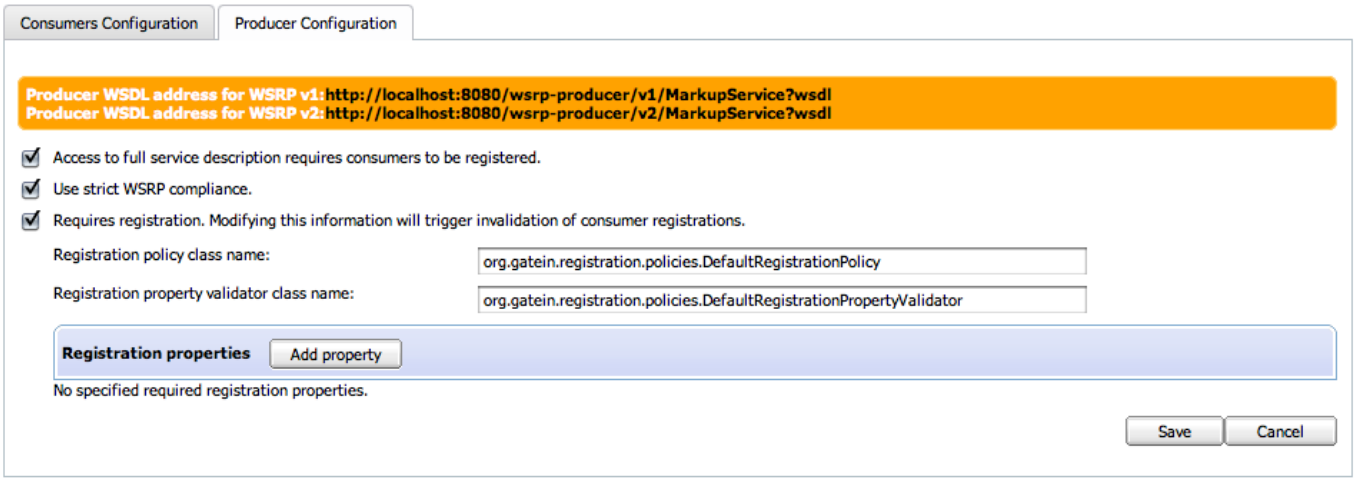


Figure 34.1. DefaultRegistrationPolicy

The portal can automatically detect deployed implementations of **`RegistrationPolicy`** and **`RegistrationPropertyValidator`**, assuming they conform to the following rules:

- The implementations must follow the Java ServiceLoader architecture:
  - They must be packaged as JARs
  - For **`RegistrationPolicy`** implementations, they must contain a special **`META-INF/services/org.gatein.registration.RegistrationPolicy`** file.
  - For **`RegistrationPropertyValidator`** implementations they must contain**`META-INF/services/org.gatein.registration.policies.RegistrationPropertyValidator`** and include a line with the fully qualified name of the implementation class.

Note

It is possible to package several implementations in the same JAR file, provided that each implementation class is referenced on its own line in the appropriate service definition file. To make things easier, we provide an example project of a RegistrationPolicy implementation and packaging at https://github.com/gatein/gatein-wsrp/tree/master/examples/policy.

- The implementation classes must provide a default, no-argument constructor for dynamic instantiation.
- The implementation JARs must be deployed in ***JPP_HOME*/gatein/extensions**.
- The implementation JARs should use the **.wsrp.jar** extension. This is not mandatory but helps process the file faster by marking it as a WSRP extension.

If you deployed the example **RegistrationPolicy** provided from the github repository (**registration-policy-example.wsrp.jar**) to the **JPP_HOME/gatein/extensions** directory, it will appear in the list of available policies in the producer configuration screen.

Report a bug

34.3. WSRP Validation Mode

The lack of conformance kit, and the wording of the WSRP specification leaves room for differing interpretations, resulting in interoperability issues. It is therefore possible to encounter issues when using consumers from different vendors.

A way to relax the validation that the WSRP producer performs on the data provided by consumers has been introduced to help with interoperability by accepting data that would normally be invalid. Note that this validation algorithm is only relaxed on aspects of the specification that are deemed harmless such as invalid language codes.

By default, the WSRP producer is configured in strict mode. In case of issues with a given consumer, you can skip the validation mode. To skip validation, clear the "Use strict WSRP compliance." check box on the Producer configuration screen.

Report a bug

Revision History

Revision 6.2.0-4	**Mon May 11 2015**	**Aakanksha Singh**
Prepared guide for 6.2.0 GA.		