



Gateway Portlet Configuration

Confidential & Proprietary

Contents

Contents	2
1.0 Introduction	4
1.1 Security Limitations	4
2.0 Configuration	6
2.1 The <gateway> Element	6
2.1.1 Example	6
2.1.2 Attributes of the <gateway> element	6
2.1.3 Minor sub-elements of the <gateway> element	6
2.2 The <sso-entry> Element	7
2.2.1 Example	7
2.2.2 Attributes of the <sso-entry> element	7
2.2.3 Minor sub-elements of the <sso-entry> element.....	8
2.3 The <target> Element	8
2.3.1 Example	8
2.3.2 Attributes of the <target> element.....	8
2.3.3 Minor sub-elements of the <target> element	9
2.4 The <sequence> Element	9
2.4.1 Example	9
2.4.2 Attributes of the <sequence> element	9
2.4.3 Minor sub-elements of the <sequence> element.....	10
2.4.4 Major sub-elements of the <sequence> element.....	10
2.5 The <sequence-start-trigger> element	10
2.5.1 Example	10
2.5.2 The <context> sub-element	10
2.5.3 The <image> sub-element	10
2.5.4 The <text> sub-element	11
2.6 The <window> Element	11
2.6.1 Example	11
2.6.2 Attributes of the <window> element.....	11
2.6.3 Minor sub-elements of the <window> element	11
2.7 The <authentication> Element	12
2.7.1 SsoMultiAuthentication.....	12
2.8 The Evaluator Element:	13
3.0 Declaring Gateway Portlets	14
3.1 The purpose of this section	14
3.2 Declaring portlets in general.....	14
3.3 Declaring AcademusApps portlets	15
3.4 Declaring new instances of the Academus Gateway SSO Portlet.....	15
3.4.1 The <portlet/> element wrapper	15
3.4.2 The <portlet-name/> element.....	15
3.4.3 The <portlet-class/> element.....	15
3.4.4 The “id” <init-param/> element.....	15
3.4.5 The “configPath” <init-param/> element	16
3.4.6 The “userContextTimer” <init-param/> element.....	16
3.4.7 The <expiration-cache/> element.....	16
3.4.8 The “<supported-locale/> element.....	16
3.4.9 The “<portlet-info/> compound element.....	16
3.4.10 The “<supports/> compound element	17
3.5 Wait, why do I have to configure gateway portlets in multiple places?.....	17

4.0 Finding Parameters for Your Application	18
5.0 Creating New Application Icons	19
5.1 Create the icon image.....	19
5.2 Make the icon image accessible as a CSS class.....	19
5.2.1 Pick a CSS class name.....	19
5.2.2 Create the new CSS class.....	19
6.0 Creating an instance of the SSO Gateway Portlet	20
6.1 Create the SSO Gateway Portlet Configuration File.....	20
6.2 Add the New Portlet to the Application's web.xml File.....	20
6.3 Add the New Portlet to the Application's portlet.xml File.....	21
6.4 Restart the Portal.....	21
6.5 Publish the Portlet.....	21
7.0 Appendix	23
7.1 SSO Gateway Portlet Configuration Example.....	23
7.2 SSO Gateway Portlet Example Configuration in Action.....	24
7.2.1 Main View: Links.....	24
7.2.2 Focused View: The SSO Test Application.....	24
7.3 Additional requirements in configuring the AJAX callback.....	25
7.3.1 Configuring the callback servlet in Academus Portal web.xml.....	25
7.3.2 Configuring the callback servlet in the web container.....	25

1.0 Introduction

The Gateway Portlet provides an extensible Single Sign-On (SSO) solution that allows both form-based (GET and POST) and HTTP-Basic authentication into external systems. In addition to straight-through authentication mapping from the portal environment, there is additional support for credential mapping by username and site; these mappings will be created by the user on first use of the application if the SSO application requires it.

New SSO applications can be configured using an XML configuration file, which allows an admin to specify the target URL, form parameters, and authentication mapping if desired. The form parameters may be constant strings, person attributes, portal credentials, or mapped authentication credentials. The configuration file also allows the administrator to specify how the application should be launched: in an iframe within the portal, or as a new browser window. All aspects of the window size and parameters are configurable in both instances.

The SSO application entry points are configurable to display as a list of applications that a user may log in to, or so that they are launched directly into an application making the SSO appear as a seamless integration.

The counterpart to Single Sign On is Single Sign Out. This means the ability to sign out of external applications without explicitly clicking the sign out functionality for each individual application. Academus does offer a Single Sign Out solution that will also support the closing of pop-up windows that had been launched from the portal. For more explanation of this Single Sign Out solution and how to configure it, see the [Single Sign Out Configuration Guide](#).

1.1 Security Limitations

Default configuration security limitations

The Academus SSO Gateway Portlet uses a method known as "password replay" or "store and forward" to provide access to other web applications without logging in. The advantage of this method is that it does not require the target applications to be modified at all. This method works with various applications: in-house developed, off-the-shelf, and even external applications. The disadvantage to this approach is that it necessarily creates potential security risks stemming from the storing and propagation of the user's password.

In order to accomplish the replay of the user's credentials to the target application, the SSO Gateway Portlet must provide the user's credentials to the browser in plain-text. The browser then automatically submits these credentials to the login form of the target application, thereby accomplishing the single sign-on effect without modification of the application.

Potential security risks may be a consideration. The user's plain-text credentials are being passed around the network and are resident in the client web browser. This means they could be intercepted or cached in a variety of ways. Encrypting all the involved network traffic via SSL helps to eliminate these interception vulnerabilities, such as those of network sniffing or intermediate proxy caching, but it does not eliminate the use of the plain-text credentials within the browser. The portal response conveying the embedded credentials has all the appropriate no-caching headers associated with it, but some browsers (notably Internet Explorer 6) do not properly honor these headers. The browser can expose these credentials in the source of the SSO Gateway Portlet web content and can cache them on the local disk. Use of a publicly-accessible computer could result in the user's plain-text credentials being stolen through use of the "back button" to view previously accessed content or by sifting through the local browser cache where the data may

be stored for an extended period of time. This risk is especially troubling in a “kiosk computer” environment where each user does not log in as and use his or her own user account, but it can also be an issue in a multiple-account environment where computer is configured such that users can access one another’s browser caches. Having the credentials cached on disk at all can result in computer administrators being able to recover the passwords from cache files.

When providing single sign-on access to web-based applications without any modifications to those applications, the browser uses the user’s plain-text credentials. The Gateway portlet therefore conveys these plain-text credentials to the browser. Measures have been taken to reduce the propensity of browsers to cache these credentials, but the caching behavior is not eliminated by default by these measures.

Providing the credentials to the browser via an AJAX callback rather than directly in the web response

While we have taken measures to minimize the environments in which the standard Gateway SSO Portlet configuration will result in web browsers caching credentials to disk, with the help of one of our clients we have implemented an optional further measure for discouraging browsers from caching credentials.

New in Academus 2.0.6 is an optional configuration of the Gateway SSO Portlet wherein it will supply credentials to the browser via an asynchronous JavaScript callback to the server. This request and response is configured such that some browsers that will cache the standard response will not cache this response.

A given Gateway portlet configuration can opt into this mode of provisioning the browser with credentials by adding a simple element to the gateway portlet configuration file.

This is fully discussed later in this file where the gateway portlet configuration language is detailed.

Note that since the credentials are passing through the browser, there is no guarantee that the browser will not cache them. While this solution has been tested against a variety of presently available web browsers and verified to result in the browsers storing the password only in memory, it is possible that particular browsers now or in the future will cache even these AJAX callback responses.

Alternatives to using the Gateway SSO Portlet

The Academus Gateway SSO Portlet is designed to increase end user authentication experience convenience in linking from Academus to target applications without requiring modifications to those applications. Measures can be taken to limit the exposure of the plain-text credentials to caching. However, these approaches all involve some amount of storing and forwarding end user credentials. Moving beyond the store-and-forward approach to achieve greater single sign on usability and an improved risk profile involves adopting a more robust single sign-on framework and modifying target applications to participate in the chosen single sign on framework. The Central Authentication Service (CAS), Pubcookie, and Shibboleth are all popular options in higher education that provide robust single sign-on capabilities with rigorous security measures. The downside of these frameworks is that they are more complex to setup and manage and do require some level of integration with each target application.

Tradeoffs of using the Gateway SSO Portlet

Using the Gateway SSO Portlet entails making certain tradeoffs of adopting a store-and-forward authentication brokering technology to achieve increased user convenience. The advantages are those of increased user convenience. The disadvantages are those of storing and forwarding end user passwords. This tradeoff should be carefully considered on a case by case basis; the Gateway SSO Portlet is not appropriate for all situations.

2.0 Configuration

This section describes each section of a SSO Gateway Portlet configuration file.

An example of a complete configuration file can be found in section 7.1 section by section. In addition, a well documented example configuration file is provided with your Academus deployment; this file makes a great starting point for creating your own new single sign-on integrations. The file is named **gateway-portlet.xml** and can be found in the directory:

```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-  
a/webapps/AcademusApps/WEB-INF/classes/config/
```

2.1 The <gateway> Element

The top-level element of every SSO Gateway Portlet configuration file is the <gateway> element. This element contains all other elements, and contains one attribute, "peephole".

2.1.1 Example

```
<gateway peephole="gateway_main">  
  <title>My SSO Gateway Portlet</title>  
  <!-- ... -->  
</gateway>
```

2.1.2 Attributes of the <gateway> element

This section describes the known attributes of the <gateway> element.

2.1.2.1 The "peephole" attribute

This attribute is used to define the screen used to render the SSO Gateway Portlet. There is currently only one accepted value for this attribute: gateway_main

2.1.3 Minor sub-elements of the <gateway> element

This section describes the minor sub-elements of the <gateway> element. Minor sub-elements are defined as elements containing simple values.

2.1.3.1 The <title> element

This element defines the title to be shown at the top of the SSO Gateway Portlet.

2.1.3.2 The <ajax-callback-url> element

This optional element became available in Academus XXXX.

This element defines the relative URL that JavaScript running in the web browser can call back to obtain the credentials for signing the user onto the target system. This element is optional. If present, it should have the simple value "/portal/ssoCallback", as in

```
<gateway peephole="gateway_main">  
  <title>My SSO Gateway Portlet</title>
```

```
<ajax-callback-url>/portal/ssoCallback</ajax-callback-url>
</gateway>
```

When this element is present, the portlet will provision credentials into the browser via AJAX callback. When this element is not present, the portlet will provision credentials directly into the response to the browser web request for the page containing the gateway portlet. This latter is the default and legacy behavior and was the only configuration available prior to Academus 2.0.6.

If this element is present, it introduces the requirement that the callback servlet actually be mapped in web.xml and that the web container (Apache or IIS, e.g.) be configured to pass through requests for that URL to the Academus application. This is discussed in the Appendix.

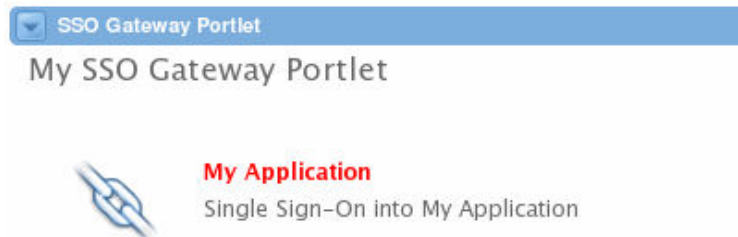
2.2 The <sso-entry> Element

The second-level element of every SSO Gateway Portlet configuration file is a <sso-entry> element. Each <sso-entry> element defines a single external site SSO integration. A SSO Gateway Portlet configuration file can define 1 or more <sso-entry> elements.

If only one <sso-entry> element exists, and is of <window> type *iframe*, then that sso entry will be immediately shown, rather than the standard list format.

2.2.1 Example

```
<sso-entry handle="MyApplication" class="largeLink">
  <label>My Application</label>
  <description>Single Sign-On into My Application</description>
  <!-- ... -->
</sso-entry>
```



2.2.2 Attributes of the <sso-entry> element

This section describes the known attributes of the <sso-entry> element.

2.2.2.1 The “handle” attribute

This attribute is used to define the internally used handle to reference this <sso-entry> element. The value of this attribute must be unique within this SSO Gateway Portlet configuration file.

2.2.2.2 The “class” attribute

This attribute is used to declare which CSS class should be associated with the links created in the SSO Gateway Portlet. This allows the image shown next to the application link to be changed. Please see section 5.0 below for instructions on creating your own icon classes.

2.2.3 Minor sub-elements of the <sso-entry> element

This section describes the minor sub-elements of the <sso-entry> element. Minor sub-elements are defined as elements containing simple values.

2.2.3.1 The <label> element

This element defines the human-readable label to be associated with this <sso-entry>. The value of this element is shown in the list mode of the SSO Gateway Portlet, and also at the top of the SSO Gateway Portlet when clicked into an *iframe* sso-entry.

2.2.3.2 The <description> element

This element defines the description to be associated with this <sso-entry>. The value of this element is shown in the list mode of the SSO Gateway Portlet, just beneath the label.

2.3 The <target> Element

The first element under a <sso-entry> element is <target>. This element and everything it contains defines the request that will be sent when the <sso-entry> is selected.

The parameter value and URL sub-elements support a token substitution mechanism, allowing for dynamic replacement based on the current user's attributes (as dictated by the list at the bottom of portlet.xml), and any attributes supplied by the <authentication> module (see section 2.7 for details). A tokenized string is in the form of { **somekey** }, and may be placed anywhere within a string (it does not have to be the entire value, for instance).

2.3.1 Example

```
<target handle="login">
  <url>http://www.mycompany.com/MyApplication/login</url>
  <method>POST</method>
  <parameter name="username"><value>{user.login.id}</value></parameter>
  <parameter name="password"><value>{password}</value></parameter>
</target>
```

2.3.2 Attributes of the <target> element

This section describes the known attributes of the <target> element.

2.3.2.1 The “handle” attribute

This attribute is used to define the handle used to reference this <target> element. The value of this attribute must be unique within this <sso-entry>.

2.3.3 Minor sub-elements of the <target> element

This section describes the minor sub-elements of the <target> element. Each of these sub-elements, except for <method>, allows the value substitution syntax described in section 2.3.

2.3.3.1 The <url> element

This element defines the Uniform Resource Locator (URL) for the SSO integration. If the application utilizes HTTP-Basic authentication, the username and password substitutions should be specified here in the form of:

```
http://{auth.username}:{auth.password}@mysite.com/
```

Replace the tokens {auth.username} and {auth.password} with the appropriate tokens in your configuration.

2.3.3.2 The <method> element

This element defines the method that should be used to submit the request. This will most likely be “GET” or “POST”. This element does not support value substitution.

2.3.3.3 The <parameter> element

This element defines a single parameter to be sent in the request to the specified URL. This element must contain a single attribute “name”. The value of this attribute is the name of the parameter to be sent.

In addition, this element must contain a sub-element named <value>. The contents of the <value> element will contain the value of the named parameter to be sent with the request.

Zero or more <parameter> elements may be specified per <target> element. The contents of the <value> element allow the value substitution syntax described in section 2.3. The parameter “name” attribute does not support substitution.

2.4 The <sequence> Element

The <sequence> element defines the order in which the <target> elements should be executed for a given action. There are two special sequence types, though you may define as many as you’d like. These two are: login and refresh. A type “login” sequence specifies the initial sequence to use for an <ssso-entry>. A type “refresh” sequence specifies the sequence to use for all subsequent requests following the “login” sequence.

2.4.1 Example

```
<sequence type="login">
  <target handle="login" />
</sequence>
<sequence type="refresh">
  <target handle="baseUrl" />
</sequence>
```

2.4.2 Attributes of the <sequence> element

This section describes the known attributes of the <sequence> element.

2.4.2.1 The “type” attribute

This attribute defines the sequence’s type. The only specially handled values for this attribute are “login” and “refresh”. All other sequences can be defined with arbitrary names unique to this <ss0-entry> element.

2.4.3 Minor sub-elements of the <sequence> element

This section describes the minor sub-elements of the <sequence> element.

2.4.3.1 The <target> element

This element defines a reference to a previously defined <target> (see section 2.3above). This element **must** define a ‘handle’ attribute that matches an existing <target> handle within this <ss0-entry>. It shall not have any sub-elements.

2.4.4 Major sub-elements of the <sequence> element

This section describes the more complex sub-elements of the <sequence> element.

2.4.4.1 The <sequence-start-trigger> element

See section 2.5below.

2.5 The <sequence-start-trigger> element

The <sequence-start-trigger> element is a sub-element of a <sequence>. It is not required, but it provides the ability to offer a “button image” or “link” to the user to initiate another sequence. One example of its use might be a “Re-login” link, which could be useful in the case of a session timeout or similar problem.

2.5.1 Example

```
<sequence-start-trigger>
  <context type="popup">
    <image src="images/icons/marker_error.gif"
      class="smallImage" height="16" width="16"
      border="0" align="middle"/>
    <text class="sakaiSmall">
      <label>Log in again</label>
      <description>ReAuthenticate description</description>
    </text>
  </context>
</sequence-start-trigger>
```

2.5.2 The <context> sub-element

The <context> element declares the image and/or text to use for a given context. There are two context-types: popup and iframe.

2.5.3 The <image> sub-element

The <image> element defines the source, class, height, width and other stylistic elements of an image to be used as a link to start the sequence. The attributes on the image are as defined by the HTML 4.01 specification for the tag.

2.5.4 The <text> sub-element

The <text> element defines an optional label and description of the sequence start trigger. This text will show up alongside any defined image or by itself if no image is defined.

2.6 The <window> Element

The <window> element defines the style that the integration should take. There are two options: I-Frame, or Pop-up. I-Frame opens up in an inline-frame, which means that it appears within the same browser window similar to the way a portlet appears. Pop-up type entries will open in a new browser window.

2.6.1 Example

2.6.1.1 I-Frame Example

```
<window type="iframe">
  <title>My Application</title>
  <name>MyApplication</name>
  <style>width: 100%; height: 450px; border: 0px;</style>
</window>
```

2.6.1.2 Pop-up Example

```
<window type="popup">
  <title>My Application</title>
  <name>MyApplication</name>
  <properties></properties>
</window>
```

2.6.2 Attributes of the <window> element

This section describes the known attributes of the <window> element.

2.6.2.1 The “type” attribute

This attribute is used to define which type of rendering should be used for the target. The two accepted values are “iframe” and “popup”.

2.6.3 Minor sub-elements of the <window> element

This section describes the minor sub-elements of the <window> element.

2.6.3.1 The <title> element

The <title> element defines a human readable name for the window or iframe.

2.6.3.2 The <name> element

The browser-internal name of the window or iframe used to render the target. This value must be unique within the same portal page; in general, it is safest to make this value unique across all SSO Gateway Portlet configuration files.

NOTE: Window names should only contain alphanumeric or underscore (_) characters. Other special characters may cause problems in some browsers that break certain portal functionality (like Single Sign Off).

2.6.3.3 The <style> element

The CSS styles attached to the iframe. This element is only valid when using window type iframe.

2.6.3.4 The <properties> element

The contents of this element are the properties to specify on new browser windows. This element is only valid when using the window type popup.

2.7 The <authentication> Element

The <authentication> element defines an authentication handler. Authentication handlers can be specified to provide more robust mapping of user attributes to a set of credentials for an external system. The primary use for this would be single sign-on for an external system that uses a different set of credentials than the Academus portal. This element is optional, and should only be specified if you require such an authentication module.

Currently, only one such authentication handler exists: SsoMultiAuthentication.

2.7.1 SsoMultiAuthentication

The SsoMultiAuthentication handler offers a mapping of a username and site key to a set of credentials. These credentials can be entered and later modified interactively by the user, but will be stored in an encrypted format in the database to allow for them to be used in single sign-on in all subsequent logins.

SsoMultiAuthentication will provide two additional attributes available for use as substitution tokens in the <target> element. These are **{auth.username}** and **{auth.password}**, corresponding to the stored username and password respectively.

In order to enable the SsoMultiAuthentication authentication handler, the following XML fragment is needed:

```
<authentication handler="net.unicon.academus.apps.SsoMultiAuthentication">
  <parameter name="system-key">
    <value>@APPLICATION@</value>
  </parameter>
  <parameter name="username-key">
    <value>user.login.id</value>
  </parameter>
  <parameter name="encrypt-ref-key">
    <value>cromagnum</value>
  </parameter>
</authentication>
```

```
<parameter name="jndi-ref">
  <value>java:comp/env/jdbc/PortalDb</value>
</parameter>
</authentication>
```

The token **@APPLICATION@** should be replaced with the name of the application you are configuring for SSO. It must be distinct across all SSO Gateway Portlet configurations unless you wish to share stored credentials with another configuration. All other values should not have to be modified.

2.8 The Evaluator Element:

Some cases the WebAdvisor application may require to authenticate against a source that requires the username to be in all lower case, where as the portal authentication source requires mixed case. For this scenario a customization to the Gateway Portlet has been developed to manipulate user parameter values at runtime.

Any custom evaluator can be developed as long as it implements the `IAttributeEvaluator` interface.

To use the `LowerCaseAttributeEvaluator`, an xml element, 'evaluator', must be added to the Gateway xml configuration document with an `evaluatorImpl` attribute that has the `IAttributeEvaluator` implementation as the value. Please see below for example:

```
<target handle="initial">
  <url>https://adminws.wlu.edu/servlet/com.datatel.server.servlets.webadvisor.WebAdvisor?ACTION=Login</url>
  <method>GET</method>
  <parameter name="USERID"><evaluate evaluatorImpl="net.unicon.academus.apps.gateway.util.LowerCaseAttributeEvaluator">
    <value>{user.login.id}</value></evaluate></parameter>
  <parameter name="PASSWORD"><value>{password}</value></parameter>
</target>
```

3.0 Declaring Gateway Portlets

3.1 The purpose of this section

This section is intended to support deployers in declaring new instances of the briefcase portlet, discussing how portlets are declared and configured in Academus and drawing attention to the examples included with Academus and the key elements of the configuration. A full treatment of the declaration and lifecycle of JSR-168 portlets is out of scope for this document. Please contact customer support with questions not answered by this document.

3.2 Declaring portlets in general

JSR-168 portlets exist as web applications. They live in the Tomcat webapps directory and may have a web.xml deployment descriptor and must have a portlet.xml deployment descriptor. In portlet.xml, there can be one or many declarations of portlets, where each declaration might declare a different underlying portlet implementation (Java class) or might declare the same underlying portlet implementation several times with different configuration.

A portlet declaration includes a name for the portlet, the name of the Java class of the portlet, zero or more portlet initialization parameters, caching information, localization information, general information about the portlet, and supported mime types.

In general, when you re-use an existing portlet with new configuration, the way this is accomplished is by adding another declaration of the portlet in portlet.xml with different configuration. For example, this fragment of XML declares an instance of the Gateway Portlet:

```
<portlet>
  <portlet-name>GatewayPortlet</portlet-name>
  <portlet-class>net.unicon.academus.apps.gateway.GatewayPortlet</portlet-class>
  <init-param>
    <name>Id</name>
    <value>cromagnum</value>
  </init-param>
  <init-param>
    <name>configPath</name>
    <value>/WEB-INF/classes/config/gateway-portlet.xml</value>
  </init-param>
  <init-param>
    <name>userContextTimer</name>
    <value>60</value>
  </init-param>
  <expiration-cache>0</expiration-cache>
  <supported-locale>en-US</supported-locale>
  <portlet-info>
    <title>GatewayPortlet</title>
    <short-title>Gateway</short-title>
    <keywords>Gateway</keywords>
  </portlet-info>
  <supports>
```

```
<mime-type>text/html</mime-type>
</supports>
</portlet>
```

3.3 Declaring AcademusApps portlets

The configuration file for the Academus Apps portlets exists in the following location:

```
/portal/unicon/Academus/portal-tomcat-a/webapps/AcademusApps/WEB-INF/portlet.xml
```

This portlet.xml file declares several different portlets, including the Briefcase Portlet and the Gateway Portlet.

Some portlets are multiply declared, indicating that there are multiple instances of them. Not all portlets support multiple declarations, but the Gateway Portlet does support multiple declarations and multiply declaring it is the common way it is used. That is, to bring about a new usage of the gateway portlet, a deployer adds a new declaration of the portlet in portlet.xml.

3.4 Declaring new instances of the Academus Gateway SSO Portlet

The purpose of this section is to describe in detail exactly what is required for Gateway Portlet declarations in Academus Apps portlet.xml.

Again referring to portlet deployment descriptor for the Academus Apps web application:

```
/portal/unicon/Academus/portal-tomcat-a/webapps/AcademusApps/WEB-INF/portlet.xml
```

This section will address each element in detail.

3.4.1 The <portlet/> element wrapper

To add a new declaration of the Gateway Portlet, add a new <portlet> </portlet> element to portlet.xml in parallel to the existing portlet elements.

3.4.2 The <portlet-name/> element

Give the new declaration of the gateway portlet a name that is not yet used as a <portlet-name/> in portlet.xml. This name should be a short meaningful name for the usage of the gateway portlet and may be included in logging and error messages associated with this usage.

```
<portlet-name>SakaiGateway</portlet-name>
```

3.4.3 The <portlet-class/> element

This element declares the Java class of the underlying Java code implementing the portlet. For the briefcase portlet, this will always be

```
<portlet-class>net.unicon.academus.apps.gateway.GatewayPortlet</portlet-class>.
```

3.4.4 The “id” <init-param/> element

There are three required portlet initialization parameters for the Gateway Portlet. The first of these is the “id”. The value of this initialization parameter of each gateway portlet declaration must be unique. Therefore,

when you create a new gateway portlet declaration, give it a new “id” value. This is arbitrary so long as it is unique, but it may be helpful to use a name that reflects the usage of the portlet.

```
<init-param>
  <name>Id</name>
  <value>sakaigateway</value>
</init-param>
```

3.4.5 The “configPath” <init-param/> element

The second required initialization parameter for the Gateway Portlet is the path to the gateway portlet configuration file. The portlet.xml configuration is the bridge to this further configuration.

```
<init-param>
  <name>configPath</name>
  <value>/WEB-INF/classes/config/gateway-portlet-sakai.xml</value>
</init-param>
```

3.4.6 The “userContextTimer” <init-param/> element

The third required initialization parameter for the Gateway Portlet is the number of seconds the portlet should retain the user context available for browser client callback. This parameter did not exist prior to Academus 2.0.6; in Academus 2.0.6 and later this parameter became required.

```
<init-param>
  <name>userContextTimer</name>
  <value>60</value>
</init-param>
```

3.4.7 The <expiration-cache/> element

This element configures how long the portal infrastructure should cache the output of the portlet before re-rendering it. It should be set to zero seconds, indicating that the portal framework will not cache the output of this portlet.

```
<expiration-cache>0</expiration-cache>
```

3.4.8 The “<supported-locale/> element

The only locale currently supported by the Gateway Portlet is “en-US”.

```
<supported-locale>en-US</supported-locale>
```

3.4.9 The “<portlet-info/> compound element

The <portlet-info/> compound element provides three items of metadata about the portlet. This includes a title and a shortened form of the title as well as keywords for the portlet. The title and shorted title should reflect the usage of the portlet. The keywords element does not currently have an effect but may in the future be used to facilitate searching for available portlets.

```
<portlet-info>
  <title>SakaiGatewayPortlet</title>
  <short-title>SakaiGateway</short-title>
```

```
<keywords>Sakai, Gateway, SSO</keywords>  
</portlet-info>
```

3.4.10 The “<supports/> compound element

The supports element declares the supported mime-types. Currently the Gateway portlet only supports text/html.

```
<supports>  
  <mime-type>text/html</mime-type>  
</supports>
```

3.5 Wait, why do I have to configure gateway portlets in multiple places?

There are two configuration locations for two different kinds of configuration. Portlet.xml declares how many different configurations of the Gateway SSO Portlet are to be used and where to find their additional configuration. This is in the language of portlet declarations. The individual gateway portlet configuration files declare in detail exactly how particular Gateway SSO portlet usages should behave. This is in the language of gateway portlet configuration.

4.0 Finding Parameters for Your Application

This section describes how you can find the correct parameters to use in configuring a new SSO integration with the SSO Gateway Portlet and your web application.

The first step is to identify the target URL. This will most likely be found on the 'Login' page of your application, or equivalent. Navigate to the "Login" page of your application, and use your browser's "View Source" function.

In the page source, you must locate the `<form>` tag corresponding to the login form shown on the screen. This `<form>` tag and its contents will tell you everything you need to know to configure your new SSO integration. The `<method>` will be found in the "method" attribute of the form. The target `<url>` will be the value of the "action" attribute.

From within the `<form>` tag, you will need to locate any `<input>` tags. These will be the `<parameter>`s to use in your SSO configuration's `<target>`; the "name" attribute of the `<input>` tag will map directly to the "name" attribute on the corresponding `<parameter>` tag. The value for these parameters will require consideration; some of the values may be static strings that the application expects, and others may be the dynamic such as the username and password.

5.0 Creating New Application Icons

This section describes how to create new icons that can be displayed next to your application links in the SSO Gateway Portlet.

5.1 Create the icon image

The first step in adding a displayable application icon is to create the icon. This can be done using any image editing software; a few common tools are Adobe Photoshop, Jasc Paintshop Pro and GIMP. The icon should be saved in a GIF or PNG format with dimensions of 56x56 pixels or smaller.

Save this icon image file in the following directory:

```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-  
a/webapps/AcademusApps/rendering/images/
```

5.2 Make the icon image accessible as a CSS class

After having created your icon image, you need to make it accessible as a CSS class. This can be done by modifying the file:

```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-  
a/webapps/AcademusApps/rendering/css/config_icons.css
```

5.2.1 Pick a CSS class name

The first step in modifying the file is to pick a CSS class name to use. The naming convention `large@APPLICATION@` is recommended.

5.2.2 Create the new CSS class

In the `config_icons.css` file, you need to add your new class name to three places.

1. Add the class, prefaced with a "." in standard CSS syntax notation, to the comma-separated list of classes on line 4.
2. Add a new section with the contents:

```
.large@APPLICATION@ {  
background: url("../images/icons/@ICONFILENAME@") no-repeat;  
}
```
3. Add the class, prefaced with "span.", to the comma separated list of classes near the bottom of the file. The block should already have the contents "display: none".

6.0 Creating an instance of the SSO Gateway Portlet

6.1 Create the SSO Gateway Portlet Configuration File

The first step in creating your new SSO Gateway Portlet instance is to create the SSO Gateway Portlet configuration file it will use. This file should be created in the directory:

```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-  
a/webapps/AcademusApps/WEB-INF/classes/config/
```

The preferred naming scheme for SSO Gateway Portlet configuration files is **gateway-@APPLICATION@.xml**, replacing the token **@APPLICATION@** as appropriate. Be sure to remember this file name for step 6.3 below.

The contents of this file are described in section 2.0.

6.2 Add the New Portlet to the Application's web.xml File

You will need to put a new servlet definition and mapping in the 'web.xml' file for AcademusApps. This file can be found at:

```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-  
a/webapps/AcademusApps/WEB-INF/web.xml
```

This is an XML file, and as such it can be adjusted in any text editor. Common text editors include Notepad, TextPad, and Vi.

Add the `<servlet>` element as follows:

```
<servlet>  
  <servlet-name>@APPLICATION@Gateway</servlet-name>  
  <display-name>@APPLICATION@Gateway Wrapper</display-name>  
  <description>Portlet Wrapper</description>  
  <servlet-class>org.apache.pluto.core.PortletServlet</servlet-class>  
  <init-param>  
    <param-name>portlet-class</param-name>  
    <param-value>net.unicon.academus.apps.gateway.GatewayPortlet</param-  
value>  
  </init-param>  
  <init-param>  
    <param-name>portlet-guid</param-name>  
    <param-value>AcademusApps.@APPLICATION@Gateway</param-value>  
  </init-param>  
</servlet>
```

Add the `<servlet-mapping>` element as follows:

```
<servlet-mapping>
```

```

    <servlet-name>@APPLICATION@Gateway</servlet-name>
    <url-pattern>/@APPLICATION@Gateway/*</url-pattern>
</servlet-mapping>

```

It is extremely important that all instances of “@APPLICATION@Gateway” match exactly in value. Please be sure to replace all instances of the token @APPLICATION@ with the desired application name.

6.3 Add the New Portlet to the Application’s portlet.xml File

You will need to put a new portlet definition in the ‘portlet.xml’ file for AcademusApps. This file can be found at:

```

<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-
a/webapps/AcademusApps/WEB-INF/portlet.xml

```

This is an XML file, and as such it can be adjusted in any text editor. Common text editors include Notepad, TextPad, and Vi.

Add a new <portlet> element as follows:

```

<portlet>
  <portlet-name>@APPLICATION@Gateway</portlet-name>
  <portlet-class>
    net.unicon.academus.apps.gateway.GatewayPortlet
  </portlet-class>
  <init-param>
    <name>Id</name>
    <value>@APPLICATION@</value>
  </init-param>
  <init-param>
    <name>configPath</name>
    <value>/WEB-INF/classes/config/@CONFIG_FILE_NAME@</value>
  </init-param>
  <expiration-cache>0</expiration-cache>
  <supported-locale>en-US</supported-locale>
  <portlet-info>
    <title>@APPLICATION@</title>
    <short-title>@APPLICATION@</short-title>
    <keywords>@APPLICATION@</keywords>
  </portlet-info>
</portlet>

```

Be sure to replace @APPLICATION@ with the application’s name, and @CONFIG_FILE_NAME@ with the name of the file you created in step 6.1. The <portlet-name> element’s value must match the value used in section 6.2’s configuration.

6.4 Restart the Portal

You will need to restart the Academus Portal in order for your new portlet to be available.

6.5 Publish the Portlet

You will need to publish the new Portlet in Academus before users can add it to their layouts. You can use the following values for publishing parameters:

Channel Type	Portlet
Channel Title	@APPLICATION@
Channel Name	@APPLICATION@
Channel Functional Name	@APPLICATION@Gateway
Channel Description	<description at your discretion>
Channel Timeout	5000 milliseconds
Channel Secure	Unchecked
Portlet Definition ID	AcademusApps.@APPLICATION@Gateway
Channel Controls	All unchecked
Selected Categories	At your discretion
Selected Groups	At your discretion

The value for the Portlet Definition ID must match the value used in the 'portlet-guid' init-param of section 6.2.

7.0 Appendix

7.1 SSO Gateway Portlet Configuration Example

Below you can find a simple example of a SSO Gateway Portlet configuration file.

```
<gateway peephole="gateway_main">
  <title>Gateway Portlet Test</title>
  <ajax-callback-url>/portal/ssoCallback</ajax-callback-url>
  <sso-entry handle="sso-test" class="largeLink">
    <label>SSO Test</label>
    <description>Single Sign On Test Page</description>
    <target handle="login">
      <url>@WEBSERVER_IS_SSL@://@HOSTNAME@@WEBSERVER_FULLHTTPPORT@/AcademusApps/rendering/jsp/sso-test.jsp</url>
      <method>POST</method>
      <parameter name="portalUsername">
        <value>{user.login.id}</value>
      </parameter>
      <parameter name="authUsername">
        <value>{auth.username}</value>
      </parameter>
      <parameter name="authPassword">
        <value>{auth.password}</value>
      </parameter>
      <parameter name="someString">
        <value>This is my username: {user.login.id}</value>
      </parameter>
    </target>
    <sequence type="login"><target handle="login"/></sequence>
    <window type="iframe">
      <title>SSO Test</title>
      <name>SSO Test</name>
      <style>width: 100%; height: 250px; border: 0px;</style>
    </window>
    <authentication handler="net.unicon.academus.apps.SsoMultiAuthentication">
      <parameter name="system-key"><value>sso-test</value></parameter>
      <parameter name="username-key"><value>user.login.id</value></parameter>
      <parameter name="encrypt-ref-key"><value>cromagnum</value></parameter>
      <parameter name="jndi-ref">
        <value>java:comp/env/jdbc/PortalDb</value>
      </parameter>
    </authentication>
  </sso-entry>
  <sso-entry handle="google" class="largeLink">
    <label>Google</label>
    <description>Google for your username.</description>
    <target>
      <url>http://www.google.com/search</url>
      <method>GET</method>
    </target>
  </sso-entry>
</gateway>
```

```

<parameter name="hl"><value>en</value></parameter>
<parameter name="btnG"><value>Google Search</value></parameter>
<parameter name="q"><value>{user.login.id}</value></parameter>
</target>

<window type="iframe">
  <title>Google too</title>
  <name>Google too</name>
  <style>width:100%;height:200px;border:0px;</style>
</window>
</sso-entry>
<jndi-ref>java:comp/env/jdbc/PortalDb</jndi-ref>
</gateway>

```

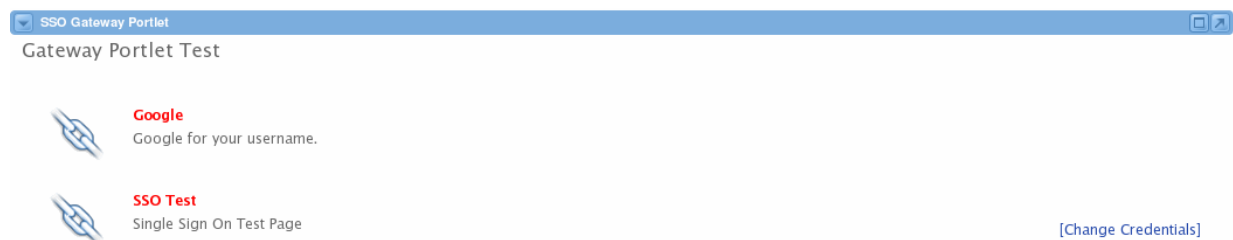
This configuration creates an SSO into a testing page that came with your Academus deployment. It utilizes both your portal login credentials and the idea of mapped credentials to demonstrate the full capabilities of the SSO Gateway Portlet.

In addition, a well documented example configuration file is provided with your Academus deployment; this file makes a great starting point for creating your own new single sign-on integrations. The file is named **gateway-portlet.xml** and can be found in the directory:

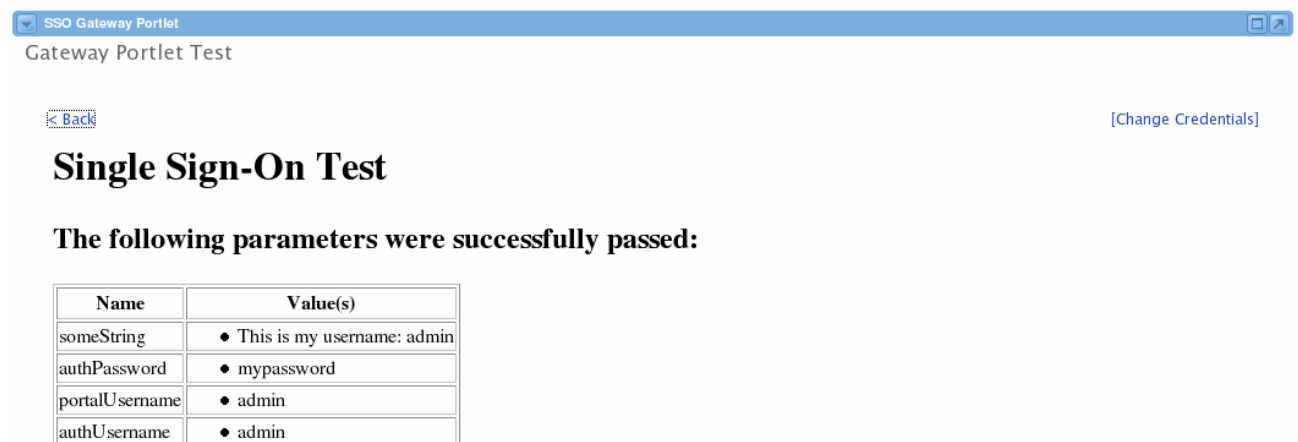
```
<ACADEMUS_UNICON_INSTALL_ROOT>/unicon/Academus/portal-tomcat-
a/webapps/AcademusApps/WEB-INF/classes/config/
```

7.2 SSO Gateway Portlet Example Configuration in Action

7.2.1 Main View: Links



7.2.2 Focused View: The SSO Test Application



7.3 Additional requirements in configuring the AJAX callback

This section documents the requirements for successful operation of the AJAX callback mechanism for supplying credentials to the web browser. In most cases your Academus instance should already include this configuration as deployed or as patched via the upgrade process to Academus 2.0.6. This section is included for completeness and to aid in those cases where customizations are made to these configuration files.

When the Gateway SSO Portlet is configured to use the AJAX callback mechanism for supplying credentials to the web browser, it includes in its response to web browsers JavaScript which executes a callback to the Academus application to obtain the credentials. This callback mechanism discourages web browsers from caching these credentials to disk.

The callback is to the Academus portal web application itself (and not to the Gateway portlet web application, which exists as its own web application under Tomcat as it is a JSR-168 portlet). Therefore the callback servlet must be mapped in `/webapps/portal/WEB-INF/web.xml`

7.3.1 Configuring the callback servlet in Academus Portal web.xml

Among the `<servlet>` declarations in `web.xml`, the gateway callback servlet must be declared:

```
<servlet>
  <servlet-name>SsoCallback</servlet-name>
  <servlet-class>net.unicon.academus.apps.sso.SsoCallbackServlet</servlet-
class>
  <load-on-startup>4</load-on-startup>
  <init-param>
    <!-- This path should map back to the Login servlet -->
    <param-name>unauthenticatedRedirectUrl</param-name>
    <param-value>/Login</param-value>
  </init-param>
</servlet>
```

Among the `<servlet-mapping>` declarations in `web.xml`, the gateway callback servlet must be mapped:

```
<servlet-mapping>
  <servlet-name>SsoCallback</servlet-name>
  <url-pattern>/ssoCallback</url-pattern>
</servlet-mapping>
```

7.3.2 Configuring the callback servlet in the web container

Additionally, the web container (IIS or Apache, e.g.) must be configured dispatch requests for the callback URL from the web container to the Tomcat, which will in turn dispatch them to the Academus portal web application and the callback servlet as configured above. How the web container is configured to map this request varies by particular web container and its configuration.

An Apache 2.0.xx server might use the following JkMount:

```
JkMount /portal/ssocallback
```

On an IIS system, Tomcat would mount the worker as follows in uriworkermap.properties:

```
/portal/ssocallback=worker1
```