



# The Briefcase Portlet Configuration Guide

# Contents

---

<b>1.0 Introduction</b> .....	<b>4</b>
1.1 What's new in the Briefcase Portlet? .....	4
1.2 What is a Portlet? .....	4
<b>2.0 Configuration</b> .....	<b>5</b>
2.1 Standard Configuration: The Big Picture .....	5
2.2 Standard Configuration: Broken Down.....	7
2.2.1 Url encryption .....	11
2.3 Shared Folders .....	11
2.4 Shared Network Space .....	11
2.5 Personal Network Space .....	12
2.5.1 File system access requirements.....	13
<b>3.0 Briefcase.config</b> .....	<b>14</b>

# 1.0 Introduction

## 1.1 What's new in the Briefcase Portlet?

Included with the Academus 1.5 release was a new briefcase channel. This channel was built upon the portlet framework and includes many enhancements over the existing briefcase channel. The briefcase portlet provides finer-grained access control, more configuration options for personal space (the ability to set the briefcase size and parameters based on group membership, etc), and the ability to mount personal and shared network space within Academus.

The briefcase portlet has its own configuration file that describes its behavior. The file is specified in XML and contains documentation on what the individual XML flags do—more of a reference than a step-by-step guide. This document will attempt to fill in the blanks that the documentation contained within the configuration file leaves and give you some good starting points for implementing your briefcase solution. This document is meant to be more of an introductory tutorial rather than a reference manual. For help with individual flags and settings please consult the configuration file (a copy of this file has been included at the end of this manual for your convenience).

## 1.2 What is a Portlet?

Portlet channels have a few fundamental differences from regular portal channels. Portlets exist within their own web application container outside of the Academus web application. For example, the Academus web application exists in the following location:

```
/portal/unicon/Academus/portal-tomcat-a/webapps/portal
```

The briefcase portlet exists here:

```
/portal/unicon/Academus/portal-tomcat-a/webapps/AcademusApps
```

The details of portlet implementation are beyond the scope of this document; the important thing to remember is that the briefcase portlet is its own web application and exists in a directory outside of Academus (specifically the AcademusApps directory).

## 2.0 Configuration

### 2.1 Standard Configuration: The Big Picture

The configuration file for the briefcase portlet exists in the following location:

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

We will start by looking at the configuration that comes standard with Academus for private space:

```
<drive handle="personal" max-upload="3145728" large-icon="largePersonal" open-  
icon="folderopen" closed-icon="folderclosed" share-target="shared">  
  <label>Personal Folders</label>  
  <description>Your Personal Folders are where you can store your  
documents, notes, photos, and more.</description>  
  <access-broker handle="personal-jit"  
impl="net.unicon.alchemist.access.permissions.PermissionsAccessBroker">  
    <access  
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>  
  
    <targets>  
      <access-broker handle="personal-resources-t"  
impl="net.unicon.alchemist.access.jit.JitAccessBroker">  
        <jit-rule>  
          <behavior>  
            <trigger type="GROUP">Everyone</trigger>  
            <target type="USER" />  
            <access  
impl="net.unicon.alchemist.access.permissions.DummyAccessType">  
              <type handle="DUMMY" value="GRANT"/>  
            </access>  
            <creator  
impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">  
              <size>50</size>  
              <root-name>My Files</root-name>  
              <seed-  
path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>  
            </creator>  
          </behavior>  
        </jit-rule>  
  
        <access-broker handle="personal-resources-t-i"  
impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker">  
          <access  
impl="net.unicon.alchemist.access.permissions.DummyAccessType"/>  
        </access-broker>  
      </access-broker>  
    </targets>  
  
  <permissions>
```

```
<access-broker handle="personal-resources-p"
impl="net.unicon.alchemist.access.jit.JitAccessBroker">
  <jit-rule>
    <behavior>
      <trigger type="GROUP">Everyone - Staff</trigger>
      <target type="GROUP">Everyone - Staff</target>
      <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        <type handle="DELETE" value="GRANT"/>
        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
      </access>
      <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
    </behavior>
    <behavior>
      <trigger type="GROUP">Everyone - Students</trigger>
      <target type="GROUP">Everyone - Students</target>
      <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        <type handle="DELETE" value="GRANT"/>
        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
      </access>
      <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
    </behavior>
    <behavior>
      <trigger type="GROUP">Everyone - Staff</trigger>
      <target type="GROUP">Everyone - Staff</target>
      <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        <type handle="DELETE" value="GRANT"/>
        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
      </access>
      <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
    </behavior>
    <behavior>
      <trigger type="GROUP">Everyone</trigger>
      <target type="GROUP">Everyone</target>
      <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        <type handle="DELETE" value="GRANT"/>
        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
      </access>
```

```
                </access>
                <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
            </behavior>
        </jit-rule>

        <access-broker
            handle="personal-resources-p-i"

impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker"
            needsDataSource="true">
            <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
            </access-broker>
        </access-broker>
    </permissions>

</access-broker>
</drive>
```

## 2.2 Standard Configuration: Broken Down

We'll look at each piece of this configuration and describe how it works within the big picture:

```
<briefcase>
```

This tag is the root tag of the briefcase configuration specification and is needed only once. This tag will encompass the entire briefcase configuration.

```
<drive handle="personal" max-upload="3145728" large-icon="largePersonal" open-
icon="folderopen" closed-icon="folderclosed" share-target="shared">
```

The next piece is the drive tag. This tag specifies general settings for a virtual briefcase 'drive', which can be personal space (as in this example), shared network space, or personal network space. The 'handle' tag specifies a unique label for this drive. Max-upload allows you to configure the maximum filesize in bytes that is allowed for a briefcase upload; in this example, the maximum filesize for upload is 3 megabytes (3\*1024\*1024). The icon specifications allow you to configure the look and feel of each drive to help users visually distinguish between them. Share-target specifies a drive handle which folders in this drive specification may be shared to. In other words, you can specify another drive that's used for accessing shared folders from this drive. This will be explained in the section about shared drives.

```
    <label>Personal Folders</label>
    <description>Your Personal Folders are where you can store your documents,
notes, photos, and more.</description>
```

The label and description tags are only used to display a description of the drive when a user accesses the briefcase. They are not used for anything else.

```
<access-broker handle="personal-jit"
impl="net.unicon.alchemist.access.permissions.PermissionsAccessBroker">
  <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
  <targets>
    ...
  </targets>
  <permissions>
    ...
  </permissions>
</access-broker>
```

Every drive specification needs an access-broker implementation to tell it how to control access and what to control access. In this example, we use the `PermissionsAccessBroker` to control briefcase sizes and permissions based on user groups. The `PermissionsAccessBroker` requires the specification of targets and permissions. The targets specify the target the user is able to access and the permissions specify the activities that the user is permitted on the given target.

```
<targets>
  <access-broker handle="personal-resources-t"
impl="net.unicon.alchemist.access.jit.JitAccessBroker">
    <jit-rule>
      <behavior>
      </behavior>
    ...
  </access-broker>
</targets>
```

Each target requires the specification of an access broker implementation to specify the target that the user will be able to access. In this example, we use the `JitAccessBroker` to specify the access broker. The `JitAccessBroker` requires the specification of a `jit-rule`, which controls the creation of targets in the briefcase. Each `jit-rule` can have multiple behavior tags that specify how to behave based on group membership or username.

```
<trigger type="GROUP">Everyone</trigger>
<target type="USER" />
<access impl="net.unicon.alchemist.access.permissions.DummyAccessType">
  <type handle="DUMMY" value="GRANT"/>
</access>
```

The `trigger` tag allows you to specify a user or group to trigger this particular behavior with. In this example, any member of the group `Everyone` will match this behavior. You could also specify an individual user by setting the type to `"USER"` and inputting the username between the trigger tag specifications. The `target` tag specifies what type of target to trigger this behavior for. This should always be `"USER"`. The access types configure various levels of access to the resources in the user's briefcase. In this example, we give the user `DUMMY` access, since the activities access will be specified in the permissions.

```
<creator impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">
  <size>50</size>
  <root-name>My Files</root-name>
  <seed-path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>
</creator>
```

This piece specifies which file creation implementation to use. FsFactoryCreator creates files on the file system and is probably the desired implementation in most cases. The size tag specifies the maximum size, or quota, that the user is allocated for storage. The root-name tag is displayed to the user as their root directory in the briefcase channel and can be anything. The seed path is the path on the filesystem to store uploaded files. A directory named with a unique id number is created in this location when the briefcase is initialized.

```
<behavior>
  <trigger type="GROUP">Everyone - Staff</trigger>
  <target type="USER" />
  <access impl="net.unicon.alchemist.access.permissions.DummyAccessType">
    <type handle="DUMMY" value="GRANT"/>
  </access>
  <creator impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">
    <size>200</size>
    <root-name>My Files</root-name>
    <seed-path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>
  </creator>
</behavior>
<behavior>
  <trigger type="GROUP">Everyone</trigger>
  <target type="USER" />
  <access impl="net.unicon.alchemist.access.permissions.DummyAccessType">
    <type handle="DUMMY" value="GRANT"/>
  </access>
  <creator impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">
    <size>50</size>
    <root-name>My Files</root-name>
    <seed-path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>
  </creator>
</behavior>
```

It is also possible to specify multiple behaviors. The first behavior that a user matches will be executed and the remaining behaviors will be ignored. In this example, our final behavior is to create a briefcase with a 200MB limit if the user is in the 'Everyone-Staff' group and 50MB limit if the user is in the 'Everyone' group. Every user is a member of the 'Everyone' group by default, so this will match against everybody. However, users may only match one behavior, so any member of the 'Everyone-Staff' group will match the first rule and skip the rest.

```
<permissions>
  <access-broker handle="personal-resources-p"
  impl="net.unicon.alchemist.access.jit.JitAccessBroker">
  <jit-rule>
  <behavior>
  <trigger type="GROUP">Everyone - Staff</trigger>
  <target type="GROUP">Everyone - Staff</target>
```

```

<access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
  <type handle="VIEW" value="GRANT"/>
  <type handle="ADD" value="GRANT"/>
  <type handle="DELETE" value="GRANT"/>
  <type handle="EDIT" value="GRANT"/>
  <type handle="SHARE" value="GRANT"/>
</access>
<creator impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
</behavior>
<behavior>
  <trigger type="GROUP">Everyone</trigger>
  <target type="GROUP">Everyone</target>
  <access
    impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
    <type handle="VIEW" value="GRANT"/>
    <type handle="ADD" value="GRANT"/>
    <type handle="DELETE" value="GRANT"/>
    <type handle="EDIT" value="GRANT"/>
    <type handle="SHARE" value="GRANT"/>
  </access>
  <creator
    impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
</behavior>
</jit-rule>

```

The `<permissions>` section is similar to the `<targets>` sections, in that, it specifies a JitAccessBroker to manage the triggering. The main difference is in the use of the `<access>` tag and the `<creator>`.

```

<access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
  <type handle="VIEW" value="GRANT"/>
  <type handle="ADD" value="GRANT"/>
  <type handle="DELETE" value="GRANT"/>
  <type handle="EDIT" value="GRANT"/>
  <type handle="SHARE" value="GRANT"/>
</access>

```

The access types configure various levels of access to the resources in the user's briefcase. In this example, we are allowing full access to the user. The handle in the access type specifies the type of action and the value specifies whether the permission is granted (GRANT) or denied (DENY). The BriefcaseAccessType specifies five different activities that can be done on the resource, namely, view, add, edit, delete and share (allowed only on folders). You can combine multiple access types to configure the desired level of access.

For example:

```

<access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
  <type handle="VIEW" value="GRANT"/>
  <type handle="ADD" value="GRANT"/>
</access>

```

The above configuration would allow the user to read and add resources but not delete, edit or share them.

```

<creator impl="net.unicon.alchemist.access.permissions.DummyCreator"/>

```

A Dummy Creator is used which does not create any targets, since the permissions block is used only for specifying the access levels on the specified target.

### 2.2.1 Url encryption

For the secure transfer of resource path within the portal, an encryption service is made available.

The `<encryption-instance>` element defines an instance of the encryption service with a given `<name>` and `<key>`. Valid values for the `encrypt` attribute are "true" (resource urls are encrypted, recommended) and "false" (resource urls are not encrypted). The `<name>` of the encryption instance is defined by the application utilizing the service. `<key>` must contain an 8 character string that will act as the key for the encryption. A longer string can be given, but only the first 8 characters will be used.

Example configuration for the encryption service is

```
<encryption-services>
  <encryption-instance encrypt="true">
    <name>monkey</name>
    <key>cafebabe</key>
  </encryption-instance>
  . . .
</encryption-services>
```

## 2.3 Shared Folders

```
<drive handle="shared" large-icon="largeShared" open-icon="sharedfolderopen"
closed-icon="sharedfolderclosed" sharing="off">
  <label>Shared Folders</label>
  <description>Access other files that have been shared with
you.</description>
  <access-broker handle="shared-resources"
impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker">
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
  </access-broker>
</drive>
```

The configuration above is the standard configuration for a shared folder area. The drive handle specified should match the share-target specified in your personal folders configuration. The icons are for visual representation of the purpose of the drive. The sharing setting enables or disables the ability to share folders with other users.

## 2.4 Shared Network Space

```
<drive handle="network" large-icon="largeNetwork" open-icon="networkfolderopen"
closed-icon="networkfolderclosed" sharing="off">
  <label>Network Folders</label>
  <description>The Network Folders contain common folders that you and
others on the Campus can access.</description>
```

```

    <access-broker handle="network-resources"
impl="net.unicon.alchemist.access.jvm.JvmAccessBroker">
    <entry
target="FSA://net.unicon.demetrius.fac.filesystem.FsResourceFactory/[Test]/F:/test">
        <identity type="GROUP">Everyone</identity>
        <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
            <type handle="VIEW" value="GRANT"/>
            <type handle="ADD" value="GRANT"/>
            <type handle="DELETE" value="GRANT"/>
        </access>
    </entry>
    <entry
target="FSA://net.unicon.demetrius.fac.filesystem.FsResourceFactory/[Authoring]/I:/Authoring">
        <identity type="GROUP">Everyone</identity>
        <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
            <type handle="VIEW" value="GRANT"/>
        </access>
    </entry>
</access-broker>
</drive>

```

The above configuration is a good starting point for setting up your shared network space. The drive tag is the same as in previous examples, as well as the label and description tags. The access-broker tag in this example is making use of the JvmAccessBroker. The entry tag describes the entry point the user should have into the target file system. In this example, the label the user will see (similar to 'My Files' above) is "Test" and this will lead the user to the directory F:/test/ on the server. These values should be replaced with the appropriate values for your institution. The identity type and access type specifications are the same as in the first example.

## 2.5 Personal Network Space

```

<drive handle="personalldap" max-upload="3145728" large-icon="largePersonal"
open-icon="folderopen" closed-icon="folderclosed" share-target="shared">
    <label>Personal LDAP Folders</label>
    <description>Your LDAP Personal Folders are where you can store your
documents, notes, photos, and more.</description>
    <access-broker handle="personal-uattr"
impl="net.unicon.alchemist.access.userattribute.UserAttributeAccessBroker">
        <user-attribute name="homeDirectory"
impl="net.unicon.academus.apps.briefcase.UserAttributeDirectory">
            <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
                <type handle="VIEW" value="GRANT"/>
                <type handle="ADD" value="GRANT"/>
                <type handle="DELETE" value="GRANT"/>
                <type handle="EDIT" value="GRANT"/>
                <type handle="SHARE" value="GRANT"/>
            </access>
            <max-size>0</max-size>
            <root-name>My Files 1</root-name>

```

```

        </user-attribute>
        <user-attribute name="class-dir"
impl="net.unicon.academus.apps.briefcase.UserAttributeDirectory">
        <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        </access>
        <max-size>0</max-size>
        <root-name>My Files 2</root-name>
        </user-attribute>
    </access-broker>
</drive>

```

The `UserAttributeAccessBroker` allows you to define the path to a briefcase folder by reading an attribute from LDAP or Active Directory. The value of this attribute should be the full path to the share or the full path on the filesystem to the user's personal storage space. The `<drive-handle>`, `<label>`, `<description>`, and `<access-broker>` tags are specified in the same way as in the previous examples.

The `<user-attribute>` tag allows you to specify an Academus attribute that will store the path to a user's personal space. In this example we are mounting the contents of the 'homeDirectory' attribute on the virtual briefcase folder named "My Files 1". We are also mounting the contents of the 'class-dir' attribute on the virtual briefcase folder names "My Files 2". Academus attributes are mapped to LDAP or Active Directory tags in the `PersonDirs.xml` file. Please contact Unicon Customer Support if you need assistance specifying an Academus attribute.

This method of specifying a briefcase storage location works seamlessly with Windows personal network space configurations that use the 'homeDirectory' attribute in Active Directory and no additional configuration is necessary. If you are running Unix/Linux then it is still possible to allow users access to their personal network space on a Windows-based file server. You will have to mount the entire Windows personal share tree on the Academus server using Samba or CIFS for this to work correctly. For example, if a user's homeDirectory attribute is `//server/home/user` then you must mount the entire 'home' directory in `/server/home` on the Unix/Linux server. Windows will automatically translate `//server/home/user` to the appropriate share on the file server, however, Unix/Linux will interpret this as a path on the filesystem (namely `/server/home/user`).

### 2.5.1 File system access requirements

Formatted: Bullets and Numbering

Keep in mind that the Academus server must have full read/write access to the entire 'home' tree for this to work correctly. Academus cannot utilize file system permission settings on a per-user basis. The Briefcase Portlet, as part of the Academus application running within a web application container, runs as and accesses the file system as the server user account that is running the application server (that is, a system user such as "nobody" or "academus"). Since it is accessing the file system as a single administrative user, that user must have permission to read and write the files in order for the Academus system to provide read and write access to the files via its user interface.

However, the briefcase channel has its own set of permissions if you wish to restrict access. You can configure the briefcase channel to emulate the permissions you would have applied via the file system if the users were accessing the file system directly "as themselves" rather than through Academus.

## 3.0 Briefcase.config

The entire briefcase.config file has been included here for your convenience:

```
<!--
```

```
*****  
** BRIEFCASE PORTLET CONFIGURATION **  
*****
```

Use this XML document to configure the BriefcasePortlet. The root element must be <briefcase>.

```
Drive(s)  
*****
```

The BriefcasePortlet may include one or more 'drives,' top-level nodes from which folders and files may be accessed. Specify a <drive> element for each drive you wish to include.

The <drive> element has the following attributes:

- @handle           A simple identifier for the drive. No two drives may have the same handle (document scope).
- @max-upload       [Optional] Maximum file size (in bytes) that may be uploaded to the drive. 3MB=3145728, 4MB=4194304, 5MB=5242880, &c. Setting the value to zero or omitting the attribute means there is no limit to individual file size.
- @large-icon       Specifies the look of the large drive icon on the welcome page. Use 'largePersonal,' 'largeShared,' 'largeNetwork,' or a custom icon. Custom icons must be defined as CSS classes within the rendering layer (Modulo).
- @open-icon        Specifies the look of the open drive icon on the folder view. Use 'folderopen,' 'sharedfolderopen,' 'networkfolderopen,' or a custom icon. Custom icons must be defined as CSS classes within the rendering layer (Modulo).
- @closed-icon      Specifies the look of the open drive icon on the folder view. Use 'folderclosed,' 'sharedfolderclosed,' 'networkfolderclosed,' or a custom icon. Custom icons must be defined as CSS classes within the rendering layer (Modulo).
- @share-target     [Optional] Specifies the handle of a drive (defined in this document) to which folders may be shared from this drive. If a drive is not specified, folders in this drive may not be shared.

The <drive> element has the following child elements:

- <label>           The label given to this drive in the UI.
- <description>     The description given to this drive in the UI.
- <access-broker>   Responsible for managing the root folders (entry points)

that will be available to users within this drive. Entry points are highly configurable and a little complex. See the section below for details.

#### Access Broker \*\*\*\*\*

Each drive contains an access broker to manage the root folders (entry points) available within it. Specify an <access-broker> element within each drive defined by the document.

The <access-broker> element has the following attributes:

- @handle           A simple identifier for the broker. No two brokers may have the same handle (JVM scope).
- @impl             Specifies a Java class that implements IAccessBroker. At the time of this writing, there are three available implementations: JvmAccessBroker, RdbmsAccessBroker, and JitAccessBroker. The allowable content model of the <access-broker> element depends upon the specified implementation.

#### JvmAccessBroker:

As its name suggests, the JvmAccessBroker keeps its data entirely within memory (i.e. the JVM). Any information held by an instance of this class is gone on restart. The content model of the <access-broker> element specifies the data that should be loaded into the instance on creation. The JvmAccessBroker is a great way to set up access relationships that are understood at deployment time.

For the JvmAccessBroker, the <access-broker> element has the following child elements:

- <entry>           Specifies an entry in the broker. See example below...

Example <entry> element:

```
<entry
target="FSA://net.unicon.demetrius.fac.filesystem.FsResourceFactory/[Test]/F/te
st">// Use standard Resource Factory URL syntax.
  <identity type="GROUP">Everyone</identity>// Type must be 'GROUP' or 'USER.'
  Specify the path to a group.
  <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">//
1=read, 2=edit, 3=delete
  <type handle="READ" value="GRANT"/>
  <type handle="ADD" value="GRANT"/>
  <type handle="DELETE" value="GRANT"/>
  <type handle="EDIT" value="GRANT"/>
  <type handle="SHARE" value="GRANT"/>
</access>
</entry>
```

#### RdbmsAccessBroker:

As its name suggests, the RdbmsAccessBroker stores its data in a database. The RdbmsAccessBroker is a good choice for entries that need to survive a restart but are not known at deployment time.

For the RdbmsAccessBroker, the <access-broker> element has a single child element, <access>. This child element requires the attribute 'impl', which

specifies the java class that represents the list of AccessTypes used by that access-broker instance.

Example element:

```
<access-broker handle="shared-resources"
impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker">
  <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
</access-broker>
```

JitAccessBroker:

The 'J-I-T' in JitAccessBroker stands for 'just-in-time.' The JitAccessBroker creates entries where none exist according to rules specified in its content model. Use the JitAccessBroker to create individual, instanced entry points for users and groups. The JitAccessBroker uses the nested enclosure pattern, and must be backed by another broker instance.

For the JitAccessBroker, the <access-broker> element has the following child elements:

- <jit-rule> Specifies a creation rule. The JitAccessBroker will apply the \*first\* (and only thr first) behavior that triggers for each rule. See example below...
- <access-broker> Specifies another broker instance to do the actual data management.

Example <jit-rule> element:

```
<jit-rule>// The JitAccessBroker will review each rule, not just the first to
'work.'
  <behavior>// The 1st behavior to trigger will be applied (if not already
applied). Subsequent behaviors will be ignored.
    <trigger type="GROUP">Everyone-Faculty</trigger>// If the user matches
this criterion, ...
    <target type="USER" />// ...for the identity specified, ...
    <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">//
...together with the specified access type(s), ...
      <type handle="READ" value="GRANT"/>
      <type handle="ADD" value="GRANT"/>
      <type handle="DELETE" value="GRANT"/>
      <type handle="EDIT" value="GRANT"/>
      <type handle="SHARE" value="GRANT"/>
    </access>
    <creator impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">//
...we will construct a target and create an entry for it.
      <size>200</size>
      <root-name>My Files</root-name>
      <seed-path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>
    </creator>
  </behavior>
  <behavior>// Subsequent behaviors are reviewed if earlier ones didn't
trigger.
    <trigger type="GROUP">Everyone-STUDENTS</trigger>
    <target type="USER" />
    <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
      <type handle="READ" value="GRANT"/>
      <type handle="ADD" value="GRANT"/>
      <type handle="DELETE" value="GRANT"/>
```

```

        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
    </access>
    <creator impl="net.unicon.academus.apps.briefcase.FsFactoryCreator">
        <size>50</size>
        <root-name>My Files</root-name>
        <seed-path>@INSTALLDIR@@LOCALDATAPATH@/briefcase_portlet</seed-path>
    </creator>
</behavior>
</jit-rule>

```

#### UserAttributeAccessBroker:

The UserAttributeAccessBroker creates entries for the user based on their attributes in either LDAP or active directory.

For the UserAttributeAccessBroker, the <access-broker> element has the <user-attribute> child element. The name attribute specifies the attribute name that can be retrieved from the User object. The impl attribute specifies the implementing class that understands the required object type. For the briefcase this will be "net.unicon.academus.apps.briefcase.UserAttributeDirectory".

Here is an example :

```

<user-attribute name="homeDirectory"
    impl="net.unicon.academus.apps.briefcase.UserAttributeDirectory">
    <access impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
// access that the user has on this entry
        <type handle="VIEW" value="GRANT"/>
        <type handle="ADD" value="GRANT"/>
        <type handle="DELETE" value="GRANT"/>
        <type handle="EDIT" value="GRANT"/>
        <type handle="SHARE" value="GRANT"/>
    </access>
    <max-size>0</max-size>           // max-size of the factory. 0 =
unlimited space
    <root-name>My Files 1</root-name> // name of the entry root
</user-attribute>
//-->

<briefcase>

    <!--
        Civis implementation to be used to resolve usernames and group paths
        to academus users and groups. This should not require
        modification, as it utilized the Academus framework for gathering this
        information.
    -->
    <civis id="addressBook"
    impl="net.unicon.civis.fac.academus.AcademusCivisFactory">
        restrictor
        impl="net.unicon.civis.grouprestrictor.AcademusGroupRestrictor" />
    /civis>

    <drive handle="personal" max-upload="3145728" large-icon="largePersonal"
open-icon="folderopen" closed-icon="folderclosed" share-target="shared">

```



```

    </behavior>
  </behavior>
    <trigger type="GROUP">Everyone - Students</trigger>
    <target type="GROUP">Everyone - Students</target>
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
      <type handle="VIEW" value="GRANT"/>
      <type handle="ADD" value="GRANT"/>
      <type handle="DELETE" value="GRANT"/>
      <type handle="EDIT" value="GRANT"/>
      <type handle="SHARE" value="GRANT"/>
    </access>
    <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
  </behavior>
  <behavior>
    <trigger type="GROUP">Everyone - Staff</trigger>
    <target type="GROUP">Everyone - Staff</target>
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
      <type handle="VIEW" value="GRANT"/>
      <type handle="ADD" value="GRANT"/>
      <type handle="DELETE" value="GRANT"/>
      <type handle="EDIT" value="GRANT"/>
      <type handle="SHARE" value="GRANT"/>
    </access>
    <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
  </behavior>
  <behavior>
    <trigger type="GROUP">Everyone</trigger>
    <target type="GROUP">Everyone</target>
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
      <type handle="VIEW" value="GRANT"/>
      <type handle="ADD" value="GRANT"/>
      <type handle="DELETE" value="GRANT"/>
      <type handle="EDIT" value="GRANT"/>
      <type handle="SHARE" value="GRANT"/>
    </access>
    <creator
impl="net.unicon.alchemist.access.permissions.DummyCreator"/>
  </behavior>
</jit-rule>

  <access-broker
    handle="personal-resources-p-i"

impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker"
    needsDataSource="true">
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
  </access-broker>
</access-broker>
</permissions>

```

```

    </access-broker>
  </drive>
  <drive handle="personalldap" max-upload="3145728" large-icon="largePersonal"
open-icon="folderopen" closed-icon="folderclosed" share-target="shared">
    <label>Personal LDAP Folders</label>
    <description>Your LDAP Personal Folders are where you can store your
documents, notes, photos, and more.</description>
    <access-broker handle="personal-uattr"
impl="net.unicon.alchemist.access.userattribute.UserAttributeAccessBroker">
      <user-attribute name="homeDirectory"
impl="net.unicon.academus.apps.briefcase.UserAttributeDirectory">
        <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
          <type handle="VIEW" value="GRANT"/>
          <type handle="ADD" value="GRANT"/>
          <type handle="DELETE" value="GRANT"/>
          <type handle="EDIT" value="GRANT"/>
          <type handle="SHARE" value="GRANT"/>
        </access>
        <max-size>0</max-size>
        <root-name>My Files 1</root-name>
      </user-attribute>
      <user-attribute name="class-dir"
impl="net.unicon.academus.apps.briefcase.UserAttributeDirectory">
        <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
          <type handle="VIEW" value="GRANT"/>
          <type handle="ADD" value="GRANT"/>
        </access>
        <max-size>0</max-size>
        <root-name>My Files 2</root-name>
      </user-attribute>
    </access-broker>
  </drive>
  <drive handle="shared" large-icon="largeShared" open-icon="sharedfolderopen"
closed-icon="sharedfolderclosed" sharing="off">
    <label>Shared Folders</label>
    <description>Access other files that have been shared with
you.</description>
    <access-broker handle="shared-resources"
impl="net.unicon.alchemist.access.rdbms.RdbmsAccessBroker">
      <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType"/>
    </access-broker>
  </drive>

  <drive handle="network" large-icon="largeNetwork" open-
icon="networkfolderopen" closed-icon="networkfolderclosed" sharing="off">
    <label>Network Folders</label>
    <description>The Network Folders contain common folders that you and
others on the Campus can access.</description>
    <access-broker handle="network-resources"
impl="net.unicon.alchemist.access.jvm.JvmAccessBroker">

```

```
    <entry
target="FSA://net.unicon.demetrius.fac.filesystem.FsResourceFactory/[Test]/F:/test">
    <identity type="GROUP">Everyone</identity>
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
    <type handle="VIEW" value="GRANT"/>
    <type handle="ADD" value="GRANT"/>
    <type handle="DELETE" value="GRANT"/>
    <type handle="EDIT" value="GRANT"/>
    <type handle="SHARE" value="GRANT"/>
    </access>
    </entry>
    <entry
target="FSA://net.unicon.demetrius.fac.filesystem.FsResourceFactory/[Authoring]/I:/Authoring">
    <identity type="GROUP">Everyone</identity>
    <access
impl="net.unicon.academus.apps.briefcase.BriefcaseAccessType">
    <type handle="VIEW" value="GRANT"/>
    </access>
    </entry>
    </access-broker>
</drive>
</briefcase>
```