

The page features several decorative squares of varying sizes and colors (blue and grey) scattered across the layout. A large blue square is in the top left, a smaller blue square is in the middle left, a grey square is below it, a large grey square is on the right side, and a large blue square is in the bottom right. The text is centered on the right side of the page.

PAGS

Configuration and Examples

Contents

| | |
|---|-----------|
| 1.0 Introduction | 4 |
| 2.0 Enabling PAGS | 5 |
| 3.0 Configuring PAGS Groups | 6 |
| 3.1 Creating Structure with <group-key> and <member-key>..... | 6 |
| 3.2 Testing for Group Membership..... | 6 |
| 3.2.1 The String Match Test..... | 7 |
| 3.2.2 The Case Insensitive String Match Test | 8 |
| 3.2.3 The Integer Match Test | 8 |
| 3.2.4 The Regular Expression Match Test | 9 |
| 3.2.5 Combining Multiple Tests | 10 |
| 4.0 Adding PAGS Root | 12 |
| 4.1 Adding PAGS Root to the Portal | 12 |
| 5.0 PAGS Example | 13 |
| 5.1 Classifying Students based on Major | 13 |
| 5.2 Configuring the Root Group | 13 |
| 5.3 Configuring the College Groups..... | 14 |
| 5.4 Configuring the Major Groups | 15 |

1.0 Introduction

The Person Attribute Groups Service, or PAGS, is designed to help you automatically organize your users into groups based on their Active Directory or LDAP attributes. This is done by specifying a group name and creating a set of rules that designate which users should belong to the group. These rules are specified by regular expressions executed against different Active Directory or LDAP attributes. For example, you could set up a group for each of the different majors offered at your University by testing the Active Directory or LDAP attribute containing the Student's major. The advantage of using PAGS over static portal groups is that the group lists are constructed dynamically based on the information in Active Directory or LDAP, so if a Student changes major, the change will automatically be reflected in the Student's group affiliation in Academus. There are certain limitations to using PAGS. These limitations are introduced due to the inherent nature of PAGS. PAGS groups are unaware of the list of users that belong to the group.

The channels affected with the use of PAGS are

- Broadcast emails to group members will not work (survey channel, Calendar Channel, Messaging Portlet)
- Group Chat admin - will not be able to see the members in an activated chat room
- Group Chat Admin - not able to specify chat permissions per user.

2.0 Enabling PAGS

The following files must be modified to enable PAGS on your Academus portal:

/portal/unicon/Academus/portal-tomcat-a/webapps/portal/WEB-INF/classes/properties/groups/**compositeGroupServices.xml**

/portal/unicon/Academus/portal-tomcat-a/webapps/portal/WEB-INF/classes/properties/groups/**PAGSGroupStoreConfig.xml**

The following XML block needs to be uncommented from **compositeGroupServices.xml**:

```
<service>
  <name>pags</name>
  <service_factory>
    org.jasig.portal.groups.ReferenceIndividualGroupServiceFactory
  </service_factory>
  <entity_store_factory>
    org.jasig.portal.groups.pags.PersonAttributesEntityStoreFactory
  </entity_store_factory>
  <group_store_factory>
    org.jasig.portal.groups.pags.PersonAttributesGroupStoreFactory
  </group_store_factory>
  <entity_searcher_factory>
    org.jasig.portal.groups.pags.PersonAttributesEntitySearcherFactory
  </entity_searcher_factory>
  <internally_managed>>false</internally_managed>
  < caching_enabled>true</ caching_enabled>
</service>
```

Uncommenting this block will enable PAGS on your Academus portal. The PAGS groups must then be configured within the **PAGSGroupStoreConfig.xml** file.

3.0 Configuring PAGS Groups

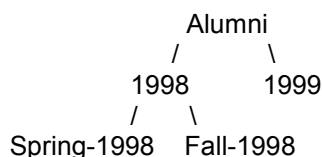
3.1 Creating Structure with <group-key> and <member-key>

PAGS groups are specified by XML blocks in the **PAGSGroupStoreConfig.xml** file. Typical PAGS group specifications might look like the following:

```
<group>
  <group-key>0</group-key>
  <group-name>PAGS Root</group-name>
  <group-description>Root group for all PAGS groups. Convenience for adding PAGS groups to a local
    group.</group-description>
  <members>
    <member-key>1</member-key>
    <member-key>2</member-key>
  </members>
</group>
```

This XML block defines the PAGS Root group, or top-level group. This is generally defined for convenience and does not actually do any filtering. The **<group-key>** is a unique identifier that must be specified for each PAGS group. The **<group-key>** is combined with the **<member-key>** to create a tree structure out of the PAGS groups for filtering purposes. For example, you may have a PAGS group for your Alumni students, and then subgroups under the Alumni group for students that graduated in 1998, 1999, 2000, etc.

To set up this structure, you would first specify a unique **<group-key>** for your Alumni group. You would then, within the Alumni group, specify a **<member-key>** for each group underneath the Alumni group. For example, the Alumni group may have a **<group-key>** of 1, the 1998 group may have a **<group-key>** of 2, and the 1999 group may have a **<group-key>** of 3. To create this structure, you would create an Alumni group with a **<group-key>** of 1 and **<member-key>**'s of 2 and 3. Your 1998 group should have a **<group-key>** of 2, and your 1999 group should have a **<group-key>** of 3. You do not need to provide **<member-key>** tags for the 1998 and 1999 groups unless you wanted to specify further sub-groups for the 1998 and 1999 groups. For example, if you had a 1998-Spring and 1998-Fall group specified with **<group-key>**'s of 4 and 5, you would setup **<member-key>**'s in the 1998 group of 4 and 5.



3.2 Testing for Group Membership

Once you've setup your group structure you need to define your tests for membership. The following tester classes (all in the package `org.jasig.portal.groups.pags.testers`) come with PAGS:

| | |
|-----------------|--|
| IntegerEQTester | Converts attribute and test value to ints. |
|-----------------|--|

| | |
|------------------------------|---|
| | Attribute must be EQ to the test value. In the event of a NumberFormatException, the test fails (true for all Integer testers.) |
| IntegerGETester | Attribute must be Greater or Equal test value. |
| IntegerGTTester | Attribute must be Greater Than test value. |
| IntegerLETester | Attribute must be Less or Equal test value. |
| IntegerLTTester | Attribute must be Less Than test value. |
| RegexTester | Attribute must match a regular expression. Do not include the delimiter. |
| StringEqualsIgnoreCaseTester | String comparison ignoring case. |
| StringEqualsTester | String comparison. |

Basic examples have been provided for your reference.

1. A string match test
2. A case-insensitive string match test
3. An integer match test
4. A regular expression

3.2.1 The String Match Test

The simplest way to test for group membership is to set up a string match test. This type of test will compare the string you specify in the PAGS group configuration with an Active Directory or LDAP attribute. Users whose attribute exactly matches the string will be given membership to the group. A basic example has been provided below:

```
<group>
  <group-key>3</group-key>
  <group-name>Students</group-name>
  <group-description>Portal users whose first names equal Student</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>firstName</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.StringEqualsTester</tester-class>
        <test-value>Student</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

In this example we're creating a group called 'Students' that tests the 'firstName' attribute to see if it's equal to the exact string 'Student'. You do this by specifying a `<test-group>` and `<test>` with the following attributes:

<attribute-name>: This is the actual Academus attribute to test against, specified in **PersonDirs.xml**. Note: this is not the Active Directory or LDAP name of the attribute, this is the name that Academus gives to the attribute in **PersonDirs.xml** (the **<alias>** tag, not the **<name>** tag).

<tester-class>: This is the type of test to be performed on the Academus attribute.

<test-value>: This is the value to test for.

3.2.2 The Case Insensitive String Match Test

This test is exactly the same as the example above; however, it ignores case when checking for group membership. In this example, we test for last names matching the string "User". However, since this is a case insensitive test, we would also match users with the last name of "USER", "user", "UsEr", etc.

```
<group>
  <group-key>3</group-key>
  <group-name>Template Users</group-name>
  <group-description>Portal users whose first names equal Student</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>lastName</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>User</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

3.2.3 The Integer Match Test

Another simple way to test for group membership is to set up an integer match test. This type of test will compare the integer you specify in the PAGS group configuration with an Active Directory or LDAP attribute. Users whose attribute tests true for the specified tester given the integer will be given membership to the group. The below example would create a group of users whose birthYear attribute is less than or equal to 1975:

```
<group>
  <group-key>3</group-key>
  <group-name>Reunion Students</group-name>
  <group-description>Portal users whose birth year is before 1976</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>birthYear</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.IntegerLETester </tester-class>
        <test-value>1975</test-value>
      </test>
    </test-group>
  </selection-test>
```

```
</group>
```

3.2.4 The Regular Expression Match Test

This is the most powerful of the three tests. It allows you to specify a regular expression as a match test against any Academus attribute. A simple example has been provided below:

```
<group>
  <group-key>2</group-key>
  <group-name>Short First Names</group-name>
  <group-description>
    Portal users whose first names are between 1 and 5 characters long
  </group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>givenName</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.RegexTester</tester-class>
        <test-value>^{1,5}$</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

In this example, we're testing for a string that contains from 1 to 5 characters. The rules of regular expressions denote the following:

- ^ - matches the beginning of a string
- \$ - matches the end of a string
- .
- {1,5} – specifies a range from 1 to 5

Summarized, the regular expression `^{1,5}$` reads as: find the beginning of the string (^), look for any character (.), look for a range of 'any characters' {1,5}, look for the end of the string (\$). Another example of regular expression usage:

```
<group>
  <group-key>3</group-key>
  <group-name>Ds</group-name>
  <group-description>Portal users whose first names begin with D</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>firstName</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.RegexTester</tester-class>
        <test-value>^D.*$</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

In this example, we're testing for users whose first name begins with the letter 'D'. The * indicates "0 or more of the previous element". The regular expression reads like this: locate the beginning of the

string (^), the first letter of the string must be a D, and we then look for 0 or more (*) of any character (.) until the end of the string (\$).

A full tutorial on regular expressions is beyond the scope of this document, however, there are many books and web sites that explain how regular expressions are created and interpreted.

3.2.5 Combining Multiple Tests

Individual tests are aggregated into test groups, and their results are AND-ed together, so all tests in a test group must return true for the test group to return true. A test group is described by a `<test-group>` element. If there is more than 1 `<test-group>` element, the results of the test groups are OR-ed together, so if any one test group returns true, the tests return true. A true result means that the candidate `user` is a member of the group.

For the “OR” combination, multiple `<test-group>`'s need to be specified, and at least one of the tests must pass for the user to be placed in the group. In the example below we combine two of the above examples into one test:

```
<group>
  <group-key>3</group-key>
  <group-name>Ds and Students</group-name>
  <group-description>
    Portal users whose first names begin with D or equal Student
  </group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>firstName</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.RegexTester</tester-class>
        <test-value>^D.*$</test-value>
      </test>
    </test-group>
    <test-group>
      <test>
        <attribute-name>firstName</attribute-name>
        <tester-class>org.jasig.portal.groups.pags.testers.StringEqualsTester</tester-class>
        <test-value>Student</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

This test will match users whose first name either begins with a ‘D’ or is exactly ‘Student’.

For the “AND” combination, multiple `<test>` elements need to be specified within a `<test-group>` and all tests must pass for the user to be placed in the group. In the example below we combine two of the above examples into one test:

```
<group>
  <group-key>3</group-key>
  <group-name>Students in 30's</group-name>
  <group-description>
    Portal users whose who are in their 30's
  </group-description>
```

```
<selection-test>
  <test-group>
    <test>
      <attribute-name>birthYear</attribute-name>
      <tester-class>org.jasig.portal.groups.pags.testers.IntegerLETester </tester-class>
      <test-value>1975</test-value>
    </test>
    <test>
      <attribute-name>birthYear</attribute-name>
      <tester-class>org.jasig.portal.groups.pags.testers.IntegerGTTester </tester-class>
      <test-value>1965</test-value>
    </test>
  </test-group>
</selection-test>
</group>
```

This test will match users who were born between the years 1966 and 1975.

4.0 Adding PAGS Root

4.1 Adding PAGS Root to the Portal

In order for the PAGS Root and its groups to be visible through the portal, the PAGS Root group must be manually added to the Everyone group. To add the PAGS Root group:

- 1) Login to the portal as a user with access to the Groups Manager Channel
- 2) Locate the Groups Manager Channel and click on 'Everyone'
- 3) Unlock the group by clicking on the lock icon
- 4) Click on 'Add Members'
- 5) In the "Search for a" field, select 'Group of Persons'
- 6) In the "whose name" text box, enter PAGS
- 7) Click 'Go'
- 8) Click in the box to the left of "PAGS Root"
- 9) Click 'Select Marked'
- 10) Click 'Done with Selection'
- 11) Lock the group by clicking on the lock icon

The PAGS Root group will now appear in the list of groups associated with the Everyone group. If you select PAGS Root, you will see the PAGS groups that you configured in the **PAGSGroupStoreConfig.xml** file.

5.0 PAGS Example

This section will walk you through an example of how PAGS can be used to classify students based on which college they attend and what major they have chosen. Please note that this information must exist in LDAP for PAGS to be able to use it. In this example, we will assume that the LDAP attribute 'college' contains the name of the college each student attends, and the LDAP attribute 'major' contains the name of the major each student has chosen.

5.1 Classifying Students based on Major

For this example, let's assume our University has 3 colleges, each of which offer 3 majors. First, we assign PAGS group numbers to each of the groups we anticipate having. The numbers below appear out of order but the ordering will make sense as we continue through the example. The breakdown looks like this:

- ❖ College of Liberal Arts -- 1
 - Anthropology -- 4
 - Sociology -- 5
 - Religious Studies -- 6

- ❖ College of Engineering -- 2
 - Electrical Engineering -- 7
 - Mechanical Engineering -- 8
 - Computer Science -- 9

- ❖ College of Business -- 3
 - Finance -- 10
 - Management -- 11
 - Marketing -- 12

5.2 Configuring the Root Group

The root group is the top-level PAGS group, as discussed previously. For this example, we're going to place our college groups directly below the PAGS root group. We decided above that our college groups would be assigned the PAGS group numbers 1, 2, and 3, so we specify this in the root group:

```
<group>
  <group-key>0</group-key>
  <group-name>PAGS Root</group-name>
  <group-description>PAGS Root group.</group-description>
  <members>
    <member-key>1</member-key>
    <member-key>2</member-key>
    <member-key>3</member-key>
  </members>
</group>
```

In a more complex configuration you may want to create an additional group level off of the PAGS Root. For example, if you wanted to create PAGS groups for both Alumni students and current students, you might

configure the root group to contain an Alumni group and a Current Students group. You would then configure the individual groups underneath these.

5.3 Configuring the College Groups

Next we configure the groups for each of the colleges. The portal will check the LDAP attribute 'college' to see if it matches the value in the `<test-value>` tag. Notice how the `<group-key>` tags correspond to the `<member-key>` tags in the root group. We also setup `<member-key>` tags for each major offered under each college. These groups will be setup in the next section.

```
<group>
  <group-key>1</group-key>
  <group-name>Liberal Arts Students</group-name>
  <group-description>Students enrolled in the College of Liberal Arts</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>college</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>LiberalArts</test-value>
      </test>
    </test-group>
  </selection-test>
  <members>
    <member-key>4</member-key>
    <member-key>5</member-key>
    <member-key>6</member-key>
  </members>
</group>
```

```
<group>
  <group-key>2</group-key>
  <group-name>Engineering Students</group-name>
  <group-description>Students enrolled in the College of Engineering</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>college</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Engineering</test-value>
      </test>
    </test-group>
  </selection-test>
  <members>
    <member-key>7</member-key>
    <member-key>8</member-key>
    <member-key>9</member-key>
  </members>
</group>
```

```

<group>
  <group-key>3</group-key>
  <group-name>Business Students</group-name>
  <group-description>Students enrolled in the College of Business</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>college</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Business</test-value>
      </test>
    </test-group>
  </selection-test>
  <members>
    <member-key>10</member-key>
    <member-key>11</member-key>
    <member-key>12</member-key>
  </members>
</group>

```

5.4 Configuring the Major Groups

The last step is to configure the groups for each individual major. The value in the `<group-key>` tags will correspond to the `<member-key>` tags above. Notice that there are no `<member-keys>` in these group definitions. This is because these are the bottom level groups and there are no additional groups underneath these. If you had, for example, two different Anthropology majors, you might want to add them underneath a group called Anthropology. In this case you would have `<member-key>` tags in the Anthropology group for the two types of Anthropology majors.

```

<group-key>4</group-key>
<group-name>Anthropology Students</group-name>
<group-description>Students majoring in Anthropology</group-description>
<selection-test>
  <test-group>
    <test>
      <attribute-name>major</attribute-name>
      <tester-class>
        org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
      </tester-class>
      <test-value>Anthropology</test-value>
    </test>
  </test-group>
</selection-test>
</group>

<group>
  <group-key>5</group-key>
  <group-name>Sociology Students</group-name>
  <group-description>Students majoring in Sociology</group-description>
  <selection-test>

```

```
<test-group>
  <test>
    <attribute-name>major</attribute-name>
    <tester-class>
      org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
    </tester-class>
    <test-value>Sociology</test-value>
  </test>
</test-group>
</selection-test>
</group>

<group>
  <group-key>6</group-key>
  <group-name>Religious Studies Students</group-name>
  <group-description>Students majoring in Religious Studies</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Religious Studies</test-value>
      </test>
    </test-group>
  </selection-test>
</group>

<group>
  <group-key>7</group-key>
  <group-name>Electrical Engineering</group-name>
  <group-description>Students majoring in Electrical Engineering</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>ElectricalEngineering</test-value>
      </test>
    </test-group>
  </selection-test>
</group>

<group>
  <group-key>8</group-key>
  <group-name>Mechanical Engineering</group-name>
  <group-description>Students majoring in Mechanical Engineering</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
```

```
<tester-class>
  org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
</tester-class>
<test-value>MechanicalEngineering</test-value>
</test>
</test-group>
</selection-test>
</group>

<group>
  <group-key>9</group-key>
  <group-name>Computer Science</group-name>
  <group-description>Students majoring in Computer Science</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>ComputerScience</test-value>
      </test>
    </test-group>
  </selection-test>
</group>

<group>
  <group-key>10</group-key>
  <group-name>Finance</group-name>
  <group-description>Students majoring in Finance</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Finance</test-value>
      </test>
    </test-group>
  </selection-test>
</group>

<group>
  <group-key>11</group-key>
  <group-name>Management</group-name>
  <group-description>Students majoring in Management</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Management</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

```
</test>
</test-group>
</selection-test>
</group>

<group>
  <group-key>12</group-key>
  <group-name>Marketing</group-name>
  <group-description>Students majoring in Marketing</group-description>
  <selection-test>
    <test-group>
      <test>
        <attribute-name>major</attribute-name>
        <tester-class>
          org.jasig.portal.groups.pags.testers.StringEqualsIgnoreCaseTester
        </tester-class>
        <test-value>Marketing</test-value>
      </test>
    </test-group>
  </selection-test>
</group>
```

This concludes the simple college and major example. Provided the LDAP values are in place, users will now be associated with the appropriate groups depending on their college and major. Students can now be referenced by college or major in various portal channels, such as the announcements and calendar channels.