

WebSphere Portal Express – An Introduction to enabling Single Sign-On to Backend Applications

by Dick Salz

IBM eServer Solutions Enablement

January 2005

Table of Contents

Overview	1
Lab Requirements.....	1
What you should be able to do	1
Introduction.....	1
Exercise Instructions	2
Part 1: Getting Started.....	2
Starting the Portal Server through the HTTP Server Administration Interface	2
Starting the Portal Server through the iSeries 5250 Interface	4
Preparing the Portal Server	4
Part 2: Exercising a Private User Slot	7
Part 3: Exercising the Shared User Slot	17
Part 4: Exercising the System Slot.....	25
Summary	30
Appendix A: Analysis of Private User Slot code.....	31
Trademarks	34

Overview

This lab will introduce you to the single sign-on support in WebSphere® Portal, providing a mechanism that assists a portlet in retrieving one of several representations of a user's authenticated identity, which the portlet can then pass to a backend application. You will be introduced to the Credential Vault, a portal service that helps portlets and portal users manage multiple identities. The credential vault provided by WebSphere Portal distinguishes between different types of vault slots and this lab will illustrate the differences in functionality of private slots, shared slots, and system slots in the credential vault. You will exercise the functionality of each of these slots by installing a number of portlets which create or use credential vault slots to store and retrieve credentials.

Lab Requirements

- ? WebSphere Portal V5.0.2 and its prerequisites
- ? Completion of the portal lab: WebSphere Portal Express – An Introduction to the Portal Access Control Facility

What you should be able to do

At the end of this lab, you will have exercised some of the following functions of the Credential Vault:

- ? Gain experience using a private user slot
- ? Gain experience using a shared user slot
- ? Configure and use a system slot

Introduction

The credential vault offers the ability for portlets to store and retrieve credentials. The primary objective of the credential vault is to enable single sign-on with backend applications, in environments where multiple user identities may be necessary. For instance, your portlet may need to send an e-mail message, and therefore, it needs to sign on to the e-mail server. Usually, there is no guarantee that the user ID and password that were utilized to log into the WebSphere Portal can also be used to sign on to the e-mail server. A portlet could store the e-mail user identity in a slot in the credential vault and retrieve those credentials to log in to the e-mail server.

There are a number of different ways to use the credential vault. Most of the times, you will use a programmatic approach. Portlets can create slots and store credentials, and then retrieve them at a later time. Slots created this way are called “user slots” and can be “shared” or “private.” A shared slot is accessible from any portlet that runs under a certain user identity. A private slot is only accessible to a specific concrete portlet running under a specific user identity.

The Administrator can also create slots – called system slots. System slots are special slots that are visible to all of the portlets, irrespective of the user identity that is used to run them. An administrator can set the credentials in a system slot using the administrative function of the credential vault, allowing portlets to retrieve those credentials programmatically.

In this exercise, you are going to install, run, and analyze three types of portlets that:

- 1) Create and use a private user slot
- 2) Create and use a shared user slot
- 3) Use a system slot which you previously created using the administrative interface

Exercise Instructions

Part 1: Getting Started

In the first portion of the lab, you will perform some preliminary actions which will later allow you to exercise the credential vault.

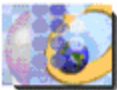
There are two ways to manage your portal server — through the HTTP server administration graphical client or through the iSeries™ 5250 interface. We will show you both methods in this lab. Use either the graphical interface or the iSeries 5250 interface to verify your WebSphere Portal Server is running.

Starting the Portal Server through the HTTP Server Administration Interface

- ___ 1. Using your browser, go to the HTTP Server Administration client. You will need to use the correct URL for your administration client: <http://fullyqualifiedhostname:2001>
- ___ 2. When prompted for your user ID and password, provide a valid iSeries user ID and password that will allow you the correct level of authority to work with the HTTP Server Administration client features.
- ___ 3. On the “iSeries Tasks” screen, click the first link, IBM Web Administration for iSeries.

iSeries Tasks

PWD3.RC



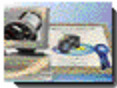
IBM Web Administration for iSeries

Configure HTTP servers, application servers and deploy applications



iSeries Navigator URL Advisor

Learn how to add OS/400 administration tasks into your web applications



Digital Certificate Manager

Create, distribute, and manage Digital Certificates



IBM Directory Server for iSeries

Administer the IBM Directory Server



IBM IPP Server for iSeries

Configure the IBM IPP Server



Cryptographic Coprocessor

Configure the cryptographic coprocessor



iSeries Web-Based Help Server

Administer the iSeries Web-based help server

4. On the **Manage All Servers** screen, click on the second tab, **All Application Servers**.
5. On the **All Application Servers** screen, you will see a list of the application servers running on your IBM® eServer™ iSeries server. You can see the status of each server instance. The screen capture shows some servers at the “Stopped” state and others at the “Running” state.

The screenshot shows the 'Manage All Servers' interface with the 'All Application Servers' tab selected. The data is current as of 00:26:18 PM UTC on 12/09/2004. The table below lists the servers:

Server	Version	Status	Address:Port
AVN100/AVN100	WAS, V5.1 (base)	Stopped	*:2010001,2010006,2010010,2010011,2010012,2010013
AVN99/AVN99	WAS, V5.1 (base)	Stopped	*:209901,209906,209910,209911,209912,209913
default/server1	WAS, V5.0 (base)	Stopped	*:2809,8880,9043,9080,9090
default/server 1	WAS, V5.1 (base)	Running	*:2809,8880,9043,9080,9090,9443
WAS51EL.WAS51EL	WAS, V5.1 (base)	Running	*:10014,10019,10023,10024,10025,10026
WAS5Portal/WAS5Portal	WAS, V5.0 (portal)	Stopped	*:10001,10006,10010,10011,10012
wps1rds/wps1rds	WAS, V5.0 (portal)	Running	*:50100,50105,50109,50110,50111
wps1team/wps1team	WAS, V5.0 (portal)	Running	*:20100,20105,20109,20110,20111

Below the table are control buttons: Refresh, Start, Stop, Restart, Manage Details, Delete, and Rename.

6. If your portal application server is not currently running, select the server by clicking the circle to the left of your portal server instance. In the example below, the WAS5Portal server has been selected.

The screenshot shows the 'Manage All Servers' interface with the 'All Application Servers' tab selected. The data is current as of 00:33:30 PM UTC on 12/09/2004. The table below lists the servers, with the WAS5Portal server selected (indicated by a green dot in the selection column):

Server	Version	Status	Address:Port
AVN100/AVN100	WAS, V5.1 (base)	Stopped	*:2010001,2010006,2010010,2010011,2010012,2010013
AVN99/AVN99	WAS, V5.1 (base)	Stopped	*:209901,209906,209910,209911,209912,209913
default/server1	WAS, V5.0 (base)	Stopped	*:2809,8880,9043,9080,9090
default/server 1	WAS, V5.1 (base)	Running	*:2809,8880,9043,9080,9090,9443
WAS51EL.WAS51EL	WAS, V5.1 (base)	Running	*:10014,10019,10023,10024,10025,10026
<input checked="" type="radio"/> WAS5Portal/WAS5Portal	WAS, V5.0 (portal)	Stopped	*:10001,10006,10010,10011,10012
wps1rds/wps1rds	WAS, V5.0 (portal)	Running	*:50100,50105,50109,50110,50111
wps1team/wps1team	WAS, V5.0 (portal)	Running	*:20100,20105,20109,20110,20111

Below the table are control buttons: Refresh, Start, Stop, Restart, Manage Details, Delete, and Rename.

7. Now, click the **Start** button to start the instance.

- ___ 8. You have successfully started your portal instance to begin the Portal Server Authorization lab.

Starting the Portal Server through the iSeries 5250 Interface

An alternate way to start your portal instance is through the iSeries 5250 interface. If you have already used the HTTP Web Administration interface to start your portal instance, you can skip directly to the “Preparing the Portal Server” section of this lab.

- ___ 1. Sign on to your iSeries server.
- ___ 2. From an OS/400 command prompt, enter **STRQSH**.
- ___ 3. Navigate to the /qibm/proddata/webas5/pme/bin directory by issuing the Qshell command: cd /qibm/proddata/webas5/pme/bin
- ___ 4. From the Qshell command prompt, type the following command, replacing “wpsXteam” with the name of your portal instance: **serverStatus wpsXteam -instance wpsXteam** (NOTE: the parameter values are case sensitive.)
- ___ 5. Output from this command indicates if your application server is started or not. Alternately, you can use the WRKACTJOB OS/400 command to determine if your application server is running in the QEJBAS5 subsystem.
- ___ 6. If your portal server instance is already started, you will see the message, **The Application Server “wpsXteam” is STARTED**. If your portal server is not started, follow the instructions in the step below to start the server.
- ___ 7. If it is not already started, you should start your WebSphere Portal server by issuing the following command: **startServer -instance wpsXteam**. Wait until you see the confirmation message: Server <your server name> open for e-business; process id is xxx
- ___ 8. Exit the Qshell environment by pressing PF3.

Preparing the Portal Server

- ___ 1. Log in as wpsadmin.
- ___ a) Using your browser, go to your portal instance using the correct URL:
http://hostname:port/wps/portal
 - ___ b) Click **Log In** at the upper right corner of the screen.
 - ___ c) Enter a user ID and password of the administration ID of the WebSphere Portal server.
For this lab example, we use id **wpsadmin**.
 - ___ d) Click the **Log in** button.
- ___ 2. Create a new user and make it part of the *wpsadmins* group. We need two users to demonstrate the flexibility of the credential vault.
- ___ a) Click **Administration on** the navigation bar in the upper right portion of the screen.
 - ___ b) Click the **Access** menu in the left navigation pane
 - ___ c) Click on **Users and Groups**.

- __ d) Click the **Search** button to populate the list of users and groups.
- __ e) Click on the **wpsadmins** group.
- __ f) Click on the **New user** button.
- __ g) Create a new user called **User2**. Specify a password of **password**. Select **English** as the language.

WebSphere Portal Administration Edit my profile ? Log out

Manage Users and Groups
Provide user information

* User ID:

* Password:

* Confirm Password:

* First Name:

* Last Name:

Email:

Preferred language:

*Required Field

- __ h) Click **OK** to create the user.

WebSphere Portal My

Manage Users and Groups

APMP0100I: User created successfully!

[Root](#) > **wpsadmins**

Members of cn=wpsadmins,o=default organization - add, edit and delete user groups

ID
User2
wpsadmin

Note – The user does not have to be part of the wpsadmins group in order to use the credential vault portlets. However, by making it part of this group, we won't have to define specific authorizations.

Part 2: Exercising a Private User Slot

___ 1. Install the Private Slot Portlet application.

___ a) From the WebSphere Portal Administration portlet, click on **Portlets** on the navigation menu.

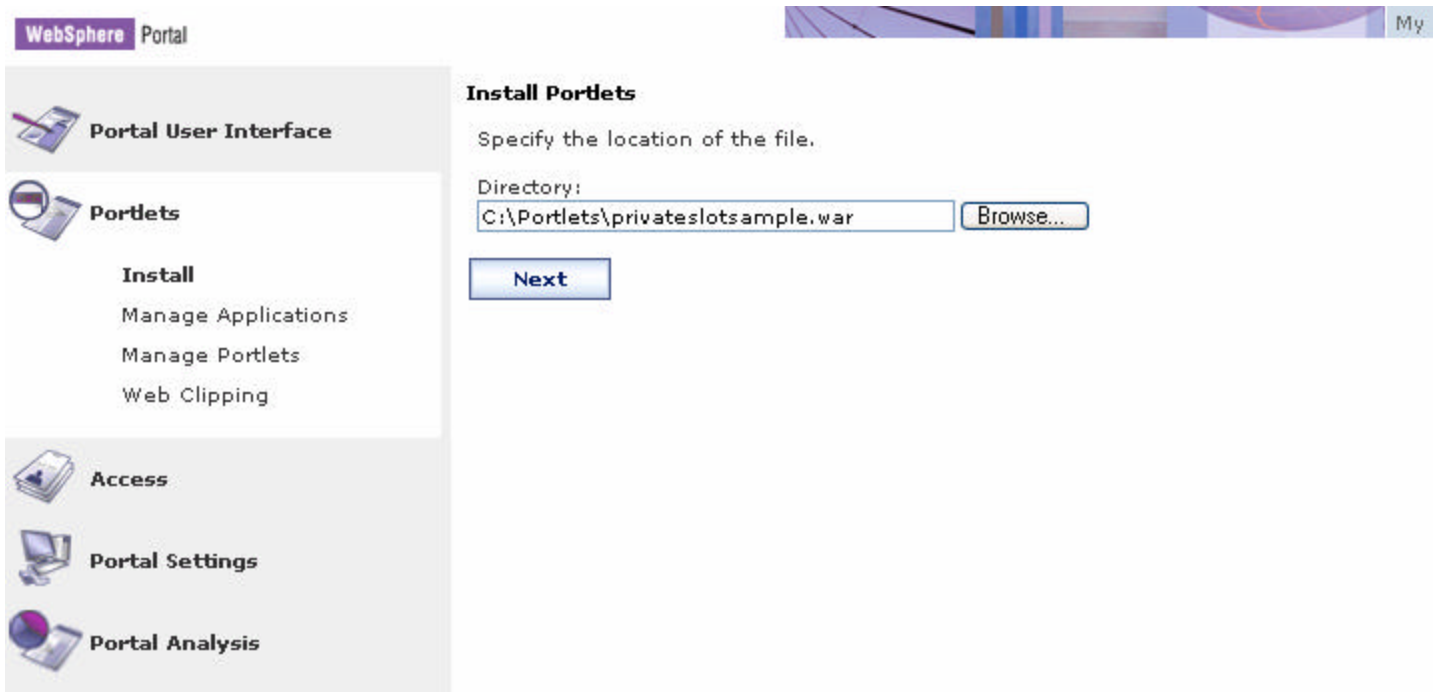
___ b) Now click **Install** to install a portlet.

___ c) To select the application to install, click **Browse** by the Directory field.

Directory:

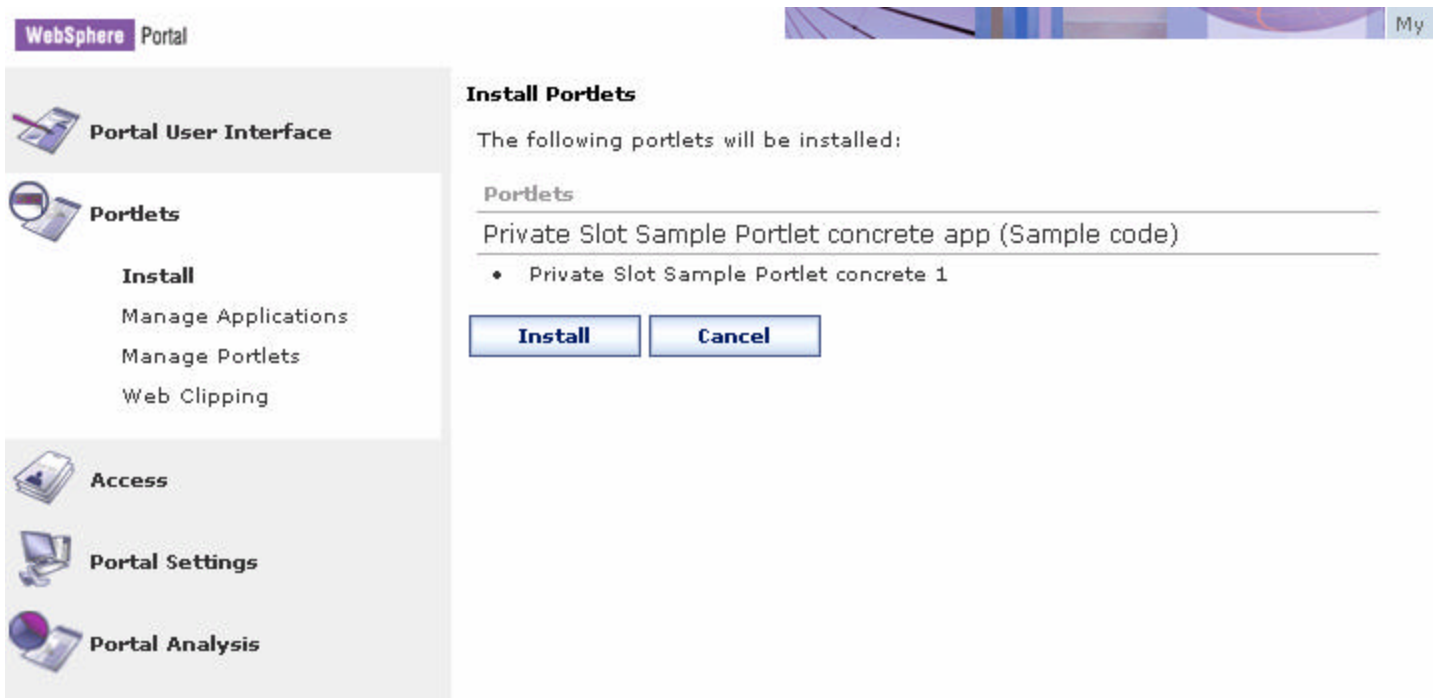
___ d) Navigate to **C:\Portlets** and select **privateslotsample.war**, assuming you extracted the download associated with this lab to your C drive.

___ e) Click **Open**.

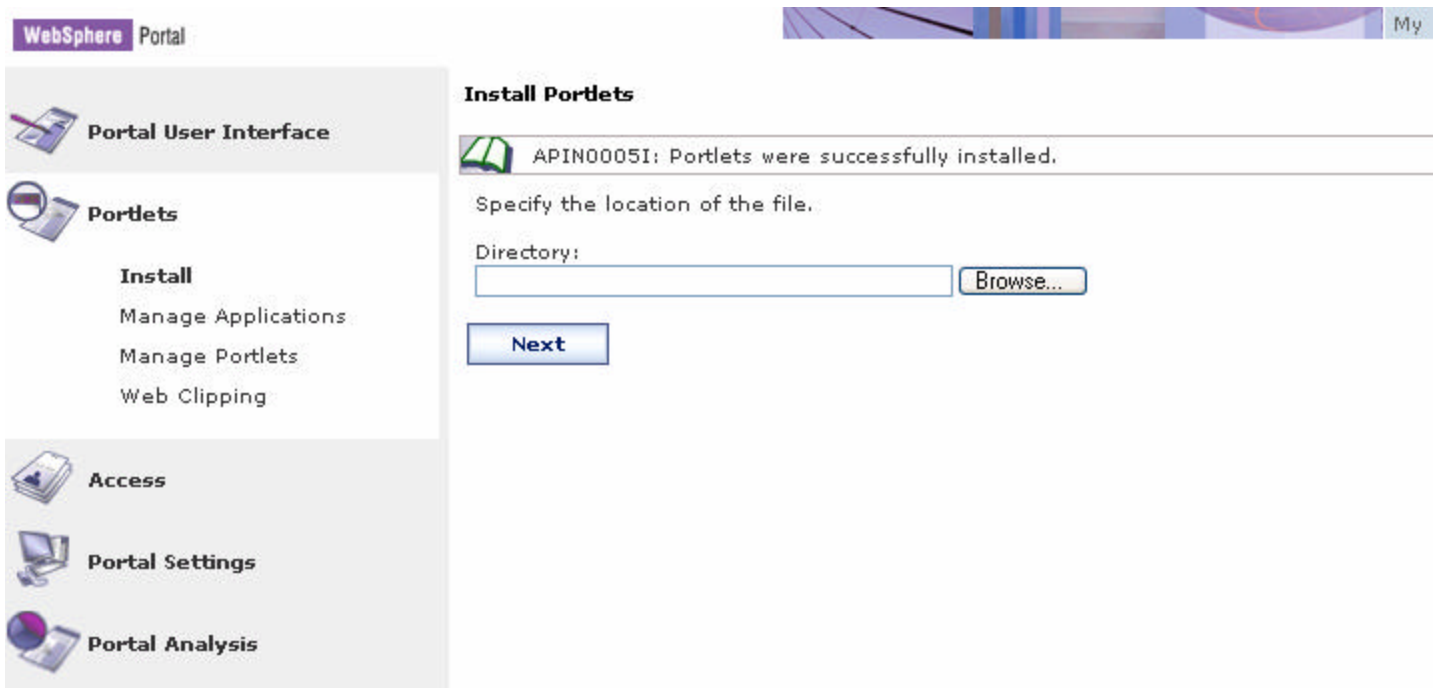


___ f) Verify the correct portlet has been chosen in the Directory field and click **Next**.

___ g) The subsequent screen shows the name of the Portlet application.



___ h) Click **Install**. The installation of the portlet will take a minute or so. You should obtain a message that confirms the successful installation.



- ___ 2. Next, you will create a page to exercise the private credential vault portlet that you just installed.
- ___ a) Click on **Portal User Interface** on the navigation menu.
 - ___ b) Next, click **Manage Pages**.
 - ___ c) Click on the **My Portal** label.
 - ___ d) Now click on the **New Label** button.
 - ___ e) Type **Credential Vault Lab** for the title. Leave the default selection for the theme.



WebSphere Portal

Page Properties

New Label: My Portal

Title:
Credential Vault Lab

Theme:
-----Inherit Parent Theme----- [v] [add]

I want to make this page my private page

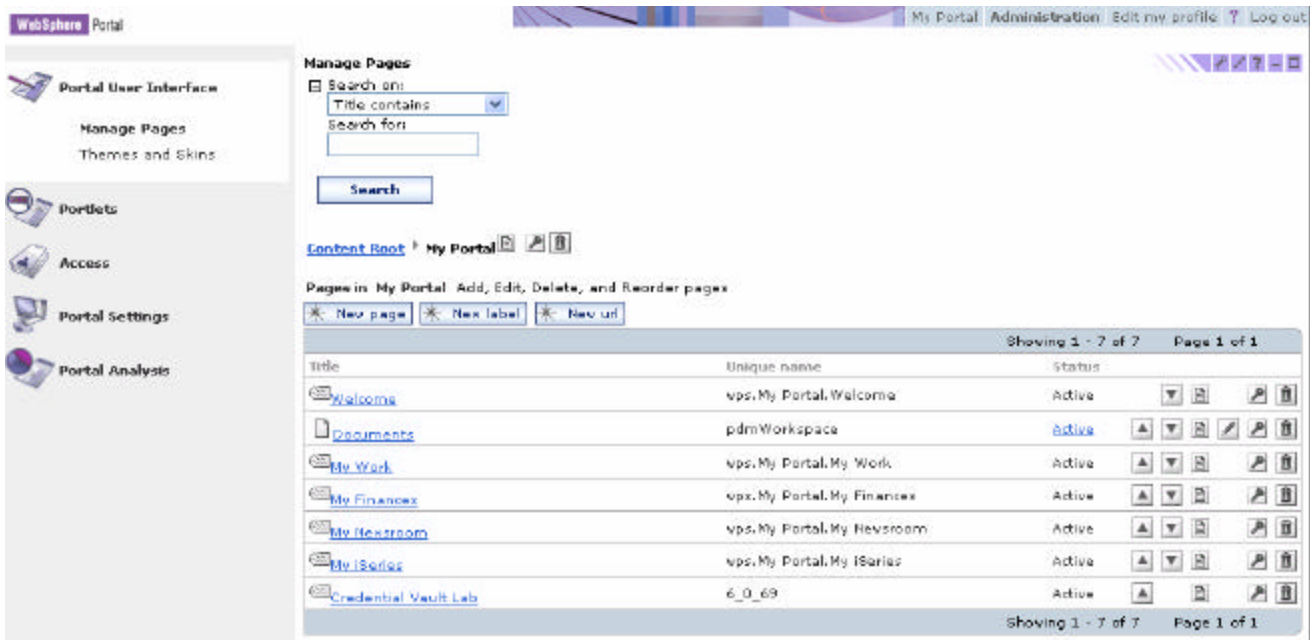
Advanced options

OK Cancel

- ___ f) Click **OK**. You will see the following confirmation message displayed indicating the page was created successfully.

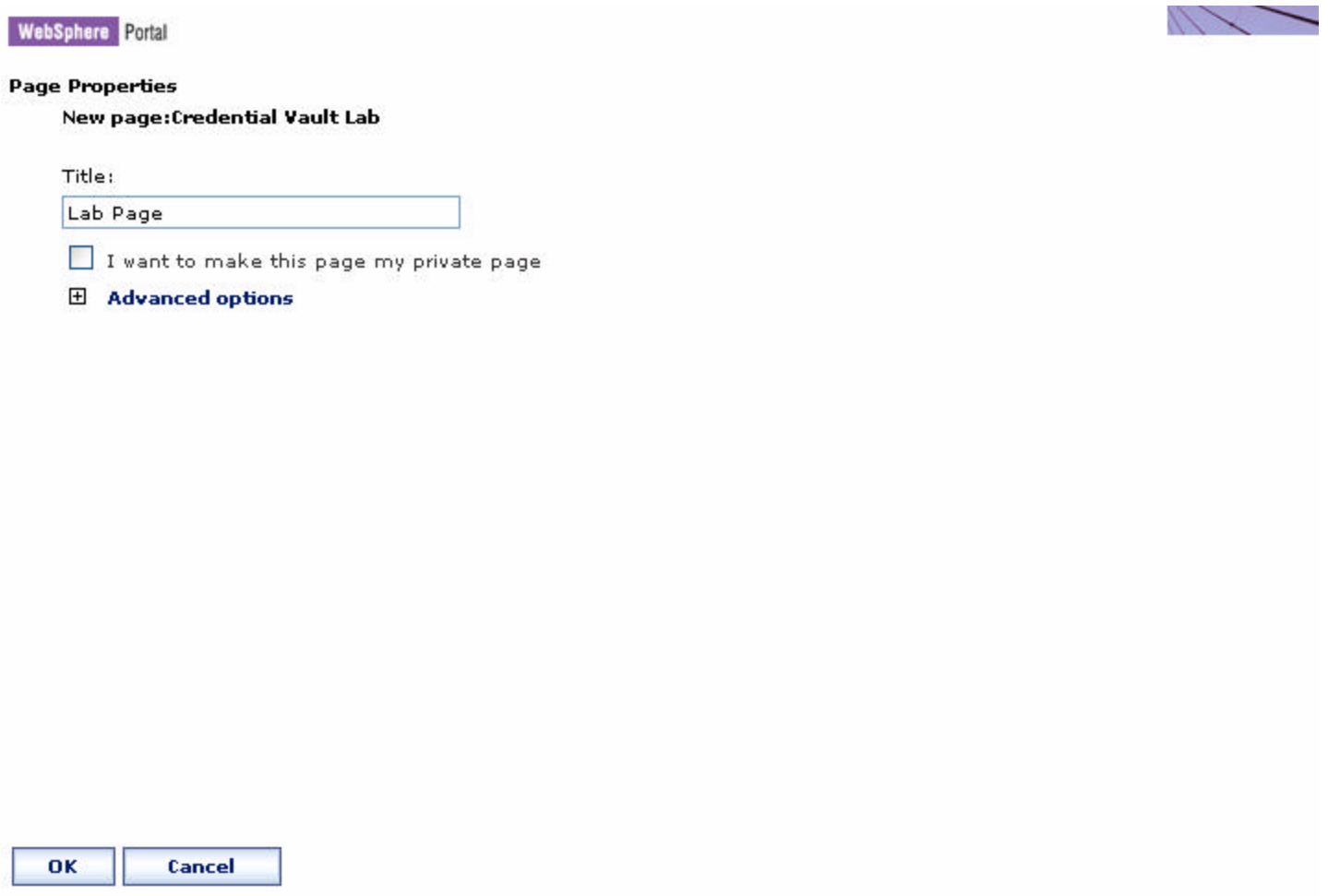


__ g) Click **OK** to acknowledge the confirmation message.



__ h) Click on the newly created **Credential Vault Lab** label.

__ i) Click on the **New Page** button to create a page for the Credential Vault Lab label.



WebSphere Portal

Page Properties

New page: **Credential Vault Lab**

Title:

Lab Page

I want to make this page my private page

Advanced options

OK **Cancel**

__ j) Type **Lab Page** in the Title field.

__ k) Click **OK**. You will receive a message notifying you of the successful creation of the new page.




WebSphere Portal

Page Properties

APPR0010I: Lab Page has been created successfully.

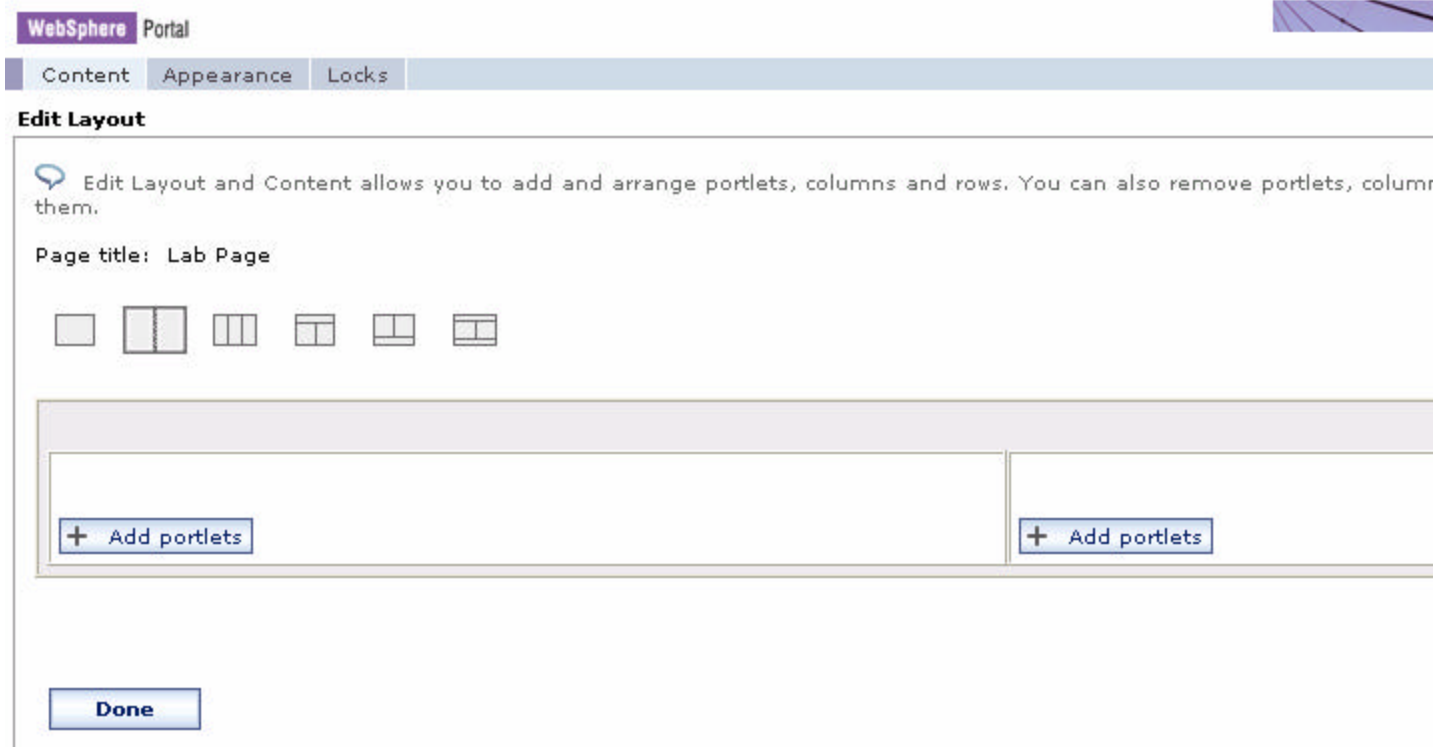
OK

__ l) Click **OK** to acknowledge the confirmation message.

__ m) Click the **Edit Page Layout** icon  next to the lab page that you just created.



__ n) Click on the **Add Portlets** button in the left pane of the lab page.



__ o) Type the word **Private** in the Search for field.

__ p) Click the **Search** button to locate the Private Slot Sample portlet.

__ q) Select **Private Slot Sample portlet** by checking the checkbox next to the portlet.

WebSphere Portal

Edit Layout

Search on:
Title contains
Search for:
Private

Search

Showing 1 - 1 of 1 Page 1 of 1

Select	Portlet Title	Unique name
<input checked="" type="checkbox"/>	Private Slot Sample portlet	

Showing 1 - 1 of 1 Page 1 of 1

OK Cancel

__ r) Click **OK** to complete editing the layout of the left pane.

__ s) Repeat the previous five steps for the right column or pane.

__ t) Your final page layout should look like the following.

The screenshot shows the 'WebSphere Portal' interface with tabs for 'Content', 'Appearance', and 'Locks'. The 'Edit Layout' section is active, displaying a message: 'APCL0115I: New portlets are added successfully.' Below this is a help tip: 'Edit Layout and Content allows you to add and arrange portlets, columns and rows. You can also remove portlets, columns and rows.' The page title is 'Lab Page'. A layout toolbar shows several icons for column and row configurations. The main workspace is divided into two columns. Each column contains a 'Private Slot Sample portlet' with a play button and a trash icon, and an 'Add portlets' button. A 'Done' button is located at the bottom left of the workspace.

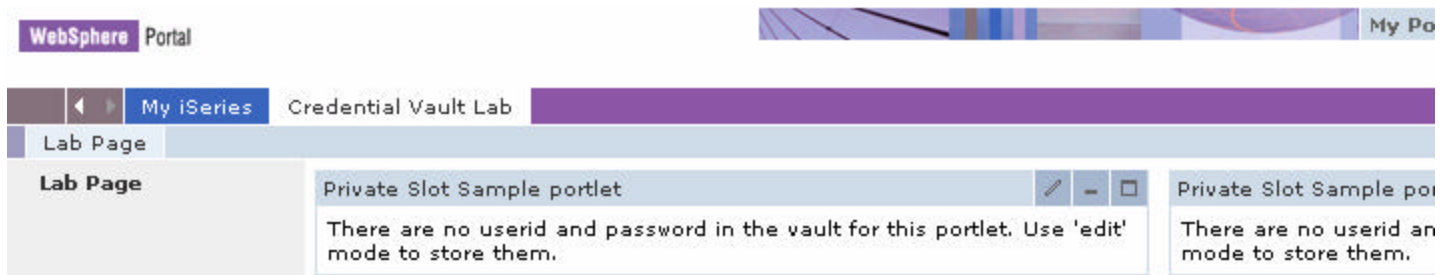
__ u) Click **Done** to complete editing the page layout of your lab page.


___ 3. In this next step, you will exercise the private credential slot portlet.

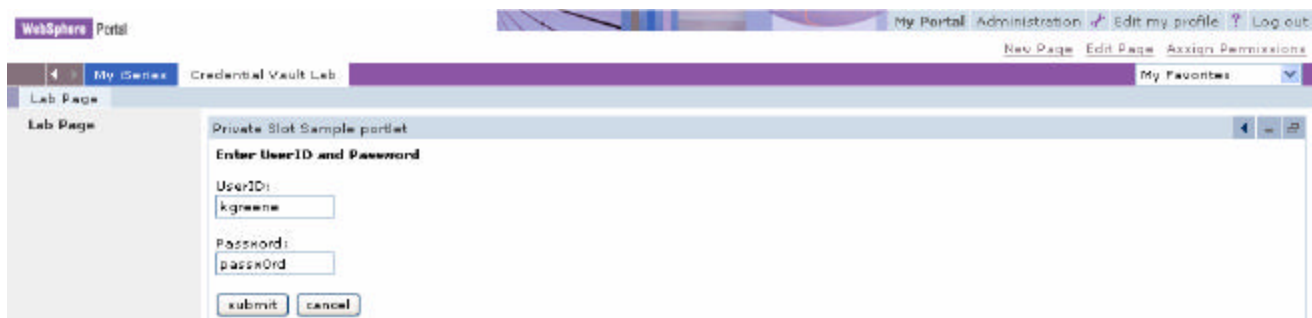
- ___ a) Click **My Portal** (“My Portal” at the very top of the page next to Administration – not “My Portal” the label).
- ___ b) Click on the **Credential Vault Lab label**. (NOTE: You may have to push the left and right scroll buttons to find the label. These scroll buttons are located to the left of the labels at the top of the screen.)



- ___ c) Click on the **Lab Page**. You should obtain a page where the two portlets display the message shown below.

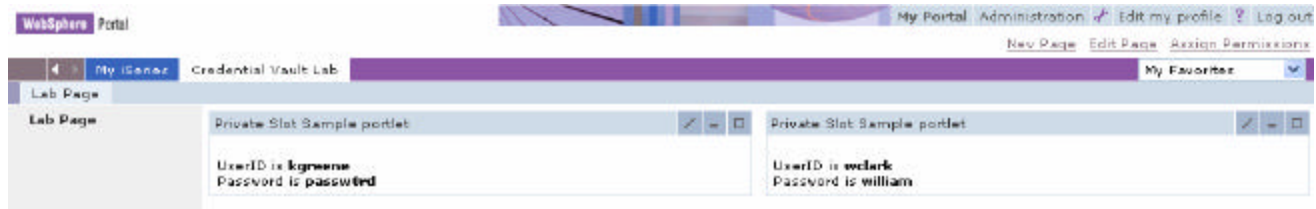


- ___ d) Click on the **Edit** icon  for the portlet in the left pane.
- ___ e) Set a user ID and a password of your liking on the subsequent screen. The user ID and password you enter do not have to be an existing user and password. They also do not need to be a defined portal profile or an operating system user. What you enter here will be stored in a private slot in the credential vault. Only this particular servlet and this particular user (wpsadmin) will be able to retrieve that slot.

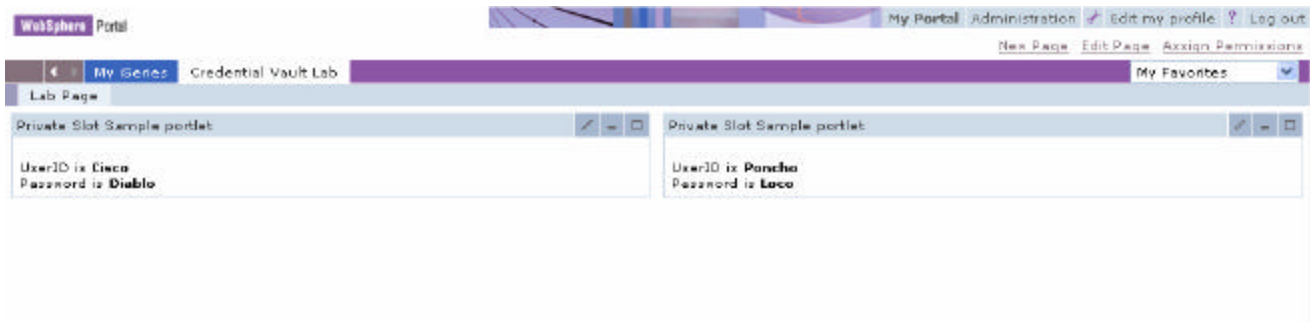


- ___ f) Click the **Submit** button. You will now see the portlet on the left display the user and password you entered.
- ___ g) Repeat the edit operations for the portlet on the right hand side.
- ___ h) You should end up with two portlet instances displaying two different user-password pairs.

WebSphere Portal Express –
An Introduction to enabling Single Sign-On to Backend Applications



- ___ i) **Log out** of portal.
- ___ j) **Log in** back in, but this time user id **User2** and its associated password. (NOTE: These both the user ID and password are case sensitive).
- ___ k) Click **Credential Vault Lab** label. You will see that the portlets do not show the users and passwords you set when running as wpsadmin. (You may see an error message because the username and password have not yet been stored. These can be ignored.)
- ___ l) Using the edit function, **set** a new pair of user ID/password combinations. Be sure to pick different user IDs and passwords than you used before. These users will be private to User2.

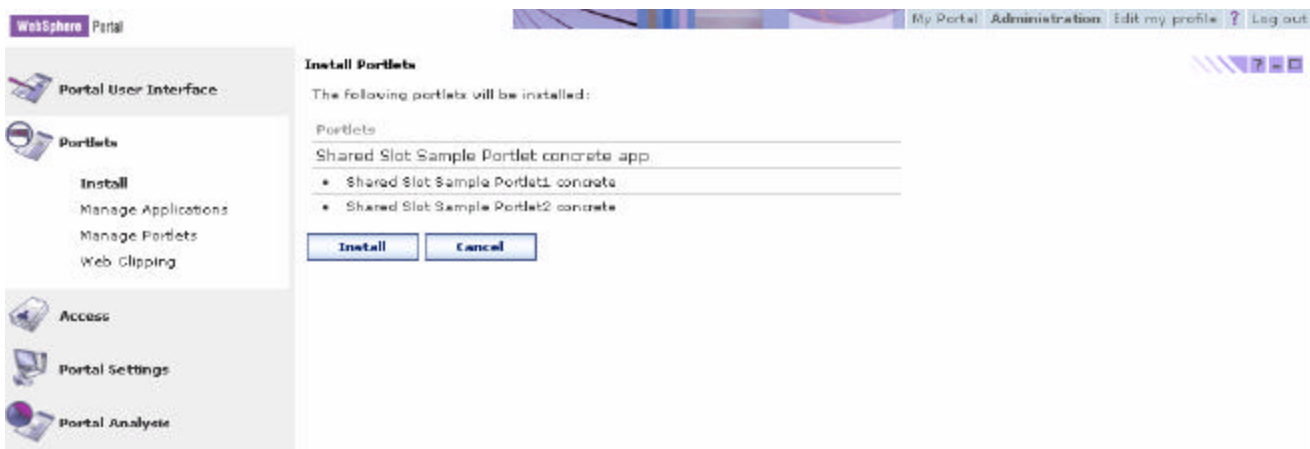


- ___ m) **Log out** of portal.
- ___ n) Now **Log in** again as wpsadmin.
- ___ o) Click **Credential Vault Lab**.
- ___ p) Click the **lab page**. You should see the users and passwords you set originally, when running as wpsadmin.

Completion of this step demonstrates the slots the portlet created are private to the user and to the specific portlet instance. For an analysis of the code that was used to code that made this example possible, see Appendix A at the end of this lab.

Part 3: Exercising the Shared User Slot

- ___ 1. In part 3 of this lab, you will install and configure the shared slot example, which demonstrates how portlets can share a credential slot.
 - ___ a) Make sure that you are logged into your WebSphere Portal server with the administration ID. We have been using **wpsadmin** as the ID in our examples.
 - ___ b) Click **Administration**.
 - ___ c) Next, click **Portlets** in the navigation menu.
 - ___ d) Click **Install**.
 - ___ e) Now, click the **Browse** button.
 - ___ f) Select file **C:\Portlets\sharedslotsample.war**.
 - ___ g) Click **Open**.
 - ___ h) Click **Next**.

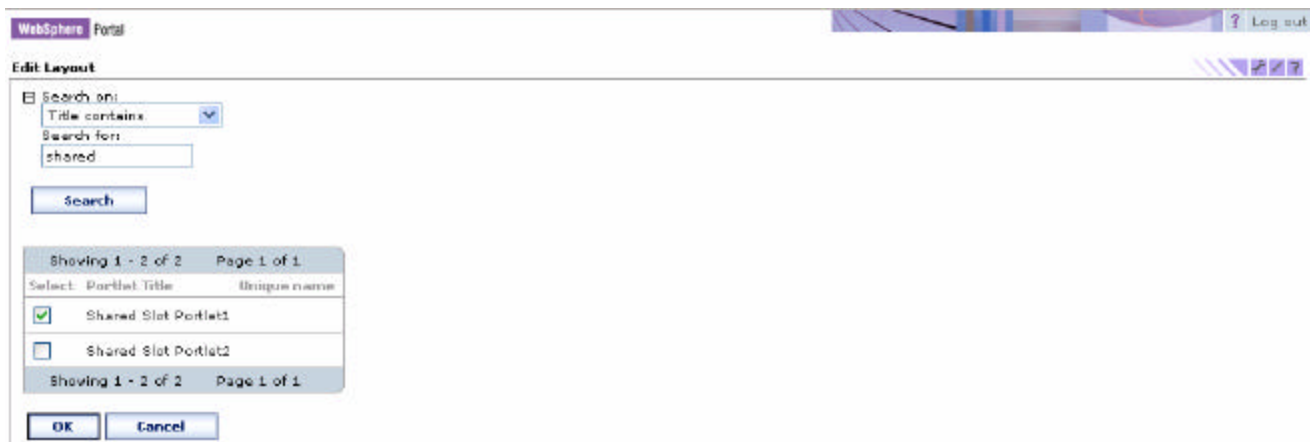


- ___ i) This WAR file contains two portlets that are exactly the same, in terms of the code they contain. Only the class name and the description are different. The objective of this section of the lab is to configure both of these portlets on the same page so that they can share the same slot.
- ___ j) Click **Install**. You should get a message that shows that the portlets were successfully installed.
- ___ k) Click on **Portal User Interface** in the navigation menu.
- ___ l) Click **Manage Pages**.

- ___ m) Navigate to the **Credential Vault Lab** label by clicking **My Portal -> Credential Vault Lab**. This will take you to a page that shows the lab page.

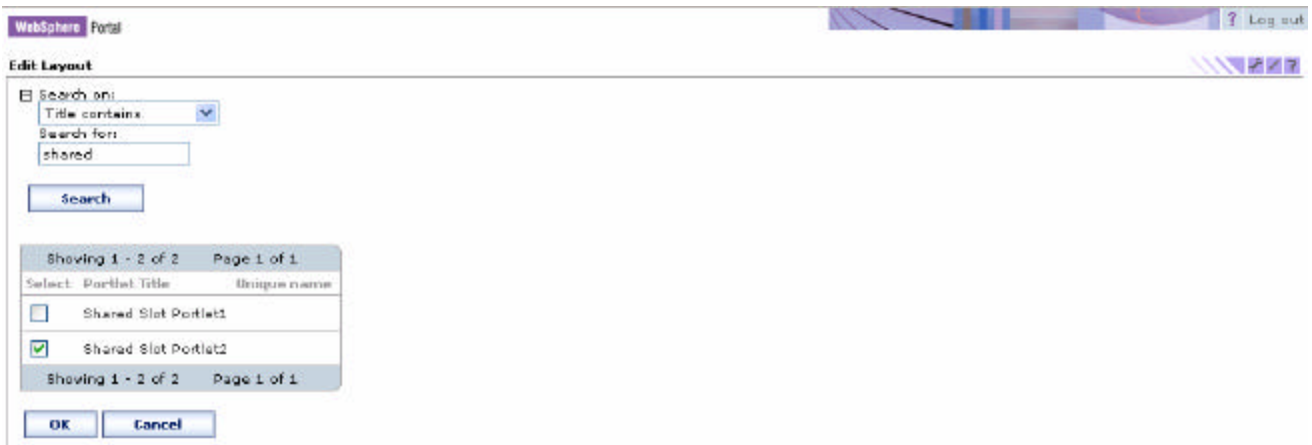


- ___ n) Click the **Edit Page Layout** icon (the pencil) to the right of the lab page. This brings you to the design page, where you can add portlets. The Private Slot Portlets should still be there.
- ___ o) Click the **Add portlets** button in the left pane.
- ___ p) Type **shared** in the search field.
- ___ q) Click **Search**. You will see a screen where the two portlets that you just installed can be selected.



- ___ r) Check the checkbox by **Shared Slot Portlet1**.
- ___ s) Click **OK** to complete adding the portlet to the left pane.
- ___ t) Click the **Add portlets** button in the right pane.

__ u) Repeat the search, this time selecting **Shared Slot Portlet2**.



__ v) Click **OK**.

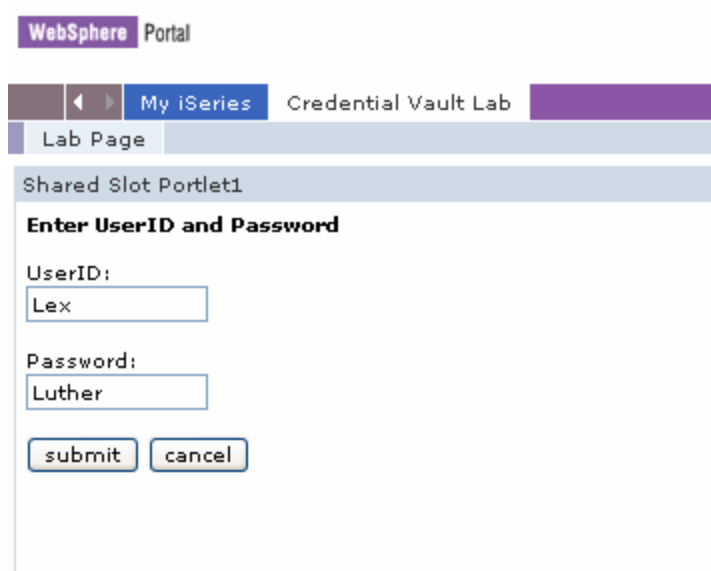
__ w) Click **Done** to complete editing the page layout.

___ 2. In this next step, you will exercise the shared slot portlets.

- ___ a) Click **My Portal** and navigate to your **lab page** by opening the credential vault. There should be four portlets on the page. The two portlets at the top should still show the values you entered when we ran the private slot example. The two new portlets we just added show the usual message about the slot not being available.



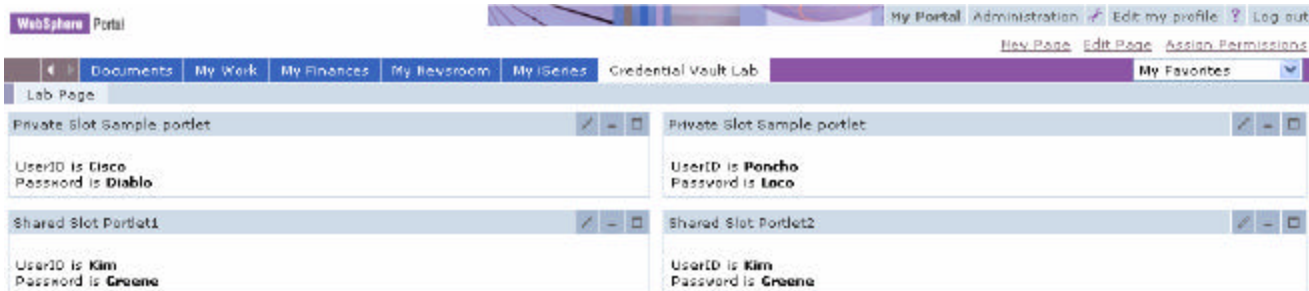
- ___ b) Click the **Edit** button for the Shared Slot Portlet1.
- ___ c) Fill in the user ID and password on the subsequent screen (choose values you have not utilized yet).



- ___ d) Click the **Submit** button. You should now notice that not only does the Shared Slot Portlet1 show the values you entered, but the Shared Slot Portlet2 does too. The secret is being shared between the two portlets. However, a new user will not be able to get to those credentials.



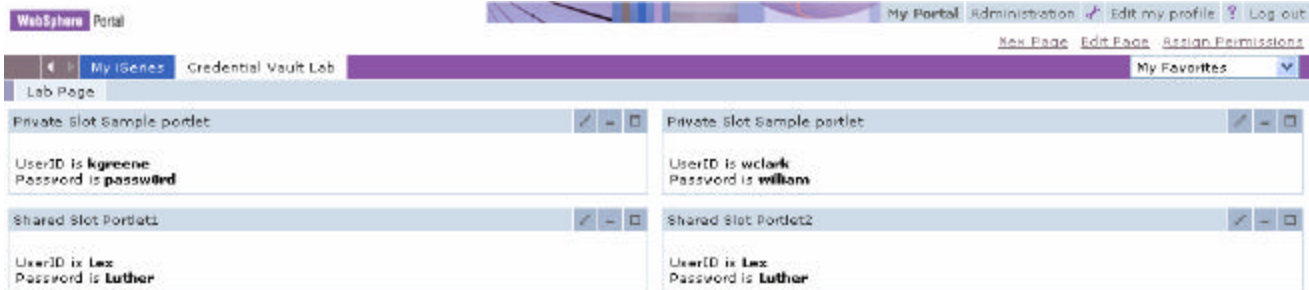
- ___ e) To see this effect, **log out** and **log in** as **User2**.
- ___ f) Click on **Lab page**. You will see that the Shared Slot portlets are not yet initialized. You will need to edit one of the Shared Slot portlets to store credentials for User2.
- ___ g) Click the **Edit** button for the Shared Slot Portlet2. Fill in the user ID and password on the subsequent screen (choose values you have not utilized yet).
- ___ h) Click the **Submit** button. You will see a screen similar to that below.



These credentials will be private to User2, but shared among the portlets on this page.

- ___ i) **Log out** and log back in as **wpsadmin**.

- ___ j) Click on **Lab Page**. The credentials you set as user ID wpsadmin will still be displayed by the Shared Slot portlets.



___ 3. Let's now analyze how the Shared Slot portlet is coded.

- ___ a) Using a text editor, such as Wordpad, open the file **SharedSlotSamplePortlet1.java** located in **C:\Portlets\credentialVaultSource\sharedSlot**. Notice there are two portlets in this directory. The only difference between the two Java files is limited to the class name. The rest of the code is exactly the same.
- ___ b) Inspect the **initConcrete()** method – notice this method provides the same function it did in the private slot portlet example, it caches the credential vault service:

- __ c) Inspect the **doView()** method next. This implementation is very similar to the private slot example – this method calls *getCredential()* and displays any credentials retrieved from the vault.
- __ d) Scroll down to the implementation of **getCredential()**. Notice the first difference from the private slot portlet. In the shared slot portlet, the *getCredential()* method does not look up a vault slot based on an identifier which is stored in the portlet request data. Rather, it calls a **getCredentialSlotConfig()** method, which iterates through all the potentially accessible shared slots in the credential vault and returns the one that matches the requested resource.

```
private CredentialSlotConfig getCredentialSlotConfig(PortletRequest
portletRequest)
{
    java.util.Iterator it = null;
    try{ // creates list of accessible slots
        it= vaultService.getAccessibleSlots(portletRequest);
    }catch(PortletException pe)
    {
        return null;
    }
    CredentialSlotConfig config = null ;
    String curResName = null;

    while(it.hasNext() )
    {
        config =(CredentialSlotConfig )it.next() ;
        curResName = config.getResourceName();

        //searches for shared resource name
        if (curResName.startsWith(resourceName ) )
            return config;
    }
    return null;
}
}
```

- __ e) The next major difference between the shared slot portlet and the private slot example is in the ***actionPerformed()*** method. With the shared slot example, the portlet creates a shared slot (notice that the *portletPrivate* parameter is set to false):

```
if(config==null) // create slot if it doesn't exist
    {
        ObjectID segmentID =
vaultService.getDefaultUserVaultSegmentId();
        Map descripMap = new Hashtable();
        Map keywordMap = new Hashtable();
        int secretType =
vaultService.SECRET_TYPE_USERID_STRING_PASSWORD_STRING;
        boolean active = false;
        boolean portletPrivate = false;

        //create the slot
        config = vaultService.createSlot(resourceName,
segmentID,descripMap,keywordMap,
secretType, active, portletPrivate, portletRequest);
    }
```

Part 4: Exercising the System Slot

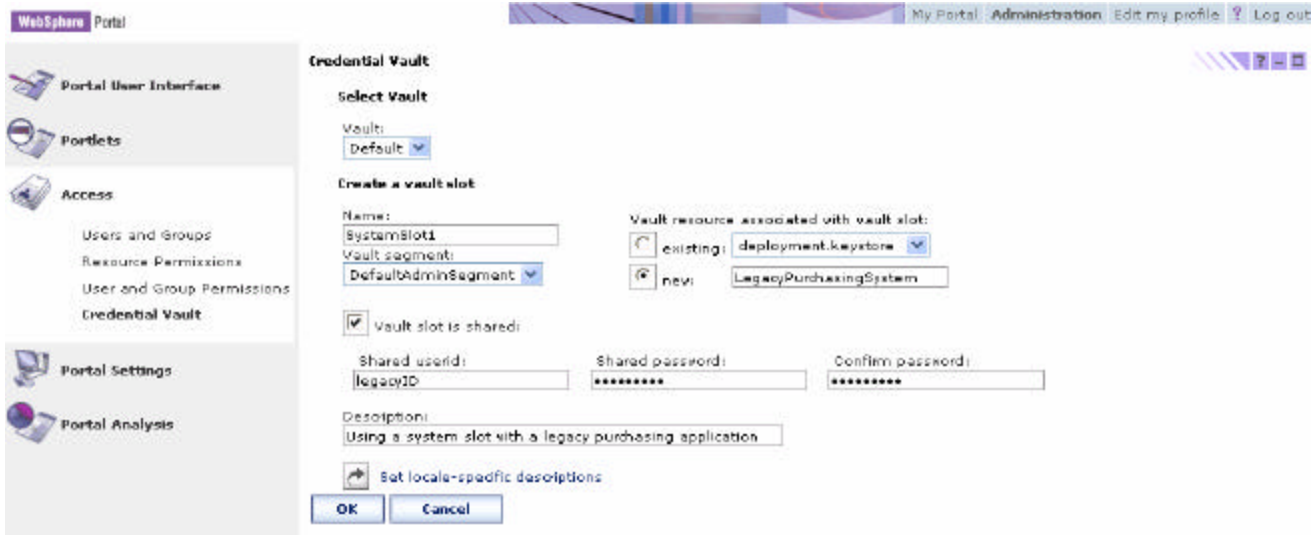
The examples we have worked with to this point in the lab created their own user slots in the credential vault. It is also possible for portlets to use slots that were created by the administrator. In particular, an administrator could create a **system slot** which will be available to all portlets, irrespectively of the user identity under which they are running. This function can be used whenever a number of Portal applications need to connect to a backend system using the same user ID. For example, your Portal applications and users may need to connect to a purchasing system that is implemented on a legacy system with all users needing to submit purchase orders using the same identity. This part of this lab will configure a system slot administratively, and then install a portlet that uses the slot.

- __ a) Make sure that you are logged into the portal server with a user ID that has administration rights, such as wpsadmin.
- __ b) Click **Administration** in the navigation bar in the upper right portion of the screen.
- __ c) Click the **Access** menu in the left navigation pane.
- __ d) Click **Credential Vault**. This is the “Credential Vault” administration portlet.



- __ e) Click **Add a Vault Slot** to create a slot in the vault segment. This vault slot will be used to store system credentials..
- __ f) Specify **SystemSlot1** in the name field.
- __ g) Check the box next to **Vault slot is shared**.
- __ h) Click the **New** radio button for the Vault resource associated with vault slot.
- __ i) Type **LegacyPurchasingSystem** in the text field next to the New radio button.
- __ j) Type **legacyID** for the shared user ID and **legacypwd** for both the shared password and confirm password fields.
- __ k) Type a description of your choosing.
- __ l) Leave all remaining parameters at their default values.

WebSphere Portal Express –
An Introduction to enabling Single Sign-On to Backend Applications



__ m) Click **OK** to complete creation of the system slot.

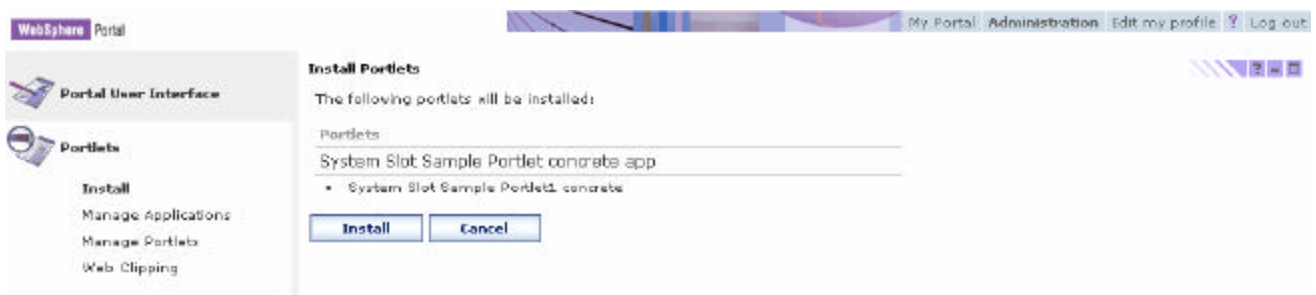
___ 4. Now, you will install a portlet that uses the system slot. This portlet will need to have the system definition in its deployment descriptor in order to utilize the system slot.

__ a) Click the **Portlets** menu in the left navigation pane.

__ b) Click **Install** and then **Browse** to select the **systemslotssample.war** file in directory **C:\Portlets\systemslotssample.war**.

__ c) Click **Open**.

__ d) Click **Next**. Your screen should be similar to the following.




__ e) Click the **Install** button to install the portlet. You should get a successful completion message.

__ f) Click **Portal User Interface** —> **Manage Pages**.

__ g) Navigate to **MyPortal** —> **Credential Vault Lab**.

__ h) You should see a page that allows you to work with your Credential Vault lab page.

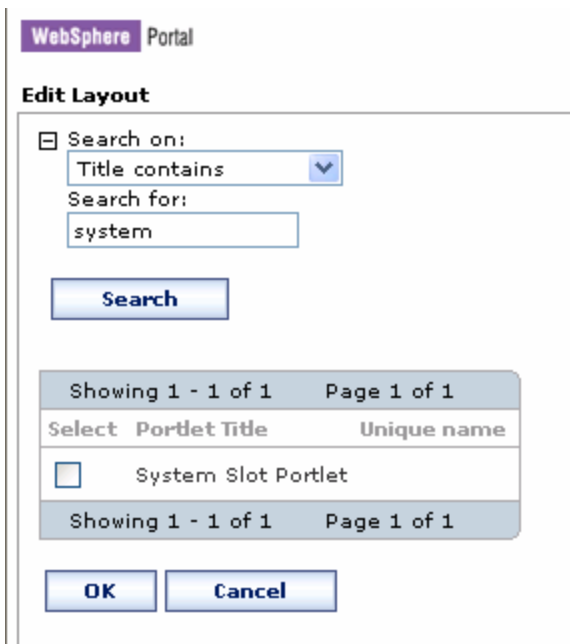


__ i) Click the **Edit Page Layout** icon, . This will take you to the design page. You will see the portlets that you have previously added.

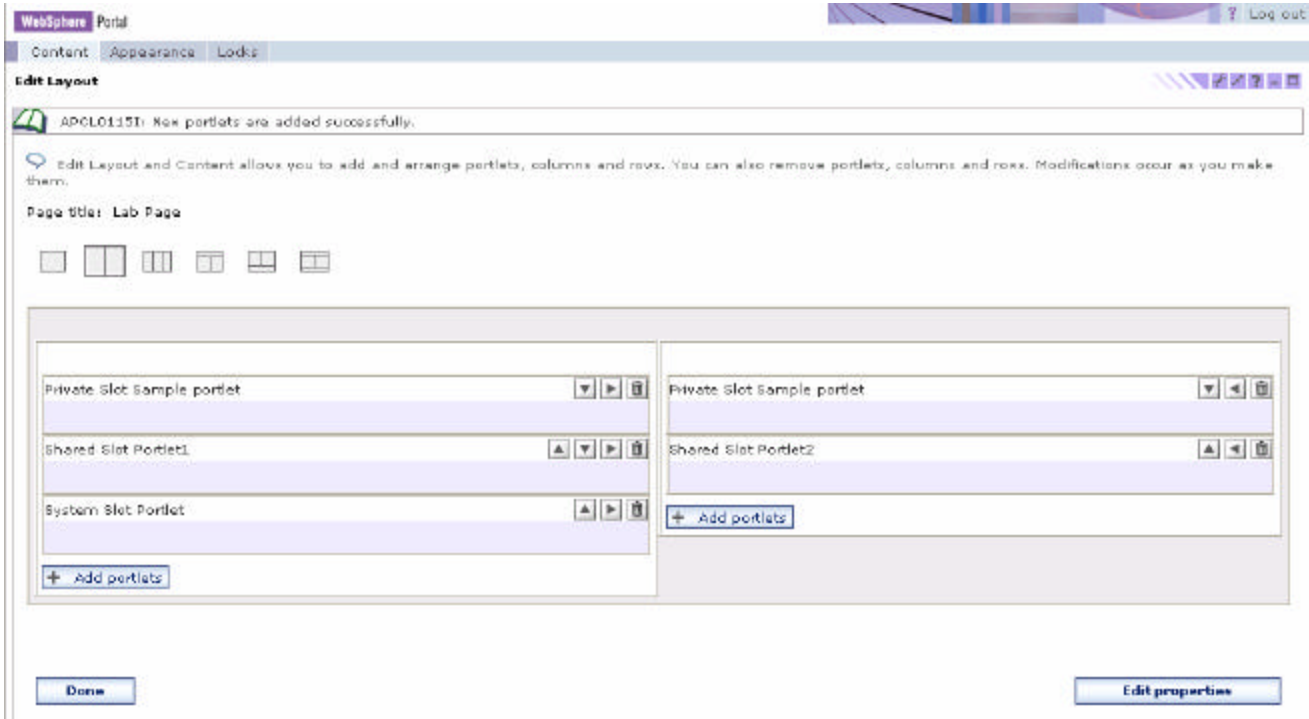
__ j) Click the **Add portlets** button in the left pane. You will be adding only one instance of the portlet.

__ k) Type **system** in the **Search for** field.

__ l) Click the **Search** button.



__ m) Check the checkbox next to **System Slot Portlet** and click **OK**.



__ n) Click **Done**.

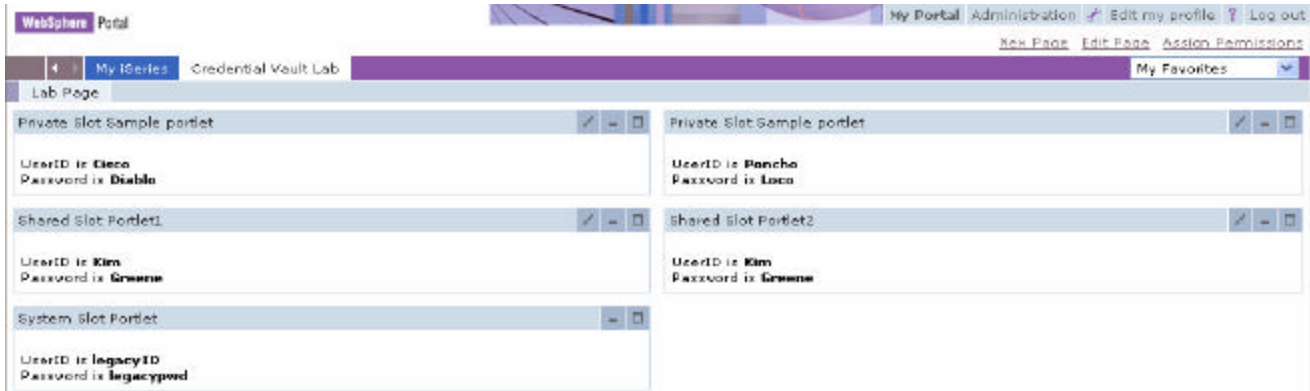
___ 5. You are now ready to exercise the portlet.

__ a) Click **MyPortal** → **Credential Vault Lab**. You should now see five portlets. The top four portlets show the credentials you set in parts 2 and 3 of this credential vault lab. The last portlet you just added shows the credentials which the administrator set in the system vault slot.



__ b) Next, we want to see that the system vault slot credentials are available to all portal users. **Log out** and then **log in** again, but this time log in as **User2**.

__ c) Click **MyPortal** → **Credential Vault Lab**. You will see the System Slot Portlet showing the shared credentials.



___ d) **Log out** and close your browser.

___ 6. In this step, we will inspect the System Slot portlet.

___ a) Using a text editor such as Wordpad, open the file **SystemSlotSamplePortlet.java** located in **C:\Portlets\credentialVaultSource\systemSlot**.

___ b) Notice this portlet is significantly more simple than the previous two portlets we analyzed. In particular, it does not need to implement the *doEdit()* and *actionPerformed()* methods. These methods are not required because it does not set the credentials in the slot. The slot and the credentials are set by the administrator.

___ c) The ***initConcrete()*** method performs the usual caching of the vault service. In addition, it retrieves the system slot name from the application settings. This parameter must be statically defined in the deployment descriptor.

```
systemVaultSlotName =  
settings.getApplicationSettings().getAttribute("SystemVaultSlotName");
```

___ d) The ***doView()*** method still calls ***getCredential()***, like in the previous two cases.

___ e) The ***getCredential()*** method very simply retrieves the credential from the system slot that was retrieved during the *initConcrete()* method execution:

```
private void getCredential(PortletRequest portletRequest,  
    StringBuffer userid,  
    StringBuffer password) throws PortletServiceException{  
    try{  
        UserPasswordPassiveCredential credential  
        =(UserPasswordPassiveCredential) vaultService.getCredential(systemVaultSlotName ,  
        "UserPasswordPassive", new HashMap(), portletRequest);  
        userid.append(credential.getUserId() );  
        password.append( String.valueOf(credential.getPassword() ) );  
    }  
  
    catch(com.ibm.wps.portletservice.credentialvault.CredentialSecretNotSetExce  
ption e){  
        return ;  
    }  
}
```

___ 7. Close the text editor you were using to view the system slot source code.

Summary

This lab illustrated the differences in functionality of private slots, shared slots, and system slots in the credential vault.

You have exercised this functionality of each of these slots by installing a number of portlets which created or used credential vault slots to store and retrieve credentials. You also experienced the different scopes of visibility offered by the various types of slots.

Appendix A: Analysis of Private User Slot code

Let's briefly analyze the code that made this example possible.

- ___ a) Using a text editor (i.e. Wordpad), open the file **PrivateSlotSamplePortlet.java** located in the **C:\Portlets\credentialVaultSource\privateSlot** folder, assuming this is the directory you have placed the .java file in.
- ___ b) Notice that the **initConcrete()** method caches the credential vault service:

```
public void initConcrete(PortletSettings settings) throws UnavailableException
{
    super.initConcrete(settings);
    try{
        vaultService = (CredentialVaultService)
getPortletConfig().getContext().getService(CredentialVaultService.class);
    }
    catch(Exception e){
        return;
    }
}
```

- ___ c) Scroll down and look at the **doView()** method. This method retrieves the credentials that are stored in the private slot. It does so by calling the **getCredential()** method.

```
getCredential(portletRequest, userid, password);
```

- __ d) The **getCredential()** method extracts the slot ID from the PortletData object. It does this by getting the slotIDKey for the current user. Once the slotIDKey has been retrieved, the slotID is retrieved for the slotIDKey. Provided a slotID is returned, the passive credentials of user ID and password are returned.

```
private void getCredential(PortletRequest portletRequest, StringBuffer
userid, StringBuffer password){
    String slotIDKey =
"PrivateSlotSamplePortletSlotID"+portletRequest.getUser().getUserID();
    try{
        String slotId = (String)
portletRequest.getData().getAttribute(slotIDKey);

        if(slotId==null)
            return ;

        UserPasswordPassiveCredential credential
=(UserPasswordPassiveCredential) vaultService.getCredential
(slotId, "UserPasswordPassive", new
HashMap(), portletRequest);
        userid.append(credential.getUserId() );
        password.append( String.valueOf(credential.getPassword() ) );
    }

    catch(com.ibm.wps.portletservice.credentialvault.CredentialSecretNotSetExce
ption e){
        return ;
    }
    catch(PortletServiceException e){
        return ;
    }
}
}
```

- __ e) The method that actually populates the credential vault slot is the **actionPerformed()**. This *method* gets called when the user presses the **submit** button in the JSP that is associated with the "edit" action. This is the key method in this example. This method first extracts the user ID and password from the Portlet request:

```
String userID    = (String) portletRequest.getParameter("userID");
String password  = (String) portletRequest.getParameter("password");
```

- __ f) The method then tries to store those credentials in the private slot. If the slot does not exist yet, the method will create it. Notice that a simple user ID/password passive credential slot and that the slot is private are created.

```
PortletData data = portletRequest.getData();
String slotId = (String) data.getAttribute(slotIDKey);

if(slotId==null) { // create slot if necessary

    String resourceName = "POP3MailApp";
    ObjectID segmentID = vaultService.getDefaultUserVaultSegmentId();
    Map descripMap = new Hashtable();
    Map keywordMap = new Hashtable();
    int secretType = CredentialVaultService.SECRET_TYPE_USERID_STRING_PASSWORD_STRING;
    boolean active = false; // passive credentials
    boolean portletPrivate = true; // Private slot
    //create the slot
    CredentialSlotConfig slot= vaultService.createSlot(resourceName, segmentID,
        descripMap, keywordMap, secretType, active, portletPrivate, portletRequest);

    slotId=slot.getSlotId();
    data.setAttribute(slotIDKey, slot.getSlotId());
    data.store(); //saving the credentials
```

- __ g) Close the source code example you have open in Wordpad.

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries: WebSphere, IBM, eServer, iSeries

All other products may be trademarks or registered trademarks of their respective companies.