



WebSphere. software

WebSphere Portal and Microsoft SharePoint Integration Guide

October, 2004

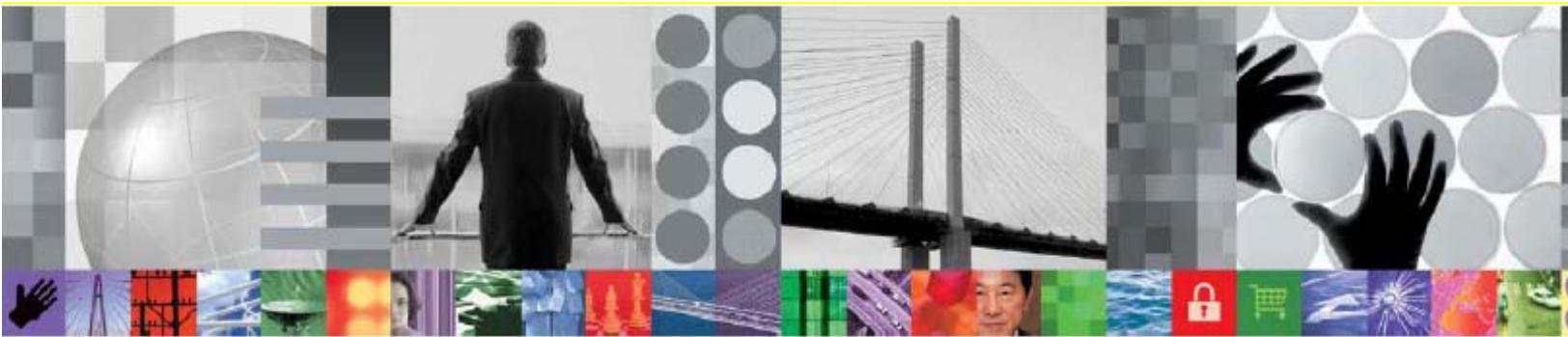


Table of contents

Abstract	1
Introduction.....	2
Audience	2
Sample portlets	2
Understanding überportals.....	3
Picking the überportal server	3
Integration points.....	4
Security	6
About WebSphere Portal security	6
Microsoft SharePoint security	9
Authentication options and recommendation.....	11
Authorization options and recommendation.....	14
Portlets and Web Parts	17
WebSphere Portal portlets	17
SharePoint Web Parts.....	17
Portlet standards	19
Portlet and Web Part options	21
Recommendation	27
Search	28
WebSphere Portal search overview	28
SharePoint search overview	29
Options and Recommendation.....	29
Personalization	33
WebSphere Portal personalization and customization	33
SharePoint personalization and customization	35
Options.....	36
Recommendations	38
Document management	39
WebSphere Portal document management.....	39
SharePoint document management.....	40
Options.....	40
Recommendation	44
Presence	45
WebSphere Portal awareness	45
SharePoint awareness	45
Options.....	46
Recommendation	48
Conclusion.....	49
Resources	50
About the authors	52

Abstract

Portal interoperability involves more than enabling portlets from one portal to run in another portal. The differences between the IBM and Microsoft portal frameworks, and between the J2EE and .NET programming models, make it impractical to embed SharePoint Web Parts within a portal running under WebSphere Portal portals, and vice versa.

Therefore, integration and interoperability of portals based on these two technologies must be approached from a number of perspectives. This document discusses six key areas which must be considered when implementing a full fledged federated portal, comprised of portals running under IBM® WebSphere® Portal and portals running under Microsoft® Windows® SharePoint™ Services and Microsoft Windows SharePoint Portal Server. Most of the solutions described here use custom code that exploit SharePoint's Web services API. Some use a shrink-wrap product and infrastructure solution. The sample code provided is intended to illustrate an approach; it is not intended to be production level code. This article also includes an accompanying set of portlets which provide generic access to SharePoint content.

Introduction

This article uses the term *SharePoint* to refer to SharePoint Products and Technologies, which is the combination of Microsoft® Windows® SharePoint™ Services (hereafter called SharePoint Services) and Microsoft Windows SharePoint Portal Server. SharePoint Services is embedded in Windows Server 2003, and provides a collection of services that support SharePoint Portal Server, Microsoft Office 2003, and Project Server 2003. SharePoint Portal Server is a separate, chargeable product which provides a portal infrastructure, and a collection of *Web Parts* (the Microsoft term for portlets) and a search engine.

Over the past few months, we have received numerous requests for advice on integrating IBM® WebSphere® with SharePoint. In the majority of these requests, the client had an installed base of departmental SharePoint Services and Team Services sites which had been installed in an unregulated piecemeal fashion. When the time came for IT to deploy an enterprise-wide portal solution, there was a need to integrate these individual sites into the overall portal fabric. Integrating portals is not merely a function of allowing Web Parts to be consumed by the federating portal.

“The emergence of portlet standards — Web Services for Remote Portlets (WSRP) and Java Specification Request 168 (JSR168) — in 2H03 will do little to provide the level of interoperability required.” – Gartner
Sept 2003

We are confronted with the need to integrate the data, documents, metadata, search indexes, security credentials, personalization settings, and other information that reside within each of these team sites into WebSphere Portal. For each integration aspect discussed in this article, we provide several options and then recommendations for the most suitable approach.

Audience

This article is for software engineers, architects, and technical management. A knowledge of Java, and the XML DOM is necessary to understand the code samples, but not required to achieve an understanding of the approach or its application.

Sample portlets

In addition to the inline code samples, you can download two portlets from the WebSphere Portal and Lotus Workplace catalog. You might want to reference the code as you read through the article. See [Resources](#) for links to download the `SharePoint Recent Documents` portlet and `SharePoint Events` portlet.

Understanding überportals

Until an industry-wide portal interoperability standard is in place, enterprises must take a tactical approach which requires some degree of bespoke programming to act as the glue. One common tactical approach is the *überportal*.

The überportal is a high level, horizontal portal on top of one or more horizontal or vertical portals. The überportal is the entry point in a multiportal deployment. It is the place where users spend most of their time and where most of the content is aggregated.

A minimalist überportal approach to integration would have these characteristics.

- Rationalized directory services (common or replicated directory)
- Single Sign-On (SSO) to all related portals
- Log-on which points the user to a personalized start page in the überportal
- Access to lower-level portals using hyperlinks or tabs on their start page
- When a user clicks on the link to the lower-level portal, his or her personalized start page in the lower-level portal is rendered, typically in a separate window
- When a user finishes tasks in the lower-level portal, he or she closes that window and returns to the überportal

Picking the überportal server

When considering an enterprise portal, senior technical management is often faced with a number of technologies that must be integrated and surfaced through the portal. When SharePoint Services is one of those technologies, the seemingly easy answer to the enterprise portal question is SharePoint Portal. When approached from the other side, though, the need for a secure Internet or extranet presense, translation, pervasive device support, and similar robust, enterprise-level technologies will usually lead one to WebSphere Portal.

Ironically, greater value is recognized by leveraging SharePoint Services through WebSphere Portal rather than through SharePoint Portal Server. Surfacing SharePoint Services through WebSphere Portal creates opportunities for dynamic translation, transcoding, "Everyplace" access, mixing and matching EJBs and Web Services, and countless other benefits of the IBM OnDemand Workplaces architecture.

Figure 1 shows how these features are integral to the WebSphere Portal architecture.

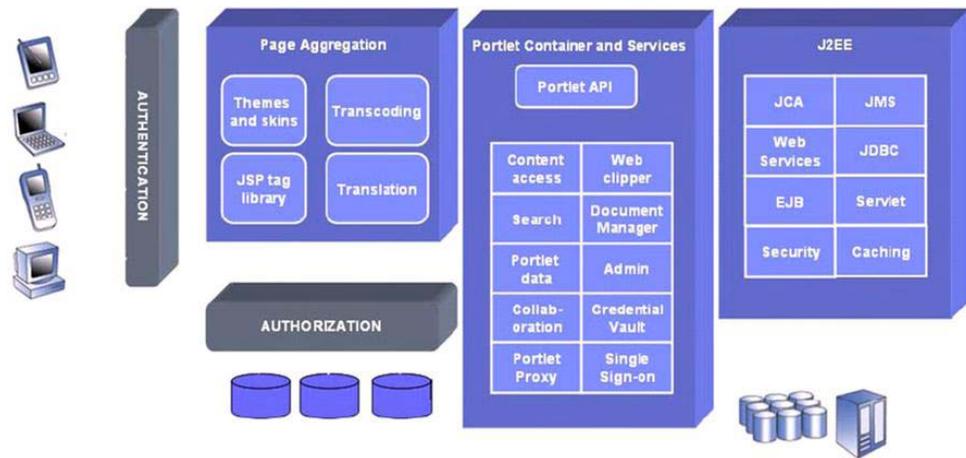


Figure 1: WebSphere Portal architecture

Integration points

A portal is composed of layers of components, each providing a services. Integration among portals must take into account each of these services interfaces, which could be categorized into the six integration points described below. The rest of this article discusses the integration points, including the issues, merits, and problems associated with each. Each discussion includes alternative approaches and recommendations for addressing the integration point.

Figure 2 represents WebSphere Portal, acting as the überportal, hosting the presentation and business logic layers of the solution, integrating with Windows SharePoint Services in six key areas.

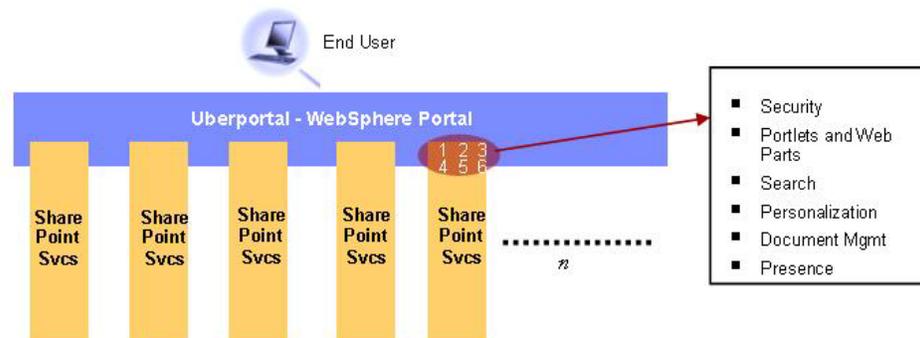


Figure 2: Six Points of Integration with SharePoint Services

Security

Users must be presented with a seamless experience when they use the überportal. They should not be required to re-register with a new userid and password, nor should they be challenged for their credentials multiple times when a subportal needs to retrieve content. In

this section, we discuss how to synchronize userids and passwords across portals, and how to pass credentials from the überportal to the subportals when accessing protected resources.

Portlets and Web Parts

In this section, we explore a topic which is typically one of the first questions that surfaces when we are asked about interoperability: Can WebSphere Portal host SharePoint Web Parts? The short answer is no, because of differences in technology implementations. Therefore, we outline several methods for mimicking SharePoint Web Parts using portlets.

Search

The user should not be required to perform multiple, separate searches in order to query each portal. Ideally, results from all portals should be returned when the user issues a search query within a portlet. We discuss one way to implement this. We also discuss how to write a custom portlet that queries SharePoint content only.

Personalization

The ability to leverage an individual's personalized or customized settings that have been set in a subportal is a desirable feature in an integrated portal. Unfortunately, implementing this support is not a straightforward endeavor because SharePoint does not expose the API's to access these settings easily. We summarize the personalization and customization capabilities of the two frameworks and suggest approaches for moving from SharePoint Portal to WebSphere Portal.

Document management

One of the primary uses of SharePoint is as a document library. It provides an attractive and easy-to-use interface that provides value-added functionality over a simple file share. An überportal should be able to provide access to the documents in SharePoint in a seamless fashion.

Presence

Most modern portals provide the ability to surface presence awareness, or the ability to determine the online status and other contact information of individuals, within a page. This section describes a scenario in which a client deploys WebSphere Portal as the überportal but decides to retain the existing Microsoft Instant Messaging infrastructure (Live Communications Server and Messenger). We explore several options to expose Microsoft presence awareness within a portlet on a page.

Security

The objective for the überportal is that each user will log in a single time with an assigned userid and password. From then on, the überportal should pass the credentials on to any other portals, which can, in turn retrieve the appropriate data. The user should not be required to enter credentials again, whether or not they are the same values.

About WebSphere Portal security

Security is concerned with authentication, authorization, administration, and audit. This section focuses on authentication and authorization.

A u t h e n t i c a t i o n

WebSphere Portal uses form-based authentication, in which a user is prompted by an HTML form for his or her user ID and password, whenever he or she tries to access the portal. The portal server requests the application server to validate this authentication information against a Lightweight Directory Access Protocol (LDAP) user registry.

WebSphere Application Server uses Lightweight Third Party Authentication (LTPA) as the authentication mechanism. A Common Object Request Broker Architecture (CORBA) Credential is used to represent authenticated users and their group memberships. When a user tries to access a protected resource, the application server intercepts the request and redirects the request to the login form. This form posts the user ID and password to the portal which requests the application server to authenticate the user. If the user can be authenticated, a valid CORBA credential is created and an LTPA cookie is stored on the user's machine.

If your system uses another third-party authentication server, trust needs to be established between that proxy and WebSphere Application Server. This is done using a Trust Association Interceptor (TAI) module, which converts security information specific to the authentication proxy into a format that can be handled by the application server. The supported authentication mechanism depends on the capabilities of the third-party product. When a user tries to access the portal, the third-party authentication proxy intercepts the request and challenges the user to authenticate. After a successful login, the original user request, along with additional security information in the request header, is passed to the application server.

The format and content of this information is vendor specific. WebSphere Application Server uses the TAI module (that is specific to the third-party product) to extract the necessary security information from the request header. The TAI module for IBM Tivoli Access Manager is packaged with the portal server. See the WebSphere Application Server InfoCenter for information about creating custom TAI modules for other third-party reverse proxy servers.

Authorization

After determining the user's identity, the portal server consults locally cached access control lists to determine which pages and portlets a user has permission to access. The portal server enforces access control to portal assets, including portlets, pages, and user groups. The access control lists are stored in the portal's administration database. It is also possible to manage access control for specific resources in an external security manager, such as IBM Tivoli Access Manager or Netegrity SiteMinder.

Access permissions are maintained using the Access Control administration portlet. The administrator uses this portlet to assign roles to individual users or to groups of users for specific portlets, pages, or documents. Roles are permission sets, such as the ability to view and update the corresponding item. Users may also delegate the permissions they hold to other users. When a role is assigned to a user or a group on a container (such as a page that contains portlets or other pages, or a folder that contains other folders or documents), that role is inherited downward through the hierarchy unless it is specifically blocked.

Resource Permissions 

Assign Access for: Pages --> Health Care
Click the Edit Role icon to view, add or delete members.

Roles	Allow Propagation	Allow Inheritance	Edit Role
Administrator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Security Administrator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Delegator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Manager	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Editor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Privileged User	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
User	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Showing 1 - 7 of 7 Page 1 of 1

[Display/Modify Owner](#)

Figure 3: Assigning resource permissions in WebSphere Portal

Single sign-on

The portal server provides comprehensive single sign-on (SSO) support. Users want to be able to log on only once, and be known to the different parts of the portal server with the same consistent user credentials. Users should not be asked to do multiple logons simply because they access different portal applications. The portal server supports single sign-on *realms* using WebSphere Application Server as well as authentication proxies. The user needs to log on only once to gain access to all enterprise applications that are installed within the single sign-on realm.

WebSphere Application Server uses Lightweight Third Party Authentication (LTPA) tokens to provide single sign-on. When a user is authenticated, the WebSphere Portal creates an LTPA single sign-on cookie containing the authenticated user credential. This encrypted

cookie conforms to the format used by WebSphere Application Server and can be decrypted by all application servers in the shared domain, provided they all have the same cipher key. This cookie enables all servers in the cluster to access the user's credentials without additional prompting, resulting in a seamless single sign-on experience for the user. To benefit from the LTPA method of single sign-on, the user's browser must support cookies and have its support for session cookies enabled.

Many portlets need to access remote applications that require some form of user authentication. For accessing applications outside the portal's realm, WebSphere Portal provides a credential vault service that portlets can use to store user ID and password (or other credentials) for a user login to an application. Portlets can use these on behalf of the user to access remote systems. The credential vault supports either local database storage or IBM Tivoli's Access Manager for secure storage and retrieval of credentials.

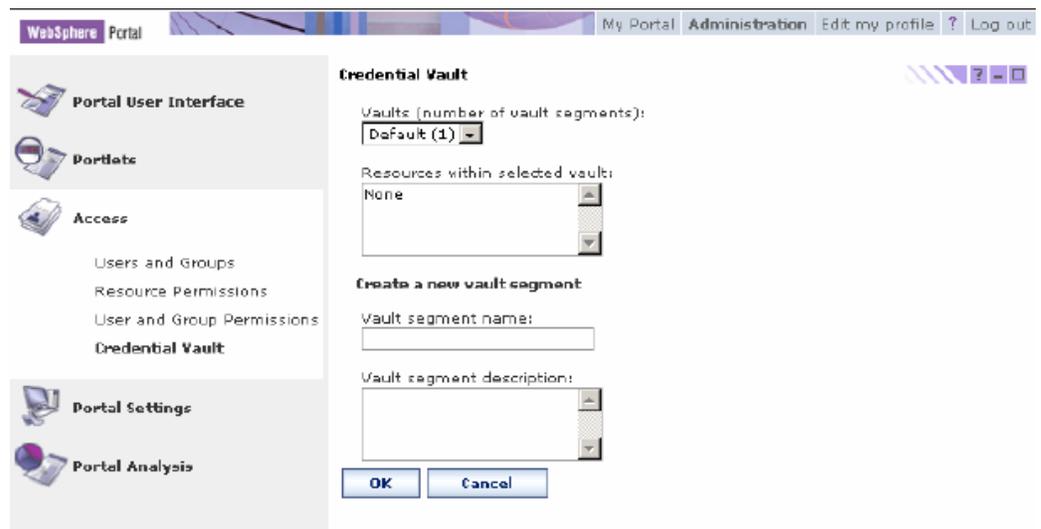


Figure 4: The WebSphere Portal Credential Vault

Portlets obtain credentials by obtaining a `CredentialVaultPortletService` object and calling its `getCredential` method. With the returned credential, there are two options:

1. Use passwords or keys from a passive credential, passing them in application-specific calls. Portlets that use passive credentials need to extract the secret out of the credential and do all the authentication communication with the backend application.
2. Call the `authenticate` method of an active credential. Active credential objects hide the credential's secret from the portlet, with no way to extract it out of the credential. Active credentials provide additional methods to perform the authentication.

The latter case allows portlets to trigger authentication to remote servers using basic authorization, SSL client authentication, digest authentication, or LTPA without knowing the credential values. Using active credentials means that the portal authenticates on behalf of the portlet, and the portlet can simply use the open connection. While this may not be

possible for all cases, it is the preferred technique. For secure transmission of data, portlets can request a secure session (HTTPS) for accessing Web applications.

Microsoft SharePoint security

WebSphere Portal provides a number of mechanisms that enable it to use most popular third party authentication and authorization schemes. This includes support for using Active Directory as an LDAP, and an add-on product, IBM Directory Integrator, which enables it to synchronize a number of sources of directory information into a single directory. These features are the keys to integrating WebSphere Portal into a Microsoft environment.

A u t h e n t i c a t i o n

Microsoft Internet Information Services (IIS) handles authentication for SharePoint. To be authenticated, you need a local user account or a domain user account (if in a networked domain). In most cases, a domain account is a better choice than a local account because credentials for domain accounts can be maintained in one place for multiple systems.

You can configure authentication to operate in either pre-existing account mode or account creation mode. In pre-existing account mode, SharePoint does not automatically create new user accounts. In account creation mode, SharePoint can automatically create new user accounts in Active Directory. Account creation mode is a feature that you must select when you install Windows SharePoint Services or SharePoint Portal Server 2003.

A u t h o r i z a t i o n

The authorization mechanism is new in SharePoint 2003. It is proprietary code modeled on the NT file system security model, and uses SQL Server to store the Access Control entries for each principal (security rule).

After IIS uses a local computer account or an Active Directory account to authenticate a user, SharePoint compares the rights assigned to the user by IIS with the access control information for the SharePoint site to determine which SharePoint site resources the user is permitted to use. The authorization mechanism is new in SharePoint 2003. It is proprietary code modeled after the NT filesystem security model and uses SQL Server to store the Access Control entries for each principal.

Active Directory is not required for Windows SharePoint Services or SharePoint Portal Server 2003. However, without Active Directory, SharePoint Portal Server cannot pre-populate and synchronize the SharePoint Portal Server profile database with the list of users from Active Directory, and users' personal sites are not registered for cross-farm synchronization in a multi-server configuration.

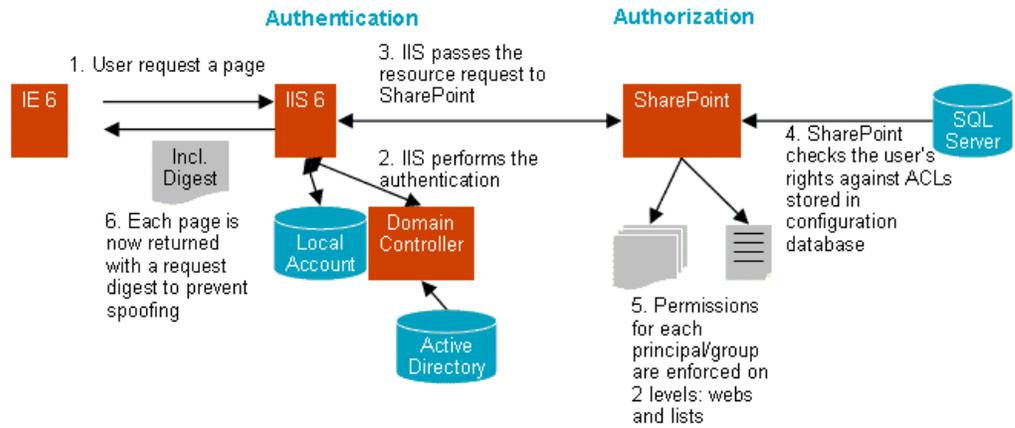


Figure 5: Authentication and authorization in SharePoint Portal

Authentication options and recommendation

There are two options for authentication with WebSphere Portal in an environment with Active Directory. The first option is for WebSphere Portal to use Active Directory directly as its LDAP. The second option is to synchronize a single reference directory from multiple sources.

Option 1: Shared directory

The most straightforward approach to ensure that users can use the same userids and passwords in the überportal and in each federated portal is to point all of them to the same shared directory.

In the case of an environment with SharePoint, that directory must be Microsoft Active Directory. In this case, we need to configure WebSphere Portal to utilize Active Directory as its LDAP repository. The steps to perform the configuration are outlined below:

Active Directory requires SSL connections in order for a 3rd party application to add, delete, and modify entries, so the first task is to install a certificate.

1. Get a certificate from your organization's certificate authority and copy it to your Active Directory server.
2. Double-click on the .cert file on your Active Directory server and follow the steps in the Wizard to import the certificate.
3. Use the Windows Address Book's Find command to test connectivity to the Active Directory LDAP using port 636.
4. Now, install WebSphere Portal and configure LDAP security using the standard deployment instructions.
5. Next, you need to edit some configuration files in WebSphere Portal.

Open this file in an editor:

```
<drive>:\websphere\portalserver\wmmLDAPServerAttributes.xml
```

Uncomment the following fragment:

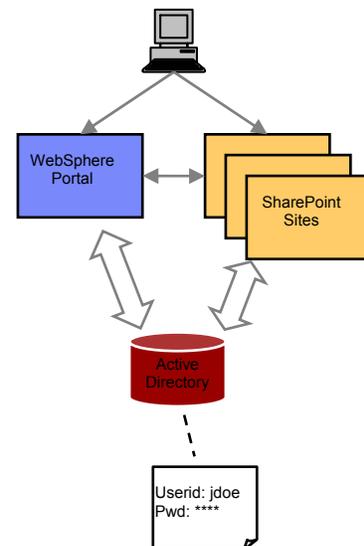


Figure 6: SharePoint Portal and WebSphere Portal referencing a single Active Directory instance

```
<attributeMap wwwAttributeName="preferredLanguage"
pluginAttributeName="preferredLanguage"
applicableMemberTypes="Person" dataType="String"
valueLength="128" multivalued="false">
```

Now, open this file in an editor:

```
<drive>:\websphere\portalserver\shared\app\wmm.xml
```

Add or cChange the entries in bold:

```
<ldapRepository name="wmmLDAP" UUID="LDAP1"
adapterClassName="com.ibm.ws.wmm.ldap.activedir.ActiveDirectoryI
AdapterImpl" supportDynamicAttributes="false"
configurationFile="C:/WEBSPH~1/PORTAL~1/wmm/wmmLDAPServerAttribu
tes.xml" wmmGenerateExtId="false"
supportGetPersonByAccountName="true"
profileRepositoryForGroups="LDAP1" supportTransactions="false"
adminId="cn=wpsadmin" adminPassword="Oi7GnZtJu6iiBK3ClqHJ5g=="
ldapHost="host.domain.com" ldapPort="636" ldapTimeOut="6000"
ldapAuthentication="SIMPLE" ldapType="0"
java.naming.security.protocol="ssl">
```

Remove any reference to `ibm-appUUID` from the rest of the file .

Save the files, and exit.

6. Copy the certificate file to your WebSphere Portal server.
7. Run: `<drive>:\WebSphere\AppServer\bin\ikeyman.bat`
8. Open the `cacerts` file and add the Active Directory certificate so that WebSphere Portal recognizes it.
9. Stop, and then restart all servers.

Option 2: Synchronized directories

Alternatively, you could can replicate Active Directory entries to IBM Directory Server (IDS) and point WebSphere Portal to IDS.

WebSphere Application Server (on which WebSphere Portal runs) does not have native support for multiple directories; therefore, a single consolidated directory is provided for WebSphere Portal to retrieve user information from. IBM Directory Server is kept in sync with one or more Active Directory servers that contain the source user account information, by IBM Directory Integrator. Because IBM Directory Integrator supports multiple sources, IDS can be populated from a combination of directory sources, such as intranet Active Directory and extranet Active Directory.

IBM Directory Integrator (IDI) server runs rules in the form of *AssemblyLines* for filtering, data transformation, and communications. *AssemblyLines* can run as batch processes or as the result of events that occur in the infrastructure. These events are detected and dispatched to relevant *AssemblyLines* by IDI EventHandlers.

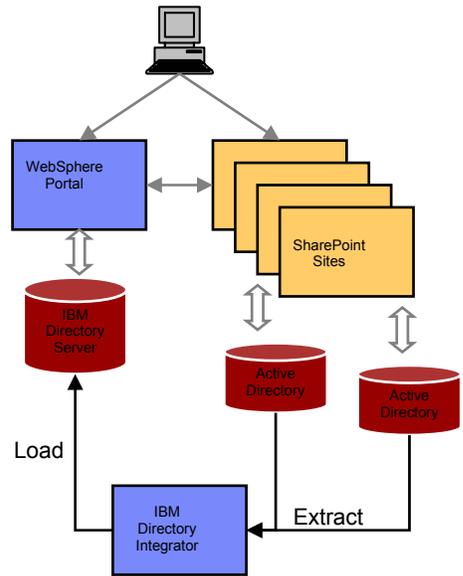


Figure 7: Consolidating multiple Active Directory instances with IBM Directory Integrator

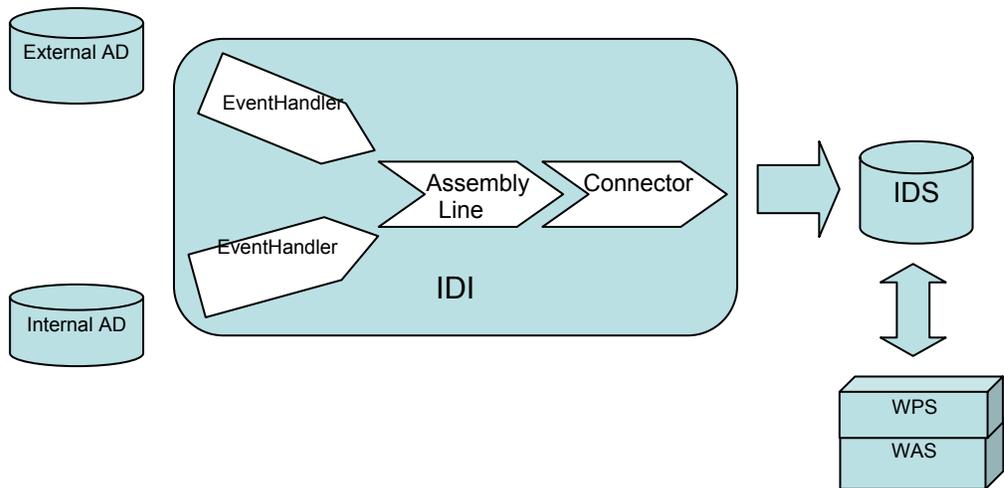


Figure 8: The IBM Directory Integrator Assembly Line

In this WebSphere Portal environment, you configure IBM Directory Integrator to run *AssemblyLines* which keep IBM Directory Server synchronized with the internal Active

Directory and the external Active Directory. The AssemblyLines are provided in an XML file, `SynchIDS.xml`, which does not have to be modified, and a properties file, `SynchIDS.props`, which is modified during the configuration process. The AssemblyLines are triggered when there is a change in either of the Active Directories, which allows users to log in to WebSphere Portal using their original userid and password.

R e c o m m e n d a t i o n

If WebSphere Portal is to be used as an intranet portal and the enterprise shares a single Active Directory repository, option 1, in which WebSphere Portal uses Active Directory as its LDAP source, is a good choice. However, in most cases, option 2 is either a better choice or is required. Most enterprises will not allow applications to write to their LDAP repository and WebSphere Portal requires this right for certain operations such as user registration.

Moreover, if WebSphere Portal is deployed in a network zone that is outside the intranet zone, port settings may not allow direct access to the corporate LDAP server. In this case, directory synchronization is the best choice. The down sides to the synchronization approach are added complexity and latency between the time a change is made in the source to the time it shows up in IDS.

Authorization options and recommendation

The goal is to allow users to execute portlets within WebSphere Portal that access SharePoint data without being challenged repeatedly for their credentials. The logical method to do this is to store the credentials in the WebSphere Portal credential vault and to have the portlet retrieve them when it needs to call a SharePoint service.

O p t i o n s f o r s t o r i n g c r e d e n t i a l s i n t h e v a u l t

WebSphere Portal differentiates between passive and active credential objects.

- Passive credential objects are containers for the credential's secret. Portlets that use passive credentials need to extract the secret out of the credential and do all the authentication communication with the back-end resource.
- Active credential objects hide the credential's secret from the portlet; there is no way of extracting it out of the credential. In return, active credential objects offer business methods to take care of all the authentication.

When you use active credentials, portlets never have direct access to the credential secrets. Therefore, there is no risk a portlet could violate any security rules such as storing the secret in the portlet session.

Recommendation

Although using active credentials is more secure, you cannot be certain that, in the environment in which the portlet will be installed, the SharePoint server is configured to accept basic authentication. If it is only configured to use the Windows proprietary "NT Lan Manager" authentication scheme, NTLM, our [sample portlet code](#) would fail. Because of this, we recommend using the `UserPasswordPassive` credentials in either a private or shared slot. Because our code supports both basic and NTLM, and we don't know ahead of time which will be used, we still recommend passive.

1. Generate the portlet using the portlet wizard in WebSphere Studio.
2. On the Single sign-on dialog, check **Add Credential Vault Handling**.
3. Enter a slot name.
4. Click **Finish**. The Wizard generates a source file called `CredVaultBasicAuthSecurityManager.java`.

The following methods are provided by this class for credential vault handling:

- `init` - initializes the `vaultService` data member
- `getCredential` - returns the user name and password by using a string buffer
- `setCredential` - sets the user name and password
- `getSlotId` - returns the ID of the slot. Depending on the type of slot, this method uses `PortletData` or `VaultService` to get the ID. New slots are created in the `createNewSlot` method.
- `getPrincipalFromSubject` - retrieves the specified Principal from the provided subject.
- `isWritable` - checks whether the password can be saved.

The wizard also creates a form called `CredVaultBasicAuthPortletEdit.jsp`. You can use this form to enable a user to enter his or her SharePoint userid and password the first time he or she accesses WebSphere Portal. When a user clicks on **Save**, the userid and password are stored into the vault.

5. In your portlet code, add the following lines of code to access the credential vault and to log in to SharePoint:

```
public void doView(PortletRequest request, PortletResponse response)
    throws PortletException, IOException {
    // Check if portlet session exists
    CredVaultPortletPortletSessionBean sessionBean = getSessionBean(request);
    if( sessionBean==null ) {
        response.getWriter().println("<b>NO PORTLET SESSION YET</b>");
        return;
    }
}
```

```

}

// Retrieve user credentials
StringBuffer userId = new StringBuffer("");
StringBuffer password = new StringBuffer("");
try {
    CredVaultPortletPortletSecretManager.getCredential(request,
        sessionBean,userId, password);

    // Use the credentials to log into SharePoint to call services
    java.net.URL url = new java.net.URL("http://host/_vti_bin/lists.asmx");
    ListsSoapStub port = (ListsSoapStub) locator.getListsSoap(url);
    port.setUsername(userId.toString());
    port.setPassword(password.toString());
    //.... Perform your operations here
}
catch( Exception e ) {
    if( getPortletLog().isWarnEnabled() )
        getPortletLog().warn("Warning on " +
            "CredVaultPortletPortletSecretManager" +
            ".getCredential(): "
            + e.getMessage());
}

// Invoke the JSP to render
getPortletConfig().getContext().include(VIEW_JSP+getJspExtension(request),
    request, response);
}

```

Portlets and Web Parts

Portlets and their corresponding Microsoft technology, Web Parts, are conceptually similar; however, in terms of implementation and technology, they have entirely different foundations. Although it would be ideal to be able to use a Microsoft Web Part within WebSphere Portal, technical limitations make this scenario impossible.

WebSphere Portal portlets

A portlet is a server side application that runs in the context of the WebSphere Portal server. It inherits from the `javax.servlet.http.HttpServlet` class and is treated as a servlet by the application server. The portlet executes inside a Web container managed by the application server. In the IBM Portlet API, this container is referred to as the portlet container.

WebSphere Portal is derived from the open source Apache `jetspeed` project. Portlets are Java-specific elements that can only run inside an application server that supports the Portlet API. IBM provides the IBM Portal Toolkit which provides the necessary wizards, templates, and test harnesses to create and test portlets within WebSphere Studio.

A simple “Hello World” portlet looks like the following:

```
package helloworld;
import java.io.IOException;
import org.apache.jetspeed.portlet.*;

public class HelloWorldPortlet extends PortletAdapter {
    public void doView(PortletRequest request, PortletResponse
response)
        throws PortletException, IOException {
        java.io.PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```

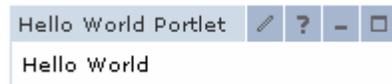


Figure 9: “Hello World” Portlet

SharePoint Web Parts

A Web Part is Microsoft’s term for the web component which corresponds to a portlet. Web Parts are contained within Web Part pages. Web Parts are standard ASP.NET custom

controls that are deployed in an assembly file. The assembly is usually deployed into the Global assembly cache.

To develop a custom Web Part, you use a development tool such as Microsoft Visual Studio .NET, and write them in either C# or VB.NET. All custom Web Parts are derived from the `WebPart` base class. This class provides basic properties that are common to all Web Parts, such as Title, Description, and so on. Microsoft provides a Web Part template that you can install to work with Visual Studio.NET. Because Web Parts are .NET elements, they require Microsoft IIS and the .NET framework environment to run.

Sample Code

A simple "Hello World" Web Part is as follows:

```
using System;
using System.ComponentModel;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml.Serialization;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Utilities;
using Microsoft.SharePoint.WebPartPages;
using Microsoft.SharePoint.WebControls;

namespace LCAWebPartLibrary
{
    [DefaultProperty("Text"),
     ToolboxData("<{0}:WelcomeUserWebPart runat=server/>"),
     XmlRoot(Namespace="LCAWebPartLibrary")]
    public class WelcomeUserWebPart :
Microsoft.SharePoint.WebPartPages.WebPart
    {
        private const string defaultText = "";
        private string text = defaultText;
        [Browsable(true),
         Category("Miscellaneous"),
         DefaultValue(defaultText),
         WebPartStorage(Storage.Personal),
         FriendlyName("Text"),
         Description("Text Property")]
        public string Text
        {
            get{return text;}
            set{text = value;}
        }

        protected override void RenderWebPart(HtmlTextWriter
output)
        {
            output.Write("Hello World");
        }
    }
}
```



Figure 10: A "Hello World" Web Part

This sample Web Part defines the configuration parameters in line. The creation of the Web Part assembly and the SharePoint Web Part framework handle the creation of the interface required to enable the end user to configure the part.

On the other hand, the WebSphere Portal framework requires the programmer to provide the configuration interface. Each approach has advantages and disadvantages. The WebSphere Portal approach requires more of the programmer and reduces the likelihood of consistency among configuration interfaces; however, more sophisticated configuration interfaces are more straightforward. SharePoint Portal Server's approach makes more sophisticated configuration more difficult, although not impossible.

Portlet standards

Several industry standards have been defined for portlets, including WSRP and JSR 168.

Web Services for Remote Portlets (WSRP)

Officially ratified as an OASIS (Organization for the Advancement of Structured Information Standards) standard in September 2003, WSRP defines how to plug remote Web services into the pages of online portals and other user-facing applications. The OASIS WSRP standard simplifies integration of remote applications and content into portals. With WSRP, portal administrators can select from a rich set of remote content and applications to integrate into their portals with just a few mouse clicks, with no programming. The WSRP specification defines a Web service interface for interactive presentation-oriented Web services. It enables users to perform the following tasks.

1. Producers can provide portlets as presentation-oriented WSRP services and make them available to Consumers who want to use these services.
2. Consumers can select from a rich choice of available Web services and integrate them into their portal.
3. Users can then access the services and work and interact with them just like they do with local portlets.

Benefits of WSRP

Using WSRP to perform these tasks has the following benefits:

- WSRP becomes the means for content and application providers to provide their services in an easily consumable form to organizations that run portals.

- All Web services that implement WSRP plug in to all WSRP-compliant portals without requiring any service-specific adapters. A single, service-independent adapter on the portal side is sufficient to integrate any WSRP services.
- Integrating content and applications into portals is made easier. No custom programming effort using a variety of different interfaces and protocols is required. Portal administrators no longer have to write interface code to adapt the WSRP services for their portal.
- Presentation-oriented services, as standardized by WSRP, allow Producer portals to deliver the requested data and their presentation to the Consumer portal. Previously, they delivered the data only, and the portal administrators had to provide the logic for how to present the data.
- Portal administrators do not have to keep the WSRP services code locally on their storage devices.
- The WSRP services appear and operate to portal users exactly like local portlets.

IBM is one of the founding members of the WSRP standards body, and supports WSRP s as a producer and as a consumer in WebSphere Portal V5.0.2.1. Microsoft joined the WSRP body in Feb 2004 but has not yet provided WSRP support in their portal products. Therefore, Web Parts are not consumable as WSRP remote portlets. Microsoft has released sample code demonstrating how WSRP could be achieved manually and on a small scale under the title WSRP Web Part Toolkit.

How to enable WSRP

To use WSRP, you need to enable both standards [JSR 168](#) (described below) and WSRP in your portal. WSRP requires JSR 168 as a prerequisite because the WSRP consumer client is implemented as a JSR 168 portlet.

To get the support for JSR 168 portlets in WebSphere Portal, you need your portal to run on IBM WebSphere Portal V5.0.2 Cumulative Fix 1 (5.0.2.1).

By default, the support for both standards, WSRP and the Portlet API as defined by Java Standard Request (JSR) 168, is disabled in the IBM WebSphere Portal V5.0.2.1. Therefore, until you specifically enable the support for these two standards, your portal would not support WSRP.

To enable WSRP, add the following two property keys to the file `wp_root/shared/app/config/services/ConfigService.properties` and set them to `true`:

```
For JSR 168:    portal.enable.jsr168=true
For WSRP:      portal.enable.wsrp=true
```

Restart the portal for your changes to take effect. Then, on the producer side, you can provide portlets as WSRP services; and, on the consumer side, WSRP services can be integrated and used as portlets. As a result, portal XML configuration interface commands for WSRP standard tasks are accepted.

JSR 168 establishes a standard API for creating portlets, which provide the integration component between applications and portals to enable delivery of an application through a portal. Without this standard, each version of an application has needed its own portlet API, and each of the various portals required that these portlets be specifically tailored for implementation through that portal. JSR168 is built on the WSRP specification. JSR168 is supported in WebSphere Portal 5.x. It is a Java only specification and, therefore, does not apply to SharePoint.

Portlet and Web Part options

Because portlets and Web Parts are based on completely different technologies, it is simply not possible to host a Web Part within WebSphere Portal, short of using an IFRAME within a portlet. The WebSphere Portal server does not contain the runtime environment to execute the Web Part. Therefore, we offer three options to mimic the Web Part functionality within a portlet.

Option 1: SharePoint Web service portlet

You can create a portlet which invokes one or more SharePoint Web services to implement its functionality. For this discussion, we will call this a SharePoint Web service portlet.

SharePoint provides an extensive Web services API which you can use to perform a wide variety of functions. However, this approach requires custom programming, and ultimately someone must maintain it.

These Web services interfaces are well documented in the MSDN library (see [Resources](#)). Because the majority of SharePoint data items (News, Events, To-Dos, Links, Contacts, and even Document Libraries) are stored in lists, you can use the `Lists` and `Lists Data Retrieval` services to get this data and display it in a portlet.

You can use WebSphere Studio Application Developer provides wizards to portlets and Web service clients. We used the following products to build the portlets in the download available with this article:

- WebSphere Studio Application Developer V5.1.2
- IBM Portal Toolkit V5.0.2.2

To generate the skeletal code for your SharePoint Web services portlet.

1. In WebSphere Studio, select **File -> New -> Portlet Project**.
2. Enter a name for your portlet project, select **Basic Portlet**, check **Configure advanced properties**, and then click **Next**.
3. Select `DefaultEAR` and `J2EE 1.3/WebSphere Portal 5.0`, and then click **Next**.

4. Accept the portlet settings page defaults and click **Next**.
5. **De-select Add Action Listener and click Next.**
6. Select **Add Credential Vault handling**, type in a name for the vault slot, and then click **Next**.
7. Select **Add Help Mode** and click **Finish**. The wizard creates all the necessary files and displays the WebSphere Studio Portlet Perspective view.
8. Next, select **File -> New -> Other -> Web Services -> Web Service Client**.
9. Leave client proxy type as **Java proxy**, check **Create folders as necessary**, and then click **Next**.
10. Select **Web service runtime protocol**, and then select **Apache Axis 1.0** as the runtime and WebSphere Portal test environment as the server.
11. Leave client type as **Web**. Select the portlet project you created above as the Client Project and **DefaultEAR** as the Client Default EAR. Click **Next**.
12. Enter the URL for your SharePoint Web service. For example:
`http://server/_vti_bin/web.asmx?wsdl`
13. Click **Next**.
14. Enter your userid and password to SharePoint and click **OK**.
15. Click **Finish**.

You are now ready to start coding!

Sample Code

Figure 11 shows a sample portlet that was developed using the instructions detailed above. This example retrieves a list of events from a SharePoint team site and displays it in a portlet.

WebSphere Portal My Portal Administration

New Page Edit

My Finances My Newsroom SharePoint integration

Events Document Library

SharePoint Events

July 2004						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21 Tech Talk	22	23 Holiday	24
25	26	27	28	29	30	31

Detailed Event Listing

Date/Time	Event
2004-03-23 09:00:00	SWAT Bootcamp
2004-07-21 11:00:00	Tech Talk
2004-07-23 00:00:00	Holiday

Figure 11: Event portlet in View mode

This portlet lets the user store their SharePoint credentials in the credential vault; it also stores the site URL. The majority of code is found in the `SharePointList` `getEvents()` method. The portlet uses classes generated by WSDL2Java, and obtains a handle to the `Lists` Web service port. You also need to build the query XML document using the Microsoft (proprietary) CAML format. The query XML is passed to the `Lists` Web service when the portlet calls it, and the Web service returns the results in CAML, in the following form. The XML elements in bold font represent the rows resulting from the query.

```
<GetListItemsResponse
  xmlns="http://schemas.microsoft.com/sharepoint/soap/">
<GetListItemsResult>
  <listitems xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
    xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
    xmlns:rs='urn:schemas-microsoft-com:rowset'
    xmlns:z='#RowsetSchema'>
    <rs:data ItemCount="3">
      <z:row ows_EventDate='2004-05-27 00:00:00'
        ows_Title='YourCo Quarterly Results'
        ows_fRecurrence='0' ows_ID='8'
        ows_owshiddenversion='1' />
      <z:row ows_EventDate='2004-07-05 12:00:00'
        ows_Title='YourCo Proposal Decision'
        ows_fRecurrence='0' ows_ID='9'
        ows_owshiddenversion='1' />
      <z:row ows_EventDate='2004-06-04 00:00:00'
        ows_Description='Sample' ows_Title='New appointment'
```

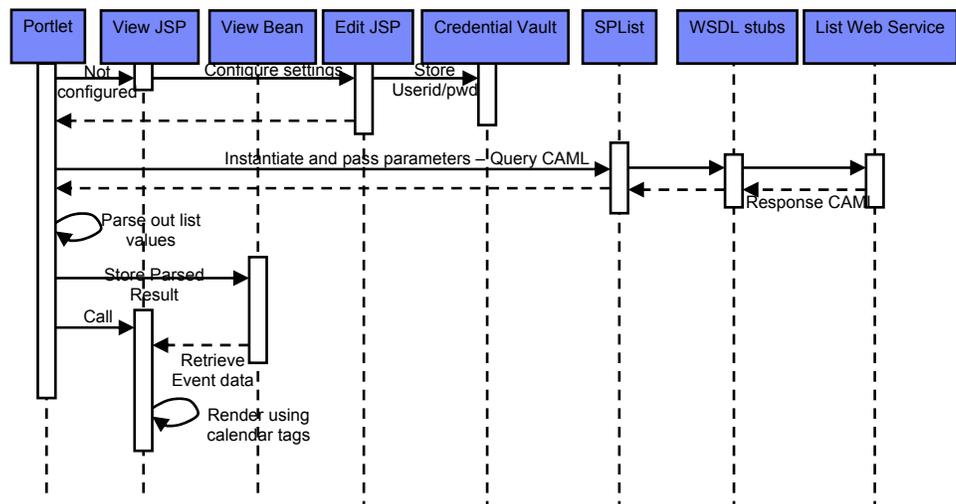
```

        ows_fRecurrence='0' ows_ID='10'
        ows_owshiddenversion='1' />
    </rs:data>
</listitems>
</GetListItemsResult>
</GetListItemsResponse>

```

The results are parsed using XPath to retrieve the `<z:row>` data which contains the events and the dates. This data is stored in the `ViewBean`. The portlet then calls the `View.jsp` which renders the data in the `ViewBean` in a tabular format.

Figure 12 shows a high level interaction diagram between the various objects.



Note: For illustration only. Does not comply with UML

Figure 12: Interaction diagram of a portlet invoking a SharePoint Web service

See the source for `EventPortletPortlet.java`, `SharePointList.java`, `EventPortletPortletViewBean.java`, and `EventPortletPortletView.jsp` in the `Event` portlet (in the download) for an illustration.

This is a relatively simple example meant to illustrate the use of the `Lists` Web service. Although it contains some error handling code, a production portlet would need more robust error handling and scalability testing.

The following tips might be helpful to you if you want to re-produce the samples.

1. To access the lists data in a particular SharePoint Services site, use the full site URL. For example:
http://sharepointserver/sites/sitename/_vti_bin/lists.asmx?wsdl

2. Use Axis instead of the default Web service engine (WebSphere 5) so that you can manipulate the results using DOM instead of iterating through a list of SOAPElements (this option is selected in the Web Service client wizard in WebSphere Studio).

3. If you decide to use XPath to extract the values from the CAML response document, you need to address the element using `/child` instead of `.../rs:data/z:row` because XPath does not like the pound symbol in the namespace, `z=#RowsetSchema`. You would address the row elements using the following:

```
/ns1:listitems/rs:data/child::node()[self::*]
```

You use `self` to test that you have retrieved the correct elements.

4. Add a `commons-logging.properties` file to your `commons-logging.jar` to force WebSphere to use the proper log factory instead of `TrLogFactory`. The contents of the properties file are:

```
org.apache.commons.logging.LogFactory=org.apache.commons.logging
.impl.LogFactoryImpl
org.apache.commons.logging.Log=org.apache.commons.logging.impl.S
impleLog
```

5. Modify

`org.apache.axis.components.logger.LogFactory.getLogFactory()` so that it calls `org.apache.commons.logging.LogFactory.getFactory()` instead of issuing a `find()`. Recompile `Axis.jar` with your new class. The download includes a version of Axis which has been modified to perform this.

6. Remove the `javax.xml.transform` package from `jaxrpc.jar` in order to avoid classloader conflicts.
7. Add the `PortalStruts.jar` file to your `/WEB-INF/lib` folder to run an older version of WebSphere Portal (pre 5.0.2), Lotus Workplace 1.1.1, and WebSphere Portal Express. Portlets found in these older versions contain a version of the Struts toolkit that sets a system property. This system property forces logging calls to use `StrutsLogFactory` even if you have the `commons-logging.properties` file set to the log factory of your choice and have set the classloader to `PARENT_LAST`.

8. If you wish to use JSP taglibs in your portlet JSPs, specify the following tag in your `taglib.tld` file:

```
<rtexprvalue>>true</rtexprvalue>
```

9. Accessing static resources e.g. images requires one to parse out the front part of the path. For example, `http://server:port/application_1.0.0/WEB-INF/..`. Your code needs to determine the `server:port` portion of the path. For example (`baseUrl` is the variable name):

```
int i = 0, c = 0;
```

```

StringBuffer sb = request.getRequestURL();
for(;c < 3; i++)
{
    if (sb.charAt(i) == '/')
        c++;
}
String baseurl = sb.substring(0, i - 1);

```

Option 2: WSRP Portlet with Netunity

Netunity is a third party vendor that supplies a toolkit which helps you to build .NET applications that are WSRP compliant. Netunity states that it supports both WSRP production and consumption.

The WSRP Portlet Framework for .Net is a comprehensive portlet development solution that includes multiple sets of .Net classes that addresses every aspect of portlet application construction. The .Net classes are accessed through the Visual Studio IDE, and driven with the use of Add-ins and Wizards. Some of its features include:

- Visual Studio Add-ins and Wizards
- .Net classes for all supporting layers
- database-independent
- data access layer
- repository driven
- XML/XSL forms layer
- .Net class library implements v1.0 of WSRP specification
- WS-Security
- UDDI producer and portlet registration

See Resources for a reference to the Netunity Web site, where you can find a more detailed description. We were unable to test this alternative; however, it sounds promising, and could let you create .NET classes to wrap Web Parts and expose them as WSRP portlets.

Option 3: JDBC portlet

Because SharePoint stores its data within SQL Server it is possible to write a portlet that uses JDBC to access the data directly. In order to do so, download the JDBC driver for SQL Server 2000 from the Microsoft Web site, and install it. (See Resources for a reference to the Web site.)

A list of SharePoint's tables and their schema is found in the SharePoint Products and Technologies 2003 SDK on MSDN under **Platform > Microsoft Windows SharePoint Services > Reference > Database > Tables**.

This approach should only be taken if:

1. You need to access some piece of data that is not available via the web services interface.
2. You need the performance of direct data access.

JDBC code is very mature technology and we have only provided a very skeletal snippet here:

```
import java.sql.*;
.....
public void doView (PortletRequest request, PortletResponse
response) {
    // Get connection
    DriverManager.registerDriver(new
        com.microsoft.jdbc.sqlserver.SQLServerDriver());
    Connection connection = DriverManager.getConnection(
        "jdbc:microsoft:sqlserver://<Host>:1433", "<UID>", "<PWD>");
    if (connection != null) {
        response.getWriter().println();
        response.getWriter().println("Successfully connected");
        response.getWriter().println();
        //perform your db operations
    }
}
```

Recommendation

For the majority of cases, we recommend using web services portlet to access SharePoint. Using the service provides a level of abstraction, and preserves the integrity of the underlying data structure. The only drawback would be the relative speed of a text based protocol such as SOAP (along with the requisite marshalling/unmarshalling of parameters) as opposed to a binary protocol such as JDBC. However, in most cases portlets typically display a small window of information and do not require the transfer of large amounts of data rendering the performance argument moot.

Search

The user should not need to perform multiple, separate searches in order to query the integrated portal. Ideally, results from all portals should be returned when the user issues a search query within a portlet. This section describes one way to implement this, and tells how to write a custom portlet that queries SharePoint content.

WebSphere Portal search overview

WebSphere Portal provides integrated Web content search capabilities, including a search portlet, a crawler, a document indexer, and content categorization options. Portal search can index plain text documents as well as over 200 other file formats using built-in document filters.

In order to search SharePoint data you need to use Lotus® Extended Search (hereafter called Extended Search), which is included with the WebSphere Portal Extend for Multiplatforms edition. Extended Search provides distributed, heterogeneous searching across a variety of repositories, without the user having to know the details of these various systems. The result is single-point of access to a variety of data sources without requiring a new, central index. Extended Search also searches external sources such as Microsoft Index Server and Site Server, LDAP-compliant directories, eighteen popular Web search sites and News sites, commercial content providers, and relational databases such as IBM DB2, Oracle, Sybase, MS SQL-Server, and other ODBC compliant databases. Results can be ranked by relevancy over multiple data stores.

The Extended Search search engine is configured to include the SharePoint data source. A user who executes a search from the Extended Search portlet in WebSphere Portal receives results which include documents and data in the SharePoint sites. Portlets are provided for WebSphere Portal to access Extended Search. These portlets provide several search options:

- **Simple Search** presents a query box in which users can enter search terms, which are either simple words or an expression that uses Generalized Query Language (GQL), web, or native query language syntax. The portlet manager determines which search application and which Extended Search server will be used to process the query, which sources will be searched, and which search options will be applied. Users with edit privileges can override some of these default settings and choose different search options, which are applied to all searches processed for that user's login ID. Search results are displayed with links to the original documents or sites.
- **Enhanced Search** presents a query box in which users can enter search terms, either simple words or an expression that uses GQL, web, or native query language syntax. Users can also see which sources are being searched and which search options are being applied. The portlet manager determines which search

application and Extended Search server will be used to process the query, which sources will be searched, and which search options will be applied. Users with edit privileges can override some of these default settings, and choose different sources and different search options, which will be applied to all searches processed for that user's login ID. Search results are displayed with links to the original documents or sites.

- **Advanced Extended Search** is a sample portlet that shows application developers many of the more advanced features of Extended Search, such as submitting multiple queries at the same time, using constraints to fine-tune a set of search results, setting a message severity level, saving and scheduling queries, editing saved queries, and saving search result documents to a file system. This portlet is provided for demonstration purposes only and is not supported.

If you wish to customize the search results, you can use the provided JavaBeans and JSP tags to displays the result set.

SharePoint search overview

SharePoint Services keeps all content in SQL Server content stores, and uses the Microsoft Search full-text indexing and searching technology from Microsoft SQL Server. SharePoint Services can index and search content in SQL Server content stores only.

SharePoint Portal Server 2003 uses the latest version of Microsoft Search technology to index both local and external document collections and Web sites. It supports all of the advanced indexing and searching features from SharePoint Portal Server 2001, with improved performance, scalability, and extensibility. SharePoint Portal Server 2003 can index and search approximately 20 million documents (a five-fold improvement) and can support load-balanced queries across multiple catalog servers.

Options and Recommendation

Option 1: Lotus Extended Search

Deploying Extended Search is the infrastructure approach option. It does not require any programming, but it requires that you set up and configure Extended Search.

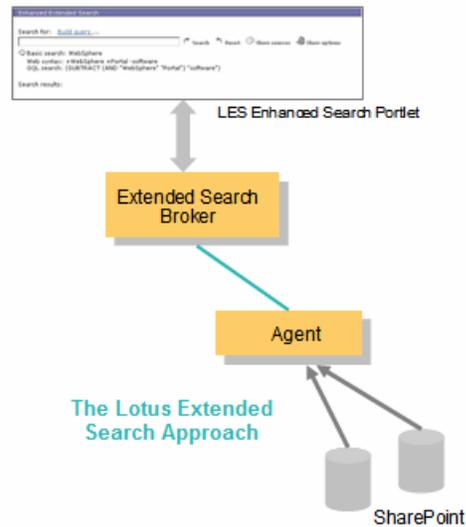


Figure 13: Lotus Extended Search

In order to setup Extended Search to use SharePoint as a data source, you first run the data source discoverer wizard:

1. To start the administration interface, in Windows, select **Start -> Programs -> IBM Extended Search -> Extended Search Administration**.
2. Click **Servers** in the navigator. Right-click on the server you want to work with and select **Discover data source**.
3. Select SharePoint Server as the data source type and specify the following information:

HOST NAME

Type the fully qualified host name for the SharePoint Portal Server. If the server is not running on the default HTTP port (80), include the port number with the host name. For example: `www.myserver.com` or `www.myserver.com:81`.

USER NAME

The user name used to authenticate with the SharePoint portal server.

PASSWORD

The password used to authenticate with the SharePoint portal server.

LANGUAGE USED

Select the locale of the target data sources.

CODE PAGE USED

Select the code page to be used when communicating with the target data sources.

4. Now click **Start Discovery**. The system displays a list of the sources that it found.

5. Select one or more data sources from the list and click **Add to Extended Search**. The selected data sources are added to your search domain.
6. Next, you configure the field definitions. You can add fields, delete fields, and select fields that you want users to search or retrieve, and so on.
7. Categorize the data source. The discoverer automatically associates each data source with a category that has the same name as the link that is used to connect to the source. If you prefer, you can put the source into different categories, including a new category that you create.
8. Associate the category with an application. You can use the Demo application to search newly added sources. If you want to use a different search application, you must associate the data source with a category that belongs to that application. You can also define application-specific properties to govern how users of the application can search the data source.
9. Finally, refresh the search domain to disseminate your changes. Now when users issue a search from the Extended Search portlet, the results will include SharePoint content.



Figure 14: Configuring Lotus Extended Search

Option 2: SharePoint Query portlet

Another option is to develop a portlet which calls the Query web service. The Query service accepts a request in the Microsoft.Search.Query XML format and returns a response in the Microsoft.Search.Response XML format. It supports queries as either a simple list of search terms, or in the Microsoft SQL Syntax for Full-text Search.



Figure 15: Using the SharePoint Portal search

The portlet calls the Query service through the following URL:
http://server_name/_vti_bin/search.asmx

We are currently in the process of developing a search portlet. Unfortunately there are a number of incompatibilities between Axis/WSDL2Java and Microsoft's WSDL description document. Furthermore, the SOAP-Action header seems to be causing a failure during the call to the Web service.

Recommendation

If you need to search multiple sources and want to include SharePoint resources in the search path, then deploying Lotus Extended Search is recommended. In most cases, a black box solution is easier to maintain than a bespoke programming approach. The SharePoint query portlet will only return results from the SharePoint to which you pointed. Although the portlet gives you greater flexibility with constructing the query, the price you pay is in code maintenance. In addition, this Web service is only available when SharePoint Portal Server 2003 is installed. It is not available with SharePoint Services.

Personalization

WebSphere Portal and SharePoint Portal Server take very different approaches to personalization and customization. Reconciling the two is largely a matter of determining migration; that is, determining what can be converted and what can simply be redone.

WebSphere Portal personalization and customization

WebSphere Portal provides end-user and administrative interfaces for customizing the content of portal pages, as well as the look and layout of the pages. With these tools, users can customize their own pages by selecting portlets and customizing the settings of each one. Users can also change the page layout and the color scheme (if the administrator allows it).

C u s t o m i z a t i o n

Users can have one or more personalized pages, and can navigate to each one from the home page. Pages are arranged in a hierarchical manner with any level of depth. Each page can have its own choice of color themes, skins, and page layouts. Themes are used to define the fonts, colors, spacing, and other visual elements; themes consist of cascading style sheets, JSP files, and images. Skins are decorations and controls placed around portlets, such as title bars, borders, and shadows. At each level of the page hierarchy, the lower pages can inherit the themes and skins from the page above them, or can override one or both. Because the look and feel of each section can be completely different, sections of the site can be used to create the appearance of different sites running on one portal server.

Each personalized page can have a different set of portlets. The portlets on a page can be selected by end users or by administrators, depending on their access rights for the page. Administrators can specify that certain portlets are required, so that end users cannot remove them or re-arrange them. Pages can also be re-arranged to provide a different navigation order for each user, or group of users, when permitted by the administrator. For end users, the portal provides a quick customizer interface for adding and re-arranging portlets. Users access the customizer by clicking on the **Edit Layout** link at the top of the page to customize; then, to add new portlets, they click the **Add portlets** button .

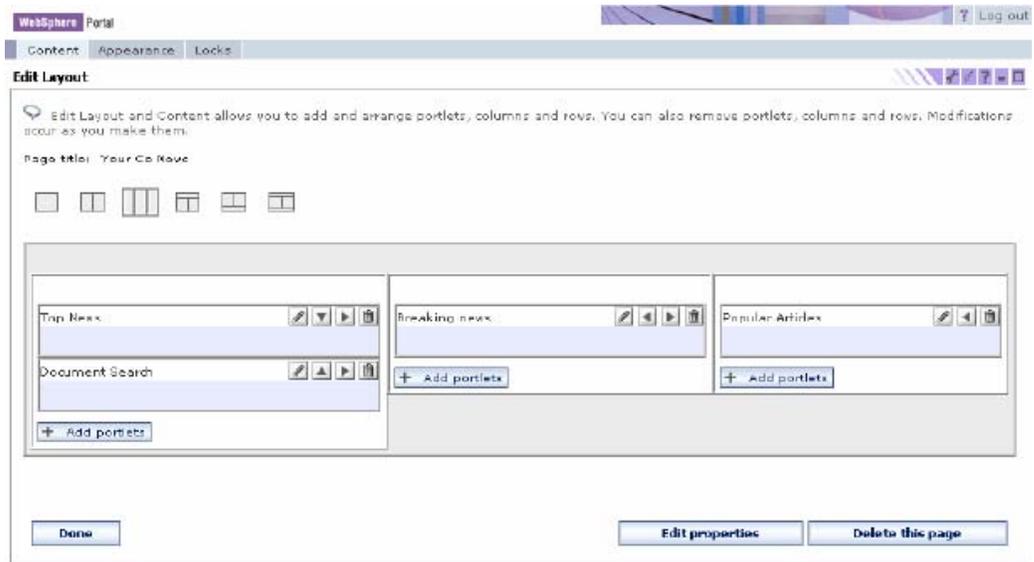


Figure 16: Configuring a WebSphere Portal page

Personalization

The WebSphere Portal offerings include the WebSphere Personalization server, which lets you target content to specific users in support of your portal's business goals. WebSphere Personalization provides facilities that allow subject matter experts to select content suited to the unique needs and interests of each site visitor.

A personalization solution involves three basic components:

1. User profile – specifies information about users of the site, including attributes about the users.
2. Content model -defines attributes about the content, such as product descriptions, articles, and other information.
3. Matching technology – various engines which match users to the right content. Includes filtering, rules, and recommendation engines or combinations of all three.

The WebSphere Personalization server and WebSphere Portal share a common user profile and a common content model. The model is based on the WebSphere resource framework interfaces classes. Therefore, you can easily add personalization rules to portlets to select portal content and target it to the portal's registered users. The basic steps involved in personalization involve classifying site visitors into segments, and then targeting relevant content to each segment. Business experts create the rules for classifying users and selecting content, using Web -based tools.

WebSphere Personalization also includes a recommendation engine, which provides collaborative filtering capabilities. Collaborative filtering uses statistical techniques to identify groups of users with similar interests or behaviors. Inferences can be made about what a particular user might be interested in, based on the interests of the other members of the group.

SharePoint personalization and customization

To help individual users find and use the information and tools they need, a portal solution must support targeted delivery of content, information, and application functionality. This includes targeting information and applications to individuals, teams, divisions, and entire organizations. This also includes effective support for personalized content and support for group-based portal page content.

Personalization

Personalization is a service included in Windows SharePoint Services. SharePoint Portal Server 2003 uses *Audiences* to extend this service. Audiences are dynamic groups of users that share one or more common properties (for example, business function, department, or team membership). The properties that determine Audience membership can reside in an enterprise directory such as Microsoft Active Directory or any other SQL Server-based database. Audiences are used to determine which Web Parts appear on a particular Web page, and they can also act as a filter for the information displayed in those Web Parts. SharePoint Portal Server 2003 also supports creating and managing individual portal pages for each user.

Customization

In SharePoint Team Services and SharePoint Portal Server 2001, you could use Subscriptions to receive messages when your shared documents were changed. The name of the Subscriptions feature has now changed to Alerts. Windows SharePoint Services and SharePoint Portal Server 2003 both support Alerts.

My Site is a personal SharePoint site that provides personalized and customized information. This site contains content targeted to users based on their membership in a particular audience. My Site also provides quick access to user's documents, people, or Web sites, as well as personal alerts they created to track changes to portal content. From My Site, users can update their user profiles and share links with other portal users.

Options

Option 1: My Site

The My Site settings for each user are stored in SQL Server. There are no APIs for accessing these settings, and we do not have adequate information on the data structure to successfully extract this information. For the most part, even if we were able to retrieve the settings, WebSphere Portal is different enough from SharePoint that it would be difficult at best to replicate the My Site environment.

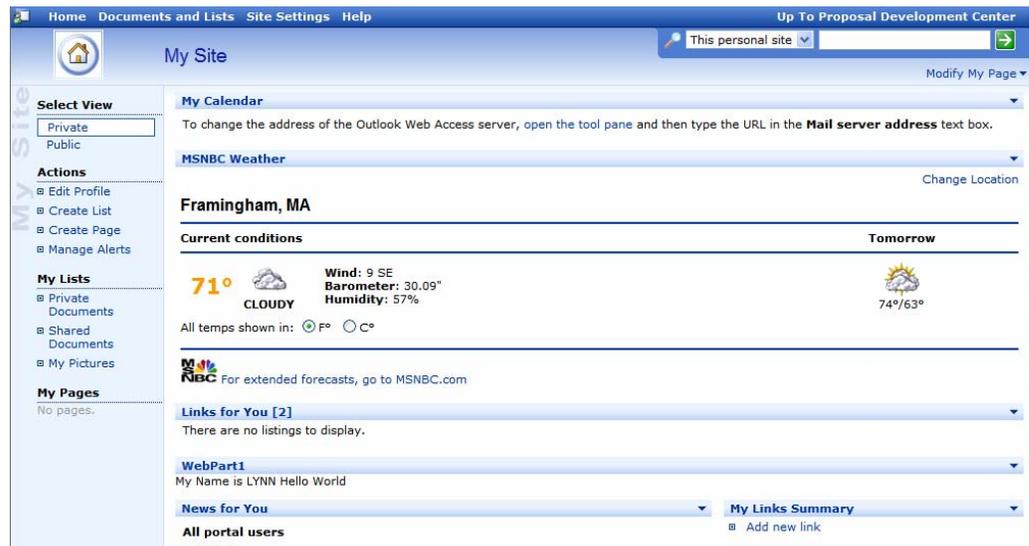


Figure 17: A SharePoint Portal "My Site"

Option 2: Alerts

Alerts are usually set on a document library list, although they are programmable to act on any list. Receiving alerts is simple because they are sent out to an e-mail address which is not likely to change when there is a change of portal infrastructure. Enabling the alert is more of a challenge. The Alerts Web service only provides `getAlerts()` and `deleteAlerts()` methods.

One approach would be to develop a custom .NET Web service to make use of the `Microsoft.SharePoint.Portal.Alerts.AlertCollection.CreateAlert()` method to create the alert, and then to call that Web service from a portlet. Alternatively, use the SharePoint Custom Event Sink toolkit to register a custom event which contacts a WebSphere Portal hosted Web service when an event triggers.

Home Documents and Lists Create Site Settings Help Up to Proposal Development Center

Tim's Demo site
New Alert: Proposal Documents: All items

Use this page to create an e-mail alert notifying you when there are changes to this item. [More information on alerts.](#)

<p>Send Alerts To</p> <p>All of your alerts will be sent to this e-mail address. Change my e-mail address.</p>	<p>My e-mail address is: bill_dipaola@us.ibm.com</p>
<p>Change Type</p> <p>Specify the type of changes that you want to be alerted to.</p>	<p>Alert me about:</p> <p><input checked="" type="radio"/> All changes</p> <p><input type="radio"/> Added items</p> <p><input type="radio"/> Changed items</p> <p><input type="radio"/> Deleted items</p> <p><input type="radio"/> Web discussion updates</p>
<p>Alert Frequency</p> <p>Specify whether you want to be alerted immediately when there is a change, or if you would rather receive a daily or weekly summary message.</p> <p>View my existing alerts on this site.</p>	<p>Alert me how often:</p> <p><input checked="" type="radio"/> Send e-mail immediately.</p> <p><input type="radio"/> Send a daily summary.</p> <p><input type="radio"/> Send a weekly summary.</p>

Figure 18: SharePoint Portal Alerts

Option 3: Audiences

There are no out of the box Web service endpoints that allow audience data retrieval and manipulation. You could develop a custom .NET Web service to wrap the `Microsoft.SharePoint.Portal.Audience.*` classes, which would get the members and populate a corresponding segment in WebSphere Personalization.

Proposal Development Center > Site Settings
Manage Audiences

Use this page to manage and compile audiences.

Audience Settings

Use these links to manage and compile audiences.

Number of audiences:	2
Uncompiled audiences:	1
Compilation status:	Idle
Compilation time:	Not compiled yet
Compilation schedule:	Specify schedule
Compilation errors:	No error

- [Refresh](#)
- [Create audience](#)
- [View audiences](#)
- [Specify compilation schedule](#)
- [Start compilation](#)

Figure 19: SharePoint Portal Audiences

Recommendations

This is one area where it is very difficult to achieve integration. SharePoint stores its personalization settings for each user in SQL Server tables. These settings are mainly the My Site settings such as page layout and Web Part placement. Event alerts in each team site are also examples of personalized settings. Audience content targeting is another form of personalization; however, it is set on a group rather than individual level. As with My Sites, all this configuration data is stored in SQL Server tables.

The best bet for My Site-type user customized pages is for the user to re-create the layout manually in WebSphere Portal. Since this is a one-time effort, the productivity loss is minimal.

Creating alerts and retrieving audience membership is most easily achieved by developing a custom .NET Web service that wraps calls to the SharePoint object model. This Web service would reside on the SharePoint server and respond to requests from portlets.

Document management

Users of the WebSphere Portal überportal should be able to access documents stored in SharePoint document libraries. In addition, they should be able to access the document management functions such as check in/checkout, versioning, and workflow.

WebSphere Portal document management

WebSphere Portal provides document management capability through a component called Portal Document Manager (PDM) which lets authorized users view, add, edit, and delete documents within a user-defined folder hierarchy, as shown below.

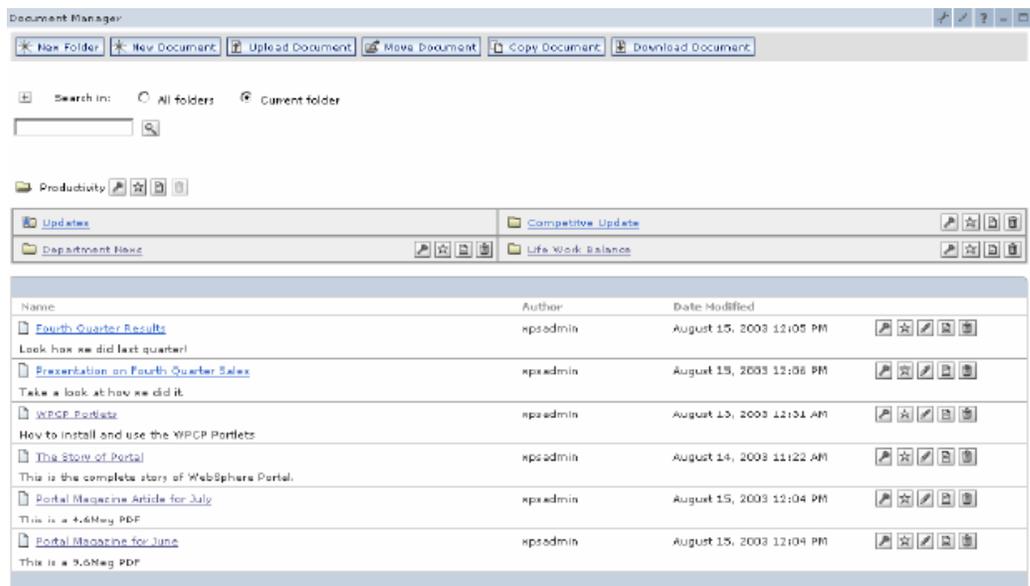


Figure 20: WebSphere Portal Document Manager

PDM supports versioning and workflow. It uses WebSphere Portal's access control capabilities to control which users can view or edit specific documents. You can set privileges on a project, folder, subfolder, or individual document. Documents are integrated with the search engine. PDM provides productivity editors for common formats such as rich text, spreadsheets, and presentations.

SharePoint document management

SharePoint supports two types of content stores. Based on Microsoft SQL Server technology, SQL Server, which is the primary store, provides a single, consistent data storage solution for document content, list content, and metadata. You can use common Windows and SQL Server management tools and development tools to easily manage, tune, back up, and enhance SQL Server content stores.

When you install SharePoint Portal Server 2003, you have the option of installing the backward-compatible document store, which is an updated version of the Web Storage System-based document store that was used in SharePoint Portal Server 2001. This document store is for users who require features from SharePoint Portal Server 2001, such as complex document routing and approval, folder-level security, minor-level version numbers, and multiple document profiles for each folder.

The primary interfaces for the document management features in SharePoint are document libraries, which you can add to any SharePoint site. A document library consists of the virtual folder where the files are stored, the files themselves, and the user-definable descriptive information (metadata) associated with each item in the document library.

Options

Option 1: Document batch loading

If your requirements call for sunsetting the SharePoint sites and migrating the documents over to Portal Document Manager, then the best approach would be to develop a migration application which reads the SQL Server documents as bytestreams and copies them to PDM in a batch operation. The application can either be custom coded or a third party ETL (Extract-Transform-Load) tool.

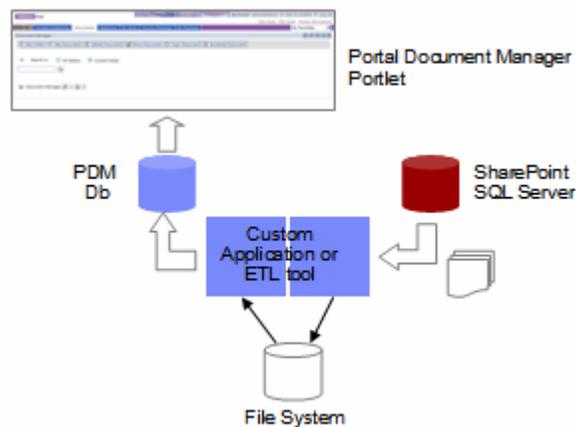


Figure 21: Migrating content from SharePoint to Portal Document Manager

The custom application would probably consist of two modules: a VB.NET or C# application which uses the SharePoint object model to extract the files, and a Java application which uses the PDM Java API to load the documents into WebSphere document management.

The SharePoint object model lets you access the documents in a document library. The listing below shows how:

```
SPWeb web = SPControl.GetContentWeb(Context);

//Get all files in a Document Library
SPDocumentLibrary doclib = (SPDocumentLibrary)
    web.Lists["Shared Documents"];
foreach (SPListItem item in doclib.Items)
{
    SPFile file = item.File;
    byte[] fileBuffer = file.OpenBinary();
    string path = "c:\\\" + file.Name;
    FileStream stream = new FileStream(path, FileMode.Create);
    stream.Write(fileBuffer, 0, fileBuffer.Length);
    stream.Close();
}
```

Option 2: SharePoint document portlet

If, on the other hand, the SharePoint sites will continue to be actively used, with documents being added, deleted, and modified, then the best approach would be to create a portlet that mimics the functionality of the SharePoint document library. You also need to add some user-selectable configuration parameters to customize the output of the display.

The *SharePoint Recent Documents* portlet, which is one of the [sample portlets](#), is an example of a portlet which could be used for this purpose. The rest of this section describes the SharePoint Recent Documents portlet.

You can use the SharePoint Lists Web service to get all the lists in a particular team site. The return result is parsed with XPath to extract only those lists that are document libraries (type = "101"). The libraries are presented in a drop down list. Once the user selects a list, the most recent documents in that library are displayed. The document names are clickable URLs. Clicking on a document name fires up the associated helper application, such as Microsoft Word. SharePoint streams the document contents to the MSOffice application using WebDAV. The user can then edit the document in-place and save it directly back to SharePoint.

SharePoint also supports subfolders in document lists. In the results from `GetListItems()`, files have an `FSObjType` attribute value of "0" and folders have a value of "1". The latest version of the our sample portlet supports subfolders.

The Lists Web service is called using the URL:

```
http://Server_Name/[sites/][Site_Name/]_vti_bin/Lists.asmx
```

It supports the following functions:

- AddAttachment
- AddList
- DeleteAttachment
- DeleteList
- GetAttachmentCollection
- GetList
- GetListAndView
- GetListCollection
- GetListItemChanges
- GetListItems
- UpdateList
- UpdateListItems

The Recent Documents portlet uses the `GetListCollection` and `GetListItems` methods. See the `SPDocs.java` in the SharePoint Recent Documents portlet for an illustration.

The results of the calls are transformed with Xalan XSLT into XML data islands. Internet Explorer renders the data islands that are embedded in the JSP. The results are shown below.

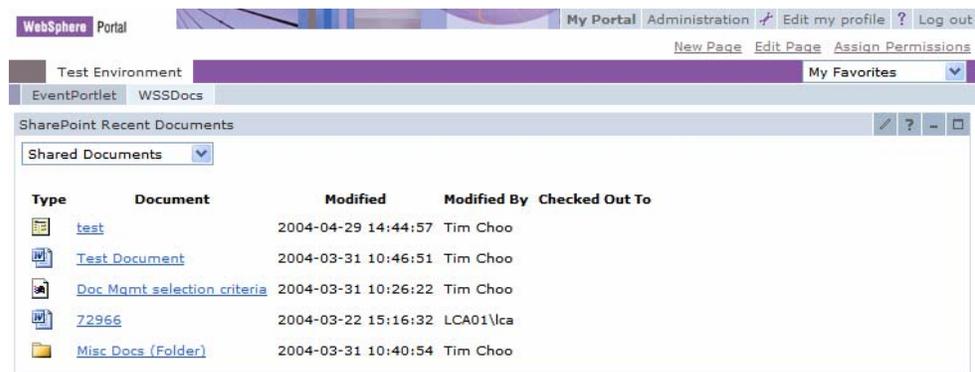


Figure 22: The SharePoint Recent Documents portlet

Option 3: Versioning information

If version information is required in the portlet, the SharePoint Versions Web service can be called. The Versions Web service is called using the URL:

`http://Server_Name/[sites/][Site_Name/]_vti_bin/Versions.asmx`

It supports a number a methods:

- DeleteAllVersions
- DeleteVersion
- GetVersions

- RestoreVersion

The GetVersions method takes a filename URL as input and returns the following information:

```
<results xmlns="http://schemas.microsoft.com/sharepoint/soap/">
  <list id="{26E516B0-8241-4B97-984D-1FA7C985080D}" />
  <versioning enabled="1" />
  <settings
    url="http://Server_Name/Site_Name/_layouts/1033/LstSetng.aspx?
    List={26E516B0-8241-4B97-984D-1FA7C985080D}" />

  <result version="4"
    url="http://Server_Name/Site_Name/Shared
    Documents/File_Name.doc"
    created="6/7/2003 5:55 PM" createdBy="DOMAIN\User"
    size="19968"
    comments="" />

  <result version="1"
    url="http://Server_Name/Site_Name/_vti_history/1/Shared
    Documents/File_Name.doc"
    created="6/7/2003 5:49 PM" createdBy="DOMAIN\User"
    size="19968" comments="" />
    .
    .
    .
  </results>
```

Option 4: Accessing Documents directly

If, for some reason, you need to directly access the documents in SharePoint then you would need to write low level database manipulation routines. You would need to understand SharePoint's database schema, which is illustrated in the figure below.

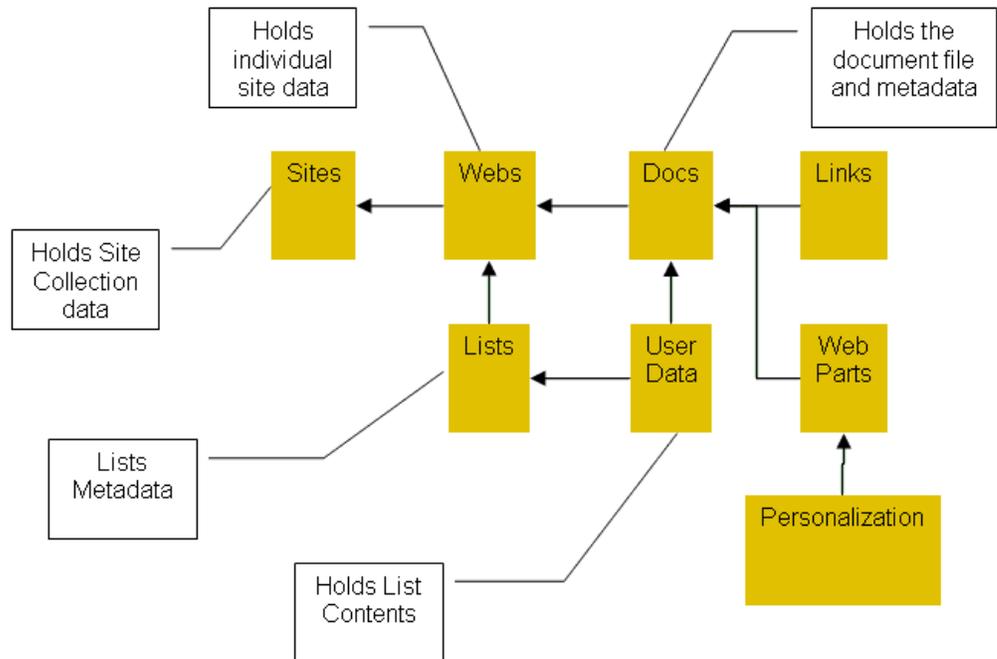


Figure 23: A logical representation of SharePoint's database

The two tables that store document contents are Docs and DocVersions. The Docs table stores the document content as well as metadata. DocVersions is only used if versioning is enabled. In both tables, the content column (SQL Server image data type) contains the file.

Recommendation

The lack of extensive Document Management API support in both WebSphere Portal and SharePoint means that any attempt to provide tight integration is not possible. However a portlet mimicking the SharePoint document library Web Part can provide a subset of functionality. The user would be able to view document lists, edit them in-place, view versioning information, and perform check in/out. The only feature that would not be available is workflow. We recommend this approach in the majority of situations.

However, if the use of PDM is mandated or approval workflow is required, then you need to copy the documents from the SharePoint document library into the PDM document repository .

Presence

Presence and awareness provide synchronous and asynchronous communication options among peers in the context of portal pages. Synchronous communications within WebSphere Portal is usually accomplished with Sametime instant messaging(IM), and with Messenger and Live Communications Server within SharePoint. E-mail provides asynchronous communications and SharePoint leverages Outlook and Exchange. WebSphere can use both Lotus Domino and the Microsoft offering.

The instant messaging servers provide presence information for their respective portals such as highlighting IM buddies who are on-line where the names appear on a portal page. In the case of WebSphere, you can see which people are looking at the same page.

WebSphere Portal awareness

WebSphere Portal includes people awareness features, so that people's names can appear in portlets as hyperlinks which enables users to contact people with whom they might want to work. Wherever people links appear, portal users can click the link to display a menu of actions for collaborating (contacting and working) with the person named by the link. If you, as portal administrator, have also configured a Lotus Sametime server to work with the portal, people links indicate whether a person is active, away, offline, or in a Do Not Disturb state.

When portal users click the name of (or the icon for) a person, a menu displays which provides actions linking them to other portal users. The actions that are visible on the person menu depend on these factors:

- Whether Lotus Companion Products, the Collaboration Center, or both, are set up to work with the portal.
- The online status (Active, Away, Do Not Disturb, Offline) of the person.

Lotus Workplace includes two tag libraries that enable portal developers to add the awareness functionality to their portlet. They are `people.tld` and `menu.tld`. The `people` tag library provides the code to enable presence awareness. The `menu` tag library enables the `people` tag library to provide options for users to interact with other people through the portal.

SharePoint awareness

User presence can be tracked everywhere a member name appears in a Windows SharePoint Services site. The presence menu integrates with Active Directory directory service, Microsoft Exchange Server, and Microsoft Windows Messenger to offer information such as free or busy status, office location, and manager. Menu actions include **Send E-mail** and **Schedule Meeting**. Presence is also indicated in Microsoft Office 2003 programs

wherever a reconcilable name is identified. The use of presence requires Microsoft Live Communications Server .

Options

Option 1: Messenger ActiveX control

Microsoft provides `MessengerWrapper.dll`, an ActiveX Wrapper for Messenger, as part of an example of creating a custom Microsoft Project Guide. You could deploy it to your portal users' workstations. Because the control does not include a user interface, you could embed The Messenger ActiveX control in a portlet with supporting JScript that would interact with the control to produce a user interface.

The result is, potentially, a "Messenger Lite", presented within the context of a portal page with contextual influence of the buddy list (different lists on different pages for the same user). You could also replace the `people` tag library to generate client-side code to enable presence capabilities.

For more information, see [Resources](#) for a link to the *Microsoft Windows Messenger Integration Custom Project Guide*.

In this model, the portal generates the client-side code, enabling the client, through the ActiveX control, to communicate directly with Live Communications Server.

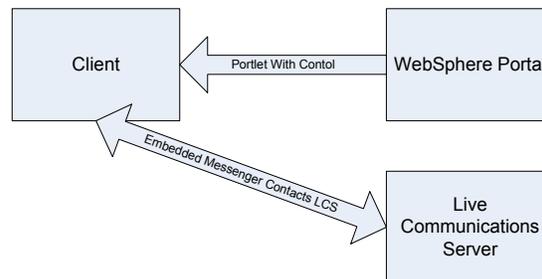


Figure 24: Live Communications Server integration with an ActiveX Control

The control would be embedded at the theme level of the portal running under WebSphere portal so that it would be present in all page views:

```
<object  
classid="clsid:ACECCC27-7CAD-44C6-8FCE-3DEE00506290"  
codebase="MessengerWrapper.dll" id="oMsgrObj" width="0"  
height="0" />
```

The following events are provided by the control:

```
OnContactStatusChange(pMContact, mStatus)
```

```

OnContactListAdd(hr, pMContact)
OnContactListRemove(hr, pMContact)
OnContactFriendlyNameChange(hr, pMContact, bstrPrevFriendlyName)
OnIMWindowCreated(pIMWindow)
OnIMWindowDestroyed(pIMWindow)
OnSignout()
OnSignin()

```

Event handling code is in `events.js` and `msgngrutil.js`, which are included in the Microsoft download.

Option 2: Real time communications DLL

Alternately, Microsoft provides a client API on Windows 2000 and higher, called `RTCDLL.DLL`, which enables interaction with Live Communications Server. The RTC Client API is a set of COM interfaces and methods which are accessible from C++ to wrap the client side of Microsoft's SIP support in Live Communications Server.

A portlet implementation may use the Java Native Interface (JNI) to invoke methods within the library to register a user's presence and query the presence of others. There is risk with any JNI programming as many of the safety nets provided by the Java environment are not provided during native code execution; these risks have to be balanced against any benefits you may believe you will enjoy using this approach.

See [Resources](#) for a link to the *RTC Reference*.

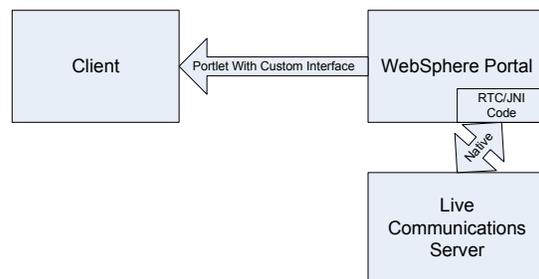


Figure 25: Live Communications Server integration with RTCDLL

Option 3: Custom LCS service portlet

Finally, you could install code with the Live Communications Server instance to expose a set of Web services specifically tailored to portal requirements. The LCS Management API enables robust interaction with LCS and LCS events to enable server applications and management functions.

A portlet implementation could use the SOAP-based Web services to interact with custom .NET services to provide the desired functionality. This option would provide greater functionality and safety than option two but would require significantly more coding.

See Resources for a link to the *Live Communication Server Management API*.

Option 4: Backend connection via SIP

In a recent [InfoWorld article](#), Jabber has recently announced a planned server-to-server gateway between its Extensible Messaging and Presence Protocol (XMPP)-based Jabber XCP platform and the IBM version of Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) used in Lotus Instant Messaging.

The company said that the technology will provide the basis for additional XMPP-SIP interoperability releases from the company in the months to come. If this is the case, you might be able to link Live Communications Server with Sametime server, which would allow users to experience full presence awareness in WebSphere Portal even if they are logged in as Window Messenger users on Live Communications Server.

Recommendation

Use the ActiveX control when a robust Messenger experience is required within the context of a portal page and a homogeneous IE client-base is in use. Options two or three may be required when there is a mixed client base. However, option two raises serious thread safety and scalability issues, not to mention the legal ramifications of connecting all users through a single session. Option three, entails a large amount of custom programming. The most attractive option would be (if time is not a factor) to use the backend connectivity provided by the Jabber gateway. It would require minimal custom coding, and would minimize maintenance requirements.

Conclusion

We hope you have come to the conclusion that integration between SharePoint and WebSphere Portal is a feasible proposition. We hope that this information helps you understand the technology challenges you face and provides a blueprint for addressing them.

This article addresses the technical issues that arise when integrating Microsoft SharePoint with IBM WebSphere Portal. However, because portals are very user-centric applications, you also need to consider the people and process issues associated with a project of this nature. Changes to the way people work will undoubtedly meet with resistance; therefore, your job is to demonstrate the added value of making the change. We advise following a phased approach.

We also advise keeping the user interface as similar to the existing interface as possible, to minimize the disruption to daily work. Generally, users just want to get their work done; they are not interested in the robust J2EE infrastructure, the best practice pattern based architecture, or the ability to consolidate servers. One of the ways to win them over might be to deploy new capabilities that are available only through WebSphere Portal, such as mobile device support or automatic transcoding so that international users can access intellectual capital. In short, remember the human aspect when planning and executing your portal federation strategy.

Resources

- The Apache Software Foundation; Apache Axis 1.1 Documentation
<http://ws.apache.org/axis/java/index.html>
- Configuring single sign-on using Tivoli Access Manager and WebSphere Portal
http://www.ibm.com/developerworks/websphere/techjournal/0406_singh/0406_singh.html
This article tells how to integrate Tivoli Access Manager for e-business V5.1 or V4.1 with WebSphere Portal V5.0.2 so you can provide authentication to a portal through Single Sign-On (SSO).
- Comparing the JSR 168 Java Portlet Specification with the IBM Portlet API
http://www.ibm.com/developerworks/websphere/library/techarticles/0312_hepper/hepper.html
This article explains the differences between the JSR 168 Java Portlet Specification and the IBM Portlet API of WebSphere Portal V5.0. It includes key concepts, examples, and programming recommendations.
- Guide to Websphere Portal 5.0
[ftp://ftp.software.ibm.com/software/websphere/portal/pdf/Guide to WebSphere PortalV5.pdf](ftp://ftp.software.ibm.com/software/websphere/portal/pdf/Guide%20to%20WebSphere%20PortalV5.pdf)
- Herman, Michael; Microsoft SharePoint Products and Technologies: Technical Overview
<http://www.microsoft.com/technet/prodtechnol/office/sps2003/plan/spst23to.mspx>
- *Jabber gateway aims to link XMPP, SIMPLE*, InfoWorld, July 22, 2004.
http://www.infoworld.com/article/04/07/22/HNjabberim_1.html
- JDBC driver for SQL Server 2000
<http://www.microsoft.com/downloads/details.aspx?FamilyID=9f1874b6-f8e1-4bd6-947c-0fc5bf05bf71&DisplayLang=en>
- Live Communication Server Management API
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rtc/rtc/about_the_rtc_server_management_api.asp
- Microsoft Windows Messenger Integration Custom Project Guide
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/pdr/PDR_WinMsgIntegratn_3499.asp
- Microsoft Windows SharePoint Services
<http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSSection.asp>
- MSDN library
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/spptsdk/html/soapnsMicrosoftSharePointSoapServer2.asp>
- RTC Reference
<http://msdn.microsoft.com/library/default.asp?url=/library/en->

us/rtcclnt/rtc/rtc_reference.asp

- SharePoint Recent Documents portlet
<http://catalog.lotus.com/wps/portal/portalworkplace?NavCode=1WP1000LR>
- SharePoint Events portlet
<http://catalog.lotus.com/wps/portal/portalworkplace?NavCode=1WP1000LS>
- WebSphere Application Server V5 InfoCenter
<http://www-306.ibm.com/software/webservers/appserv/infocenter.html>
- WebSphere Portal zone
<http://www.ibm.com/developerworks/websphere/zones/portal>
- WebSphere Portal product documentation
<http://www.ibm.com/developerworks/websphere/zones/portal/proddoc.html>
- WebSphere Portal and Lotus Workplace Catalog
<http://catalog.lotus.com/portalworkplace>
- Williamson, Heather. *XML: The Complete Reference*, Osborne/McGraw-Hill, 2001, New York.
- Using Credential Vault to Provide Single Sign-on for Portlets
http://www.ibm.com/developerworks/websphere/library/techarticles/0211_konduru/konduru.html
This article provides and explains four portlet applications that show how to use different credential vault techniques to maintain credentials.

About the authors



Tim Choo is a Software Engineer with a Lotus development team in Westford MA. Tim was previously a senior consultant in the Strategy and Change practice of IBM Business Consulting Services where he advised financial services clients on technology strategy issues. Tim came to IBM from Mainspring where he was a technical architect. Tim also spent several years building solutions with Lotus' Architected Solutions Group and developing solutions for customers as a consultant with Lotus Professional Services Asia-Pacific. You can reach Tim at choot@us.ibm.com.



Mark Moore is a Senior Software Engineer with a Lotus development team. He came to the team from the OnDemand Workplaces practice of IBM's Business Consulting Services where he spent the last two years helping clients with the IT-side of portal strategy. Mark came to IBM from KPMG, International, where he was Chief Architect for the firm's global knowledge management initiative, KWorld, which served 80,000 employees in 40 countries. He has spent the last nine years designing, developing, and deploying enterprise KM and collaborative systems on a variety of platforms. You can reach Mark at markmoor@us.ibm.com.

Trademarks

- DB2, IBM, and WebSphere are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.
- Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.