WebSphere Portal and Web Services
October 2002

Easily integrating remote portlet Web services with your portal.

By Thomas Schaeck, Dr. Carsten Leue and Peter Fischer, IBM Software Group

Contents

Introduction

Web services are an important way to make information and applications available through programs on the Internet. Portals need to allow the integration of Web services as simple data sources and as remote application components. Three important options are available for use of Web services with portals:

- *Portlets running on a portal server can access a Web service to obtain information or invoke remote methods provided by the Web service.*
- *Content or application providers can implement and publish interactive, user-facing Web services that plug and play with portals.*
- *Portals can publish local portlets as remote portlet Web services to make them available to other portals.*

IBM WebSphere® Portal for Multiplatforms supports all of these options. This white paper explains the concepts of traditional data-oriented Web services, as well as interactive, user-facing Web services. It also describes how you can use WebSphere Portal to set up distributed enterprise portal systems that allow your administrators to easily share portlets across portals. It explains how – using their portal administration user interfaces – content providers are able to use WebSphere Portal to publish content as remote portlet Web services, allowing other portal administrators to easily integrate content, without any programming effort.

IBM WebSphere Portal[1] thoroughly and flexibly integrates Web services, giving you more options to deploy Web services than other portal offerings. Supported by IBM's strong commitment to open standards, WebSphere Portal helps you use Web services effectively in a security-rich environment.

Portals provide personalized access to information, applications, processes and people. Typically, portals get information from local or remote data sources, such as databases, transaction systems, syndicated content providers or remote Web sites. They render and aggregate this information into composite pages to provide information to users in a compact and easily usable form. In addition to pure information, many portals also include a variety of applications, like e-mail, calendar, organizers, banking, bill presentment and host integration applications.

Different kinds of information or applications require different rendering and selection mechanisms. But all of them rely on the portal infrastructure and operate on data or resources owned by the portal, such as user profile information, persistent storage or access to managed content. As a result, most portal implementations provide a model that allows components, called portlets, to plug into the portal infrastructure. Typically, portlets are user-facing, interactive Web application components that run on the portal server, processing input data and rendering output. Figure 1 shows an example of a portal page generated by WebSphere Portal.

Content is often provided by external services and displayed by specific local portlets running on the portal. This approach is feasible for establishing the base function of a portal, but does not help to enable dynamic integration of business applications and information sources into portals.

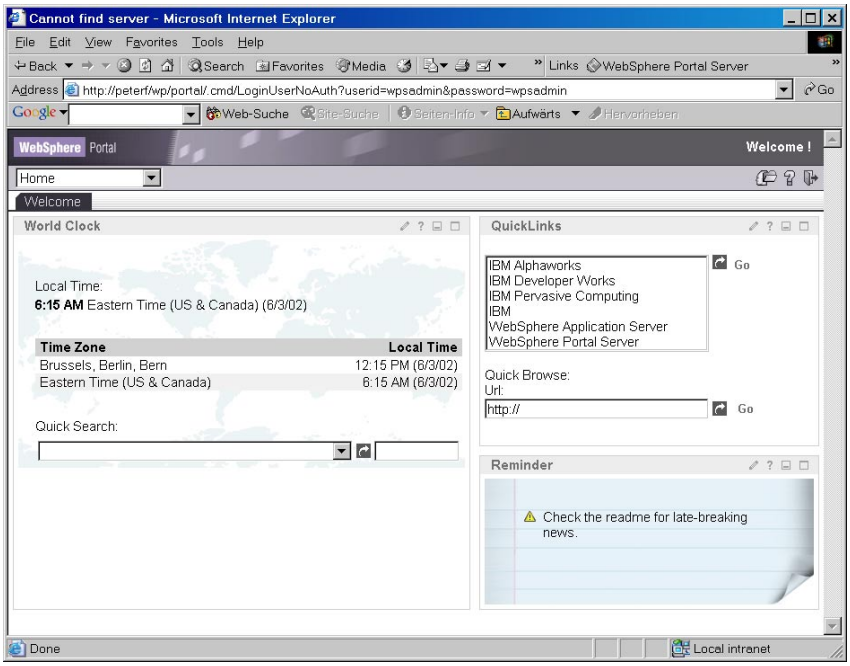

Figure 1. IBM WebSphere Portal for Multiplatforms, Version 4.1

Consider the following scenario. An employee portal manager wants to include a human resources (HR) service that calculates variable pay for employees and an external weather service that provides weather forecasts. Figure 2 shows a solution for this scenario in which an HR portlet and a weather portlet run locally on the portal server and access remote Web services to obtain the required information.

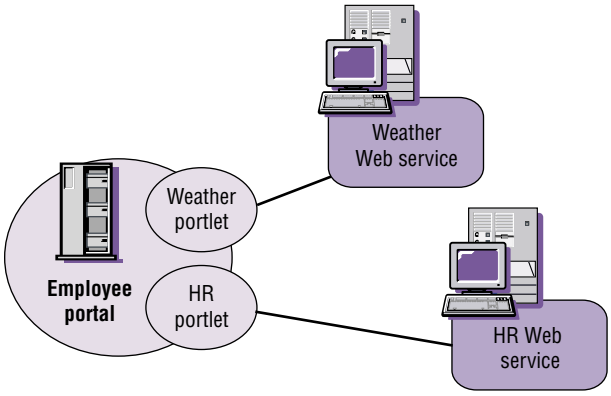

Figure 2. Local portlets using a Web service

The HR portlet uses an HR Web service to calculate the variable pay. By default, the portlet displays a form to query the required input data such as the employee's job title. When the employee provides the data to the HR portlet, it invokes the HR Web service to calculate the variable pay based on that data. The HR portlet receives the information from the Web service and displays it as a page fragment. By default, the weather portlet displays weather forecasts for configurable locations and allows the user to select locations in an edit mode. When the weather portlet is invoked during page aggregation, it requests the most recent forecasts for the selected locations from the weather Web service and renders a page fragment that displays those forecasts.

This approach works only when all portlets are physically installed at the employee portal; the process of making new portlets available is tedious and expensive. To integrate HR information in the portal using local portlets as previously described requires the HR department to implement the HR portlet and give it to an administrator of the employee portal to install it. Or an employee portal developer could implement the HR portlet according to the interface description of the HR Web service. Creation of the weather portlet requires a similar process. Either results in significant cost and loss of time.

A more efficient process is to use HR and weather Web services that include application and presentation logic producing markup fragments that are easy to aggregate at the consuming portal, as shown in Figure 3.

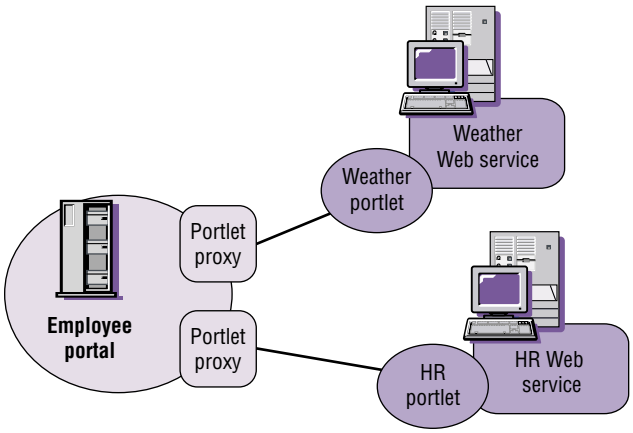

Figure 3. A portal using remote portlet Web services

Instead of providing only raw data or single business functions that still require special rendering on the portal side, remote portlet Web services are user-facing, interactive Web services that include presentation. They are easy to aggregate and can be invoked through a common interface using generic portlet proxy code on the portal side. You don't need to install special portlet code on the portal. Using generic portlet proxies that use all remote portlet Web services that conform to the common interface eliminates the need to develop service-specific portlets to run on the portal.

The administrator's task is much easier because the administrator can add portlets dynamically to the environment, and users benefit by having more services available to them in a timely manner. Administrators can easily integrate remote portlet Web services with a portal by using the portal administration user interface to locate the services and bind to them with a few mouse clicks. As a result, the portal creates a new, generic portlet proxy instance bound to the remote portlet Web service. This means that remote portlet Web services appear like local portlets, easily managed by users.

Portals need to apply an appropriate combination of caching and concurrent invocation wherever using remote portlet Web services, so that the number of remote calls is minimal. And, where possible, remote calls can occur concurrently to minimize overall page response times.

Portlets in WebSphere Portal architecture

WebSphere Portal provides an open architecture to allow for systems using remote portlets, as described in the previous section. It supports local and remote portlets to exploit the benefits of both approaches (see Figure 4). Local portlets are tightly integrated with the portal server and run on the same physical server or cluster of servers. Remote portlet Web services run on remote servers at other places in an intranet or the Internet and are loosely coupled to the portal server.

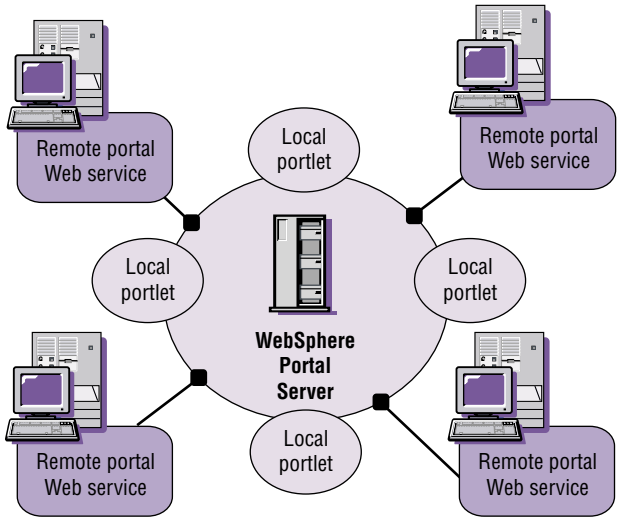

Figure 4. Local and remote portlets

The WebSphere Portal architecture uniformly manages local portlets and remote portlet Web services and integrates both seamlessly in its management user interface. WebSphere Portal uses portlet proxies to plug in remote portlet Web services, which allows the remote services to function like local portlets.

Figure 5 shows a high-level view of the relevant components, interfaces and protocols.

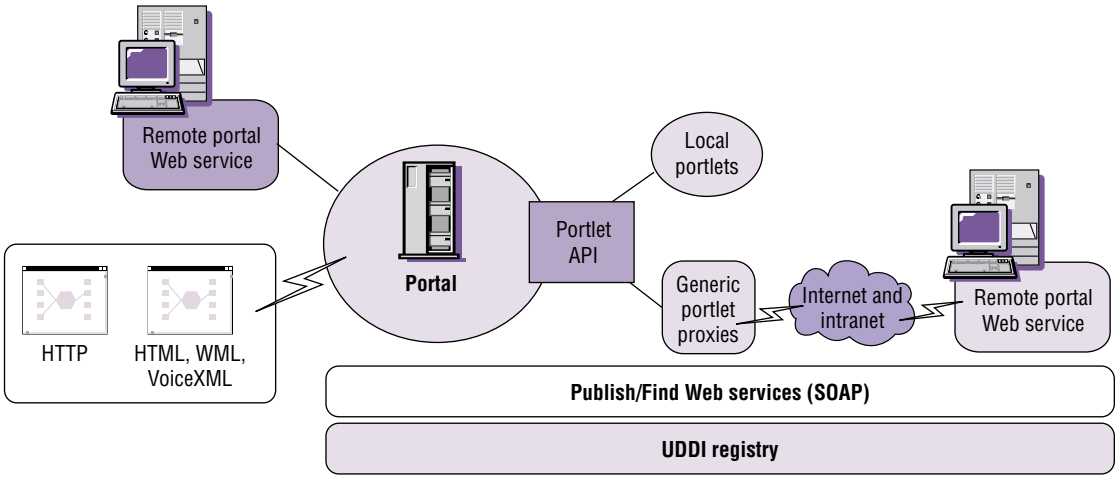

Figure 5. WebSphere Portal interfaces and protocols

Portal clients access the portal through HTTP, either directly or through appropriate gateways like Wireless Application Protocol (WAP) or voice gateways. The markup languages used by these devices may be very different. WAP phones typically use Wireless Markup Language (WML), iMode phones use compact HTML (cHTML), voice browsers primarily use VoiceXML and the well-known PC Web browsers use HTML.

When aggregating pages for portal users, the portal invokes all portlets that belong to a user's page through the portlet application programming interface (API). There are two different kinds of portlets:

- ***Local portlets***. *These portlets run on the portal server itself and are deployed by installing portlet archive files on portal servers. They are invoked by the portal server through local method calls, which means they provide minimal latency times. However, installing a local portlet usually requires assurance that it is not erroneous or malicious.*

- ***Remote portlets***. *They run as Web services on remote servers, typically published in a Universal Description, Discovery and Integration (UDDI)[2] registry so that they are easy to locate and bind to. Portlet proxies are generic local classes that invoke portlets located on remote servers through Simple Object Access Protocol (SOAP)[3].*

While local portlets can provide a large part of the base function for portals, the remote portlet concept allows dynamic binding of a variety of remote portlet Web services without any installation effort or code running locally on the portal server.

You can wrap and publish local portlets as remote portlet Web services to provide them to other WebSphere Portal instances. And conversely, you can integrate remote portlet Web services with WebSphere Portal by wrapping them in a portlet proxy written to the portlet API.

Portlet API and portlet invocation

Portlets are pluggable, user-facing, interactive Web application components running inside a portal portlet container and written to a portlet API. Unlike common servlets, portlets are special servlets that need to run embedded in a portal environment. This result in a portlet API extending the servlet API with portal-specific context and services. While servlets communicate directly with their clients, portlets are invoked indirectly through the portal application that enriches requests with portal context information. To properly run in the context of a portal, portlets must produce content suitable to aggregate in composite pages—that is, portlets must produce markup fragments that can be aggregated. Portlets may not use redirects nor send errors to the client; these commands are reserved for the portal itself.

When WebSphere Portal receives a servlet request resulting from a user clicking a link or button inside a portlet, it generates and dispatches an action for the portlet affected by parameters in the request. It then invokes all portlets for display through the portlet invocation interface (shown in Figure 6). Portlets can implement action handling and rendering in separate methods, which allow WebSphere Portal to employ portlet-markup-caching and concurrent portlet invocation when aggregating portal pages.

Portlet-markup-caching

WebSphere Portal uses markup-fragment-caching for portlets that use significant time for computations and cause latencies waiting for data from remote sources. The portlet markup is retrieved from a per-portlet cache, unless it has expired or was explicitly invalidated.

Concurrent portlet invocation

WebSphere Portal uses concurrent invocation for portlets that incur significant waiting times for retrieving external data. In fact, the latency time for the whole page is only the maximum latency time of each portlet on the page—in the optimal case.

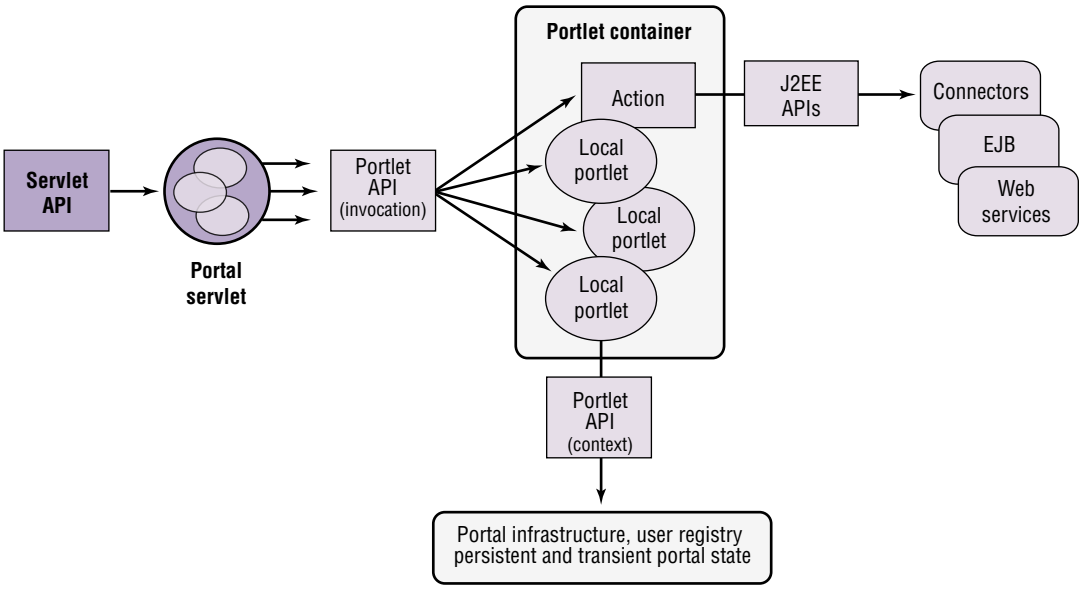

Figure 6. Invocation of portlets in WebSphere Portal

While portlets must implement the invocation methods the portlet API requires, internally they can be implemented differently. A pattern that works well for portlet programming is the model-view-controller (MVC) pattern. It separates the portlet function into a controller that receives incoming requests, invoking commands that operate on a model that encapsulates application data and logic, and calls views for presentation of the results.

Portlets have access to portal-related functions and data, including access to user profile information, persistent portlet instance data and portlet settings. Apart from portal-specific functions, portlets can use all Java™ 2 Platform, Enterprise Edition (J2EE) APIs that are available to servlets, as well as vendor-provided connectors to access back-end data and applications or even services in the Internet.

For easier deployment, portlets can be grouped in portlet applications that are packaged into portlet archive files. These are special J2EE Web Application Repository (WAR) files containing a deployment descriptor with portlet-specific extensions, Java classes, jar files and resources.

Web services

Web services have been developed to allow business applications to communicate and cooperate over the Internet, creating a new paradigm in how it is used. While traditional applications interacting with services in the Internet know of those services beforehand and need to be pointed to them manually, the Web services approach allows applications to find services in a standardized directory structure and bind to them with minimal human interaction (see Figure 7).

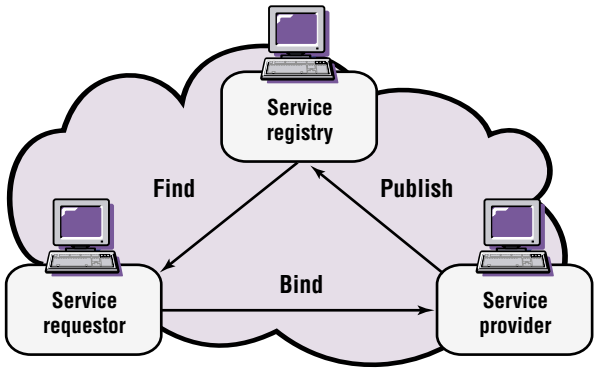

Figure 7. Publish, find and bind

Web services distribute objects across Web sites, allowing clients to access them through the Internet. Global service registries are used to promote and discover distributed services. The user that needs a particular kind of service can make a query to the global service registry to find services that suit the user's needs. The user can select one of the services, bind to that service and use it for a specified period of time. Service discovery and selection can occur without human interaction, which means services can switch very quickly. With automated service discovery, it is possible to establish robust networks of services. If multiple Web services exist that provide identical functions, a user can easily switch to a backup system when the service fails.

For registration and discovery of Web services, the most important standard is UDDI. For communication between Web services, the most important standards are SOAP and the associated Web Services Description Language (WSDL)[4] for formal description of Web service interfaces and bindings.

From a portal perspective, you can differentiate between two kinds of Web services—the traditional data-oriented Web services and presentation-oriented, user-facing, interactive Web services (see Figure 8).

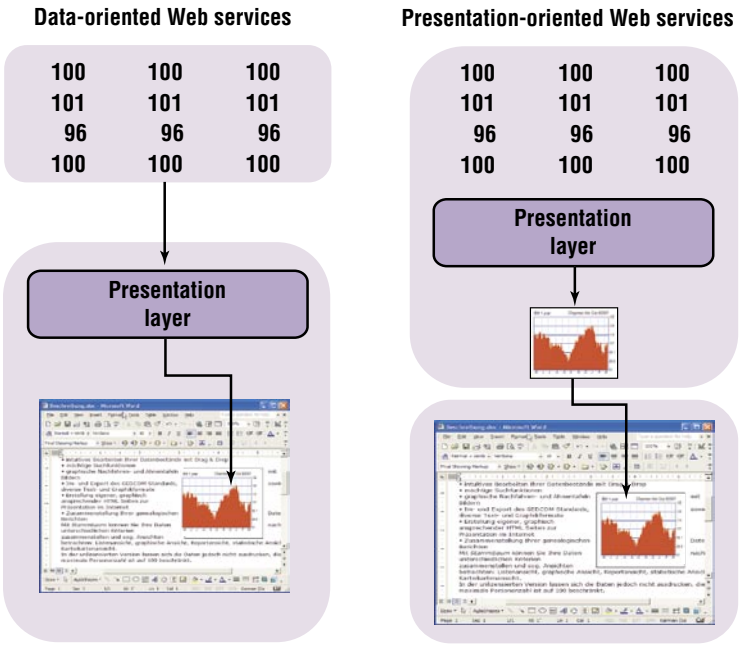

Figure 8. Data-oriented Web services compared to presentation-oriented Web services

Data-oriented Web services are Web services that receive SOAP requests and return data objects encoded in XML documents in the SOAP response. The signatures of their SOAP operations, as well as the structure and semantics of the returned data, are typically service-type-specific. It is the responsibility of the service consumer to process the received data in a service-specific manner and generate any required presentation. While this approach works well for applications that require specific data and can use and process the data received, it is not appropriate for portals that need to quickly integrate content and applications from a variety of sources.

User-facing Web services include presentation as a part of the service itself. That is, they don't simply provide raw data for processing and presentation by the consumer. Instead the user-facing Web services can produce markup fragments that can be easily aggregated by portals. User-facing Web services may also include user interaction and are well suited for integration and use in portals because they can be aggregated by portals. They can participate in user-action-processing in a generic way with no service-specific presentation code required on the consumer's side. The following section discusses how portlets, running locally on WebSphere Portal, use data-oriented Web services. And how you can integrate user-facing, interactive Web services with WebSphere Portal.

Data-oriented Web services used by portlets

WebSphere Portal allows portlets to access data-oriented Web services to obtain data from remote systems. When using data-oriented Web services with a given interface, the portlets need to contain service interface-specific code that matches the Web services operations and their particular signatures.

When a portlet receives a request that requires invocation of a remote service, the portlet makes calls on a service-specific SOAP proxy. The SOAP proxy takes the parameters and builds them into a programming language-independent SOAP request. This request is then sent to the remote Web service. The Web service receives the SOAP request, extracts the parameters and invokes the Web service implementation with these parameters. When the service implementation returns the result, the Web service builds the result into a programming-language independent SOAP response and sends it back to the SOAP proxy on the user's side. The SOAP proxy extracts the result data and returns it to the portlet as the appropriate language-specific objects (see Figure 9).

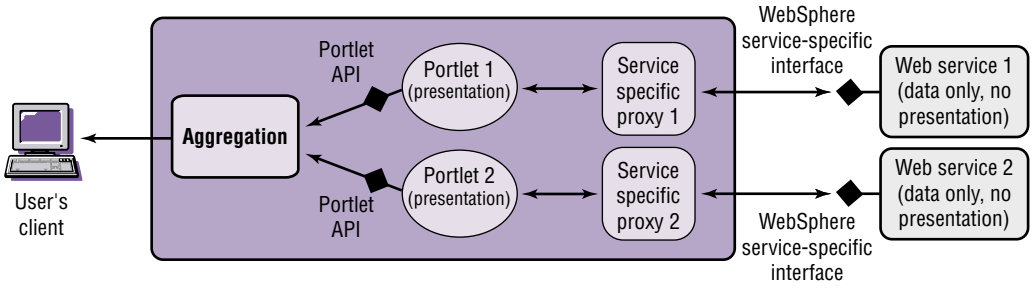

Figure 9. Portal consuming data-oriented Web services using service-specific portlet code

You can formally describe Web services by using WSDL interface descriptions. With appropriate tools, you can use the descriptions to generate service interface-specific SOAP proxies for different programming languages. This means that you can automatically generate the service-specific portlet. To simplify writing portlets using data-oriented Web services, IBM provides tools that can automatically produce the appropriate SOAP proxy code for the Java programming language from a Web services WSDL interface description.

Portlets that consume data-oriented Web services through a SOAP proxy use the service-specific Java SOAP proxy. As a consequence, those portlets indirectly depend on the particular operations of the data-oriented Web service that determines the proxy's methods. The portlets need to know how to make calls to the SOAP proxy and how to process and render the results. This means that, in addition to the service-specific proxies, the individual portlets must contain significant amounts of code tailored to the particular Web service interface.

Remote portlet Web services

To allow for dynamic integration in WebSphere Portal, remote portlets need to be provided as Web services conforming to a well-defined, common remote portlet Web service interface description defined in WSDL. This interface description specifies a common set of operations and signatures corresponding to the portlet API for local portlets. Remote portlet services do not have to be implemented in Java; they can be implemented in other languages as well, as long as they all adhere to the remote portlet Web services WSDL description.

When invoking a remote portlet, the portal uses a generic portlet proxy to invoke the remote portlet Web service. The portlet invokes the generic portlet proxy exactly like it would invoke a local portlet. The generic portlet proxy internally invokes a SOAP proxy to put all parameters into a SOAP request and sends it to the remote server hosting the remote portlet Web service. Because all remote portlet Web services adhere to the same service interface definition, the same SOAP proxy class can be used for all of them. On the producer's side, the remote portlet Web service extracts the information from the incoming request and calls on the remote portlet implementation.

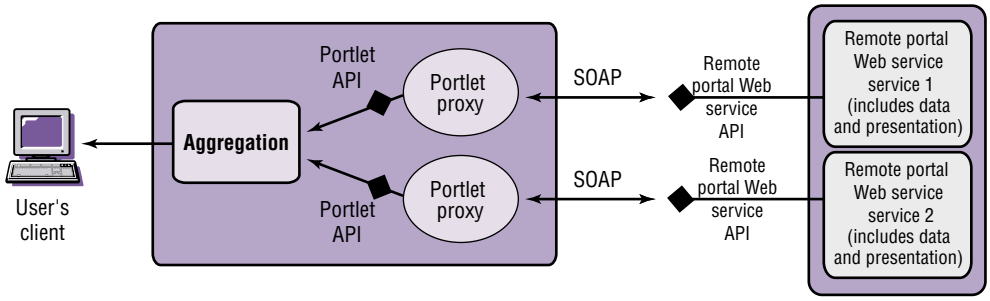

Figure 10. Portal consuming remote portlet Web services using generic portlet proxies

The result is put into a SOAP response and sent back as the reply to the SOAP proxy, which then extracts the response for the portlet proxy. Like a local portlet, a result object is returned to the portal engine that initiated the request.

The big difference between remote portlet Web services and data-oriented Web services is that remote portlet Web services are user-facing, interactive services that include presentation and all have one common interface. This means that the remote portlet Web services can be integrated in portal servers in a plug-and-play fashion and do not require any service-specific code to run on the consuming portal.

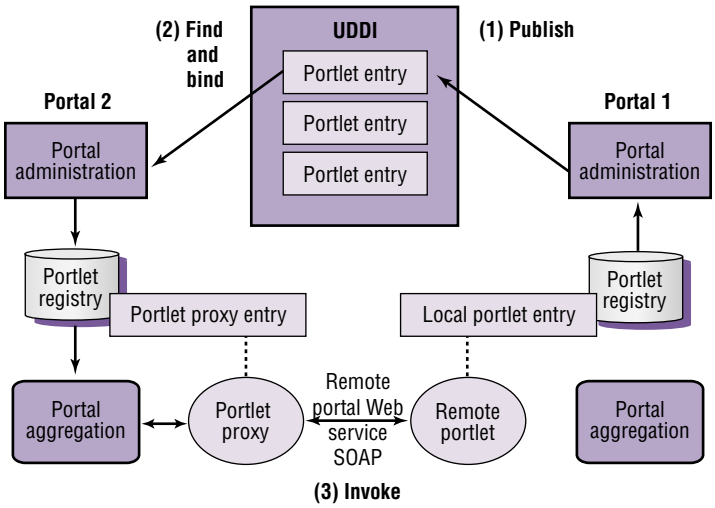

Figure 11. Publishing, finding and binding remote portlets

Content or application providers who want to offer remote portlet Web services can publish their service entries to a UDDI directory, referencing the remote portlet Web services interface description. When a remote portlet has been published, portal administrators can use their portal administration user interface to search the UDDI directory for Web services that implement the remote portlet Web services interface. The administrators can then make some of the matching portlet Web services available for their users by adding them to their local portlet registry (see Figure 11). Once the portlets are in the registry, users can select them to be displayed on their personal pages, like local portlets.

Use of remote portlet Web services in WebSphere Portal, Version 4.1

In WebSphere Portal, Version 4.1, the mechanisms for publishing portlets as remote portlet Web services, finding remote portlet Web services in a UDDI registry, binding to them and using remote portlets are smoothly integrated with the administration user interface. There are four different dialog flows that apply to the use of remote portlet Web services (see Figure 12).

- *Managing UDDI registries*

Administrators manage a list of UDDI servers they want to use for querying and publishing.

- *Publishing portlets*

Administrators can publish portlets to make them available for use by other portals as remote portlet Web services.

- *Finding and binding remote portlets*

Administrators find remote portlet Web services in a UDDI registry and bind to these services to make them available for portal users.

- *Using remote portlets*

Users select and transparently use remote portlet Web services that have been integrated by administrators, as easily as local portlets.

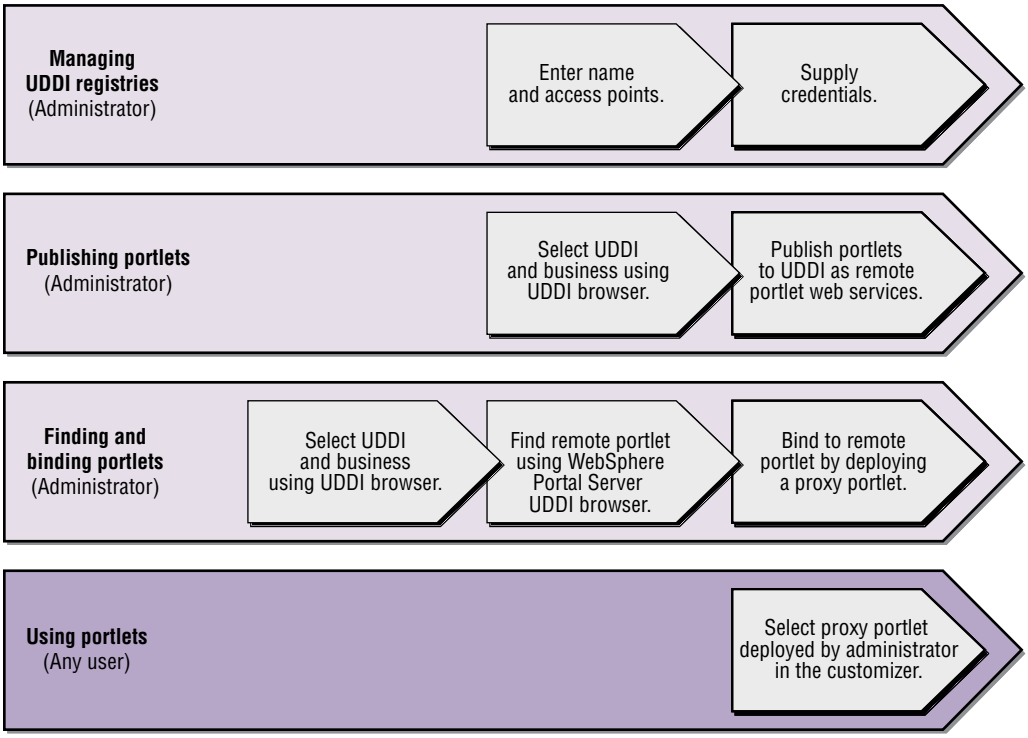

Figure 12. Web services-related dialog flows in WebSphere Portal

Managing UDDI registries

Before publishing portlets or looking for remote portlet Web services, one or more UDDI servers must be configured. WebSphere Portal manages these configurations, centrally stores the user name and password securely in the system-wide credential vault and makes use of the registered servers in the publish-and-find dialogs.

Using the Manage Web services dialog, the administrator can register a new UDDI registry. WebSphere Portal prompts for a display name for the new registry, the inquiry URL and the remote portlet Web services tModel key that is valid in the scope of the UDDI registry. To use the registry for publishing, a publish URL needs to be supplied in addition to the credentials required by the UDDI registry to publish services.

Publishing portlets

Publishing portlets is done through the Publish portlets dialog. The user can select the portlet or portlets to be published using the standard Get portlets dialog. By default, the name under which the portlet is published to the UDDI registry is taken from its registration information within WebSphere Portal. However, the portal administrator has the option to modify the name and description for each portlet for each supported locale. After names and descriptions have been defined, the administrator selects one of the UDDI registries (previously configured through the Manage Web services dialog) to publish to. The publishing dialog displays a list of all business entities in the selected UDDI registry that the current user has the right to publish to. If none of the existing businesses is appropriate or if there is no such business entity, the user has the option to create a new business entity.

Finding and integrating remote Web services

Finding and integrating remote Web services works like this: The user can select a UDDI registry to search in, using the Integrate a Web service as a remote portlet option. The administrator can search for all portlets, all portlets in a business entity or portlets by name. The dialog will find only those portlets that are compliant with the remote portlet Web service interface by making use of the associated tModel key.

From the search result, the administrator can then select the remote portlet Web services to integrate, which allows WebSphere Portal to add the remote portlet to its portlet registry and make it available for portal users.

Using a remote portlet is as simple as using a locally installed portlet. Users can select remote portlets from the Work with pages dialog just like they select local portlets.

The following sections show how these functions are realized in WebSphere Portal, Version 4.1. The screen captures shown in the figures are made from the actual product.

Publishing portlets as remote portlet Web services in UDDI
Only portal administrators may publish portlets as remote portlet Web services to a UDDI directory to make them available for integration with other portals. After logging in for the first time, the administrator needs to select the Manage Web services dialog to configure a list of UDDI registries, as shown in Figure 13.

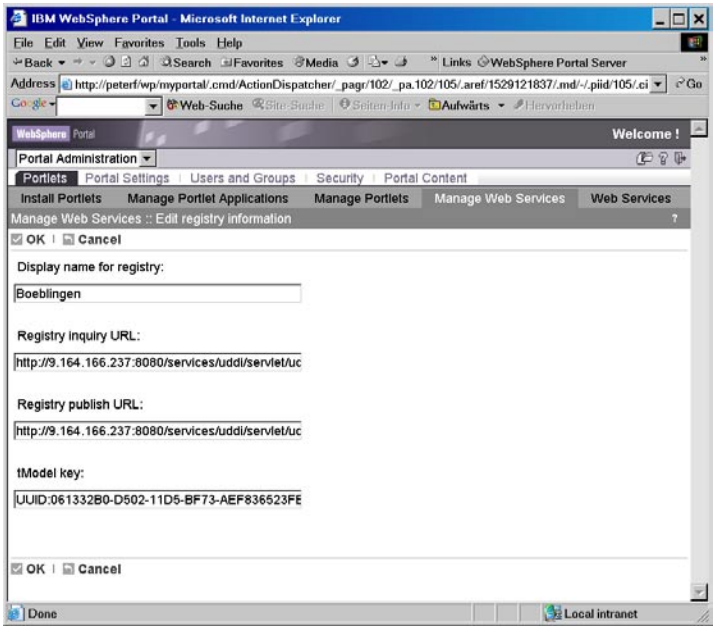

Figure 13. Administrator configuring UDDI

The Provide registry authentication information dialog allows the administrator to provide the credentials required for publishing services to the registry (see Figure 14).

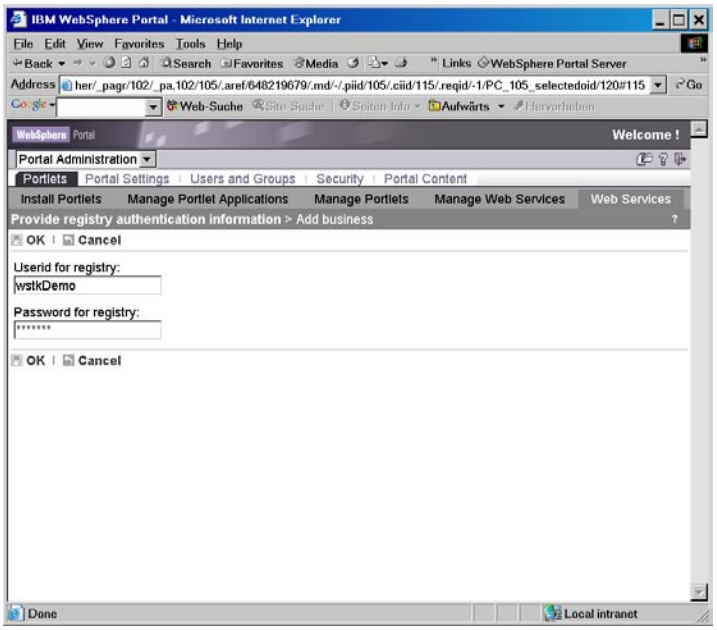

Figure 14. Providing UDDI credentials

To make a portlet available as a remote portlet Web service in UDDI, the administrator uses the Publish portlets dialog. The administrator selects the portlets to publish, the destination registry and business, and presses the publish button to finally publish a particular portlet to UDDI as a remote portlet Web service (see Figure 15).

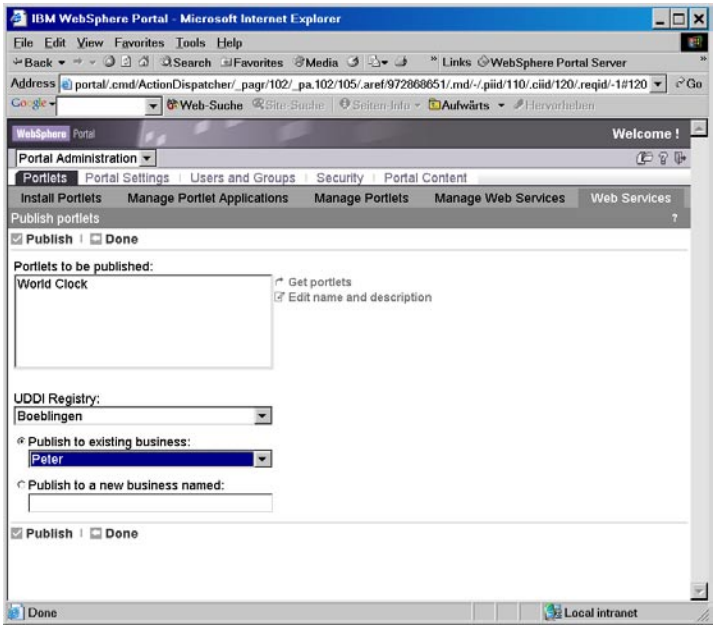

Figure 15. Selecting the portlet to be published

Finding and binding to remote portlet Web services

Only administrators can find and bind to remote portlet Web services. To find a remote portlet Web service, the administrator selects the Integrate a Web service as the remote portlet dialog (see Figure 16).

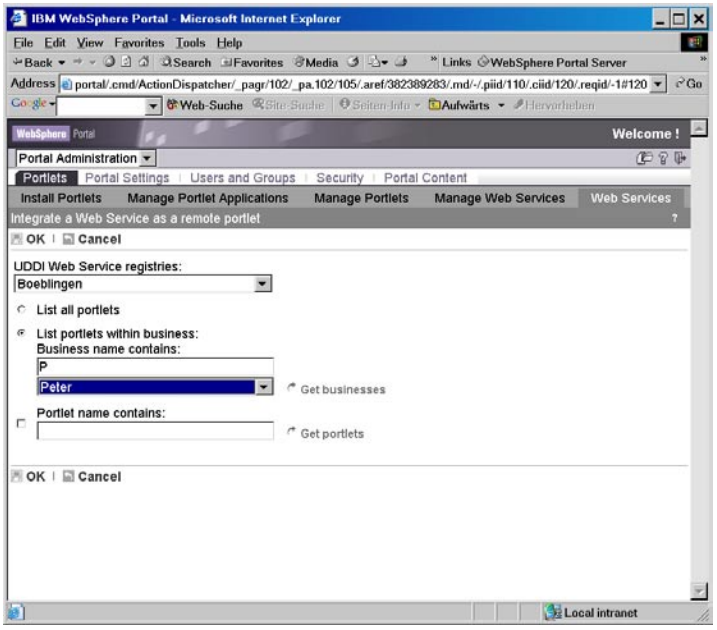

Figure 16. Finding remote portlet Web services

When the administrator clicks Get portlets, WebSphere Portal queries the UDDI directory for all remote portlet Web services that meet the search criteria (see Figure 17).

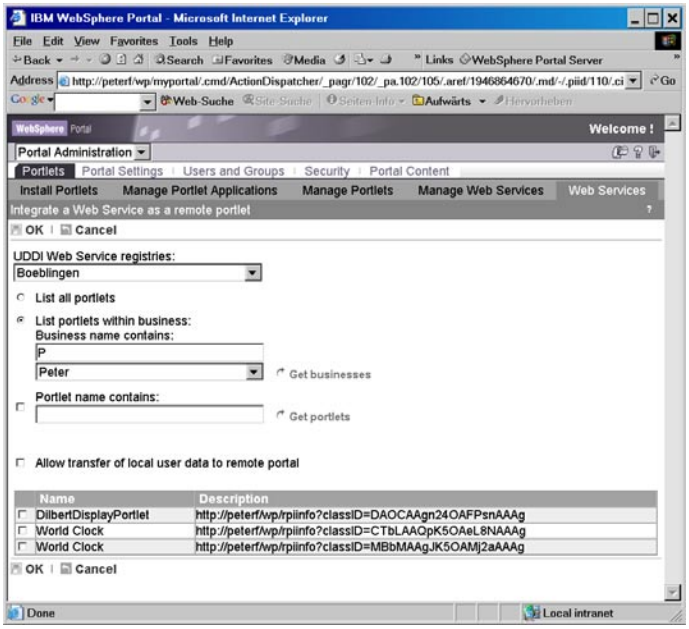

Figure 17. The portal lists remote portlet Web services.

To add the remote portlet to the WebSphere Portal portlet registry to make it available to users, the administrator checks the World Clock portlet from the search result list and selects OK. As a result, WebSphere Portal gets the relevant information about the remote portlet Web service and creates a new portlet proxy entry in its portlet registry, making the remote portlet available to users.

The administrator can also specify whether user information may be transferred to the remote portlet Web service.

Using remote portlet Web services

For users, remote portlet Web services are entirely transparent. After logging in, the user can use the Work with pages option to navigate to the WebSphere Portal Customizer screen. This screen lets the user select portlets and put them on a page (see Figure 18).

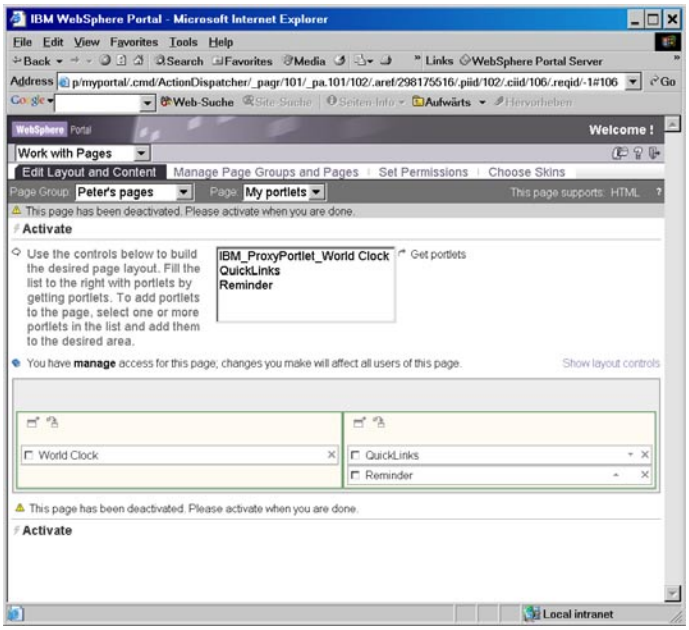

Figure 18. Configuring pages

The user can select a proxy for a remote portlet, like you can for any local portlet. After selecting the remote portlet in the customizer, it is displayed on the user's page (see Figure 19). The World Clock portlet in this screen is remote; the other portlets are from a local source.

Application examples

After explaining the basic concepts and presenting the capabilities of WebSphere Portal Server to publish, find and bind to remote portlet Web services, the following section discusses examples of applications that show how these capabilities can be exploited.

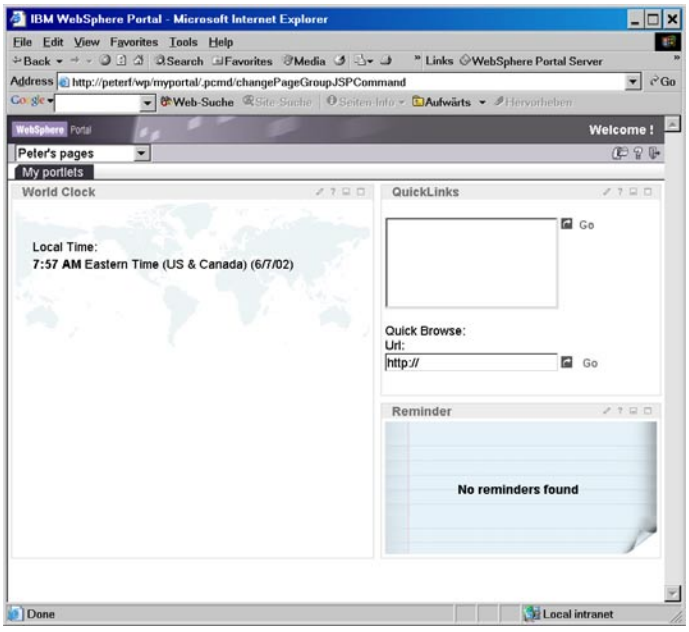

Figure 19. A portal page with a remote portlet

Content providers publishing remote portlet Web services

Most content providers publish their content live on the Internet using HTTP or file transfer protocol (FTP) servers, or they provide client software that replicates and caches content using proprietary protocols. Either way, integrating content with a portal is a difficult task. Portals may provide some portlets that support particular content sources out of the box. When an administrator sets up a portal to integrate content from different sources, it's expensive and time-consuming because the administrator must develop and install additional portlets for the remaining unsupported content. Without supported portlets, portal owners and content providers are at a disadvantage because it's difficult to include a full range of content, limiting business growth. It also limits your ability to control how a subscriber's portlet renders content.

For easy integration of content in portals without any programming or service effort, content providers can use WebSphere Portal Server to surface their content as portlets and publish these portlets as remote portlet Web services in the public, global UDDI directory.

To provide this new value to customers, the content provider runs a WebSphere Portal installation that serves remote portlets in addition to the more traditional content server. After the content provider has used the publish function provided by WebSphere Portal to advertise the remote portlet Web services in UDDI, portal administrators who want to use content from the content provider can simply look up the content provider's business entry in the UDDI directory and bind to remote portlet Web services that provide the desired content. The portlets on the content provider's server become available immediately without any programming or installation effort and can be used by portal users. At the same time, WebSphere Portal provides the content provider itself with a portal, so that the content provider can also make content available to users directly. The content provider can also easily use the same setup because all content provider clients have to test and review the actual presented outcome on a remote client portal.

Portals publishing local portlets for remote use

Portals initially operated in isolation from each other, but the demand for cooperation between portals started within big corporations. In the near future, corporate portals will also need to cooperate with supplier or customer portals, so that eventually portals will need to cooperate over the Internet, as well as within intranets. The introduction of this white paper described a scenario in which an employee portal used a service provided by the HR function within a corporation.

In that scenario, assume HR runs a portal that provides various HR-related portlets. Some portlets serve only the HR staff, for example, a payroll portlet or a curriculum vita (CV), or resume, portlet. Some portlets serve all employees, such as a variable pay portlet and an HR information portlet, providing HR-related news.

Assuming the corporation has its own corporate UDDI directory that is only accessible from an intranet, an HR portal administrator could use the WebSphere Portal server publish function to create remote portlet Web service entries for the variable pay portlet and the HR information portlet in the corporate UDDI directory. This would allow these portlets to become available for integration in other portals in the corporation. Using the WebSphere Portal Server built-in UDDI browser, the administrator of the employee portal could find remote portlet Web services published by the HR portal and integrate them as part of the administrator portal with only a few clicks.

Standardization

Portal owners can benefit from the ability to publish and share portlets running on WebSphere Portal. Content and application providers can implement and provide remote portlet Web services so that the services can easily be found and integrated by WebSphere Portal installations. Beyond the immediate benefits, however, exploiting the full potential of the remote portlet concept requires development of a standard.

IBM – together with portal vendors, content and application providers and other organizations that have a vital interest in creation of a standard for pluggable services for portals – have initiated the OASIS Web services for remote portlets (WSRP) standards initiative (see http://oasis-open.org/committees/wsrp/). The OASIS WSRP technical committee has the charter to define a standard for user-facing, interactive Web services that plug and play with heterogeneous portal servers[5]. The standard will define meta-data to describe WSRP services and illustrate how to locate the services and the protocol for use of a WSRP service by consumers[6].

WebSphere Portal will support the emerging WSRP standard for using remote portlets and for publishing local portlets as soon as the standard is available.

Summary

The remote portlet Web services approach allows deploying distributed portals cooperating within an intranet or over the Internet. IBM WebSphere Portal for Multiplatforms, Version 4.1 supports the remote portlet Web services concept to allow publishing of local portlets as remote portlet Web services and integration of remote portlet Web services in WebSphere Portal software-based portals using only a few clicks by an administrator, without any programming effort.

At the same time, the ability to host portlets and publish them as remote portlet Web services that can be easily integrated with portals makes WebSphere Portal a versatile platform that lets content and application providers offer their services in an easy-to-use form.

With IBM WebSphere Portal, your company can easily publish, find and bind, and use standards-based Web services and Web services portlets. The extensive and flexible integration of Web services in WebSphere Portal should drive increased use of portals and Web services within your organization, resulting in greater employee productivity.

For more information

For more information about WebSphere Portal, visit:

ibm.com/websphere/portal

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
10-02

[1] See the "Guide to WebSphere Portal" IBM white paper at www.**ibm.com**/websphere/portal.

[2] See "UDDI Technical White Paper" at http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.

[3] See "SOAP Version 1.2, Part 1: Messaging Framework" at http://www.w3.org/TR/soap12-part1/.

[4] See "Web Services Description Language (WSDL), Version 1.2" at http://www.w3.org/TR/wsdl12/.

[5] For more information about OASIS Web services for Remote Portlets, visit http://www.oasis-open.org/committees/wsrp.

[6] For more information about the Java Specification Request 168, visit http://www.jcp.org/jsr/detail/168.jsp.

G325-2098-00