Portal software solutions
White paper

WebSphere software

Lotus software

Evaluating the decision to buy or build a portal.

The value of WebSphere Portal compared to a proprietary portal in a J2EE technology-based environment.

By Kevin P. Haiduk, IBM Business Consulting Services, with contributing author Chris Kergaravat, IBM Software Group

August 2004

Contents

Executive summary

Modern companies have invested a significant amount of time and money over the years to develop complex intranet and extranet sites. These companies now face Web site challenges, such as insufficient organization, a lack of personalized navigation and inadequate search tools. Portal technology was created in response to these problems—to serve as simple, unified access points to Web applications. Most businesses understand the value of a business portal to its users and—anticipating that it will be a large investment—attempt to build a concrete business case to justify the costs. They face a difficult decision: whether to base the case on an off-the-shelf portal platform, like IBM® WebSphere® Portal software that can immediately be put into service, or to custom-build the functionality that WebSphere software products offer.

The purpose of this white paper is to evaluate the comparative efforts (based on IT development and support time) that are required to build a portal using WebSphere Portal software or to develop that same portal using a custom Java™ 2 Platform, Enterprise Edition (J2EE) technology-based framework approach.

To create a portal, you must build its portlets while considering two other layers of functionality: the foundation layer holding services, and an administation interface that can maintain the portlets in addition to the overall look and feel and security of the portal. This white paper examines each of the three functional areas and the different approaches you can use to build them.

- Core functionality. *The base functionality that is needed to personalize user profiles, collect content, manage content and provide single sign-on, portlet messaging and collaboration capabilities.*
- Portlets. *Portlets, the building blocks for Web applications within WebSphere Portal, are the medium for content display and disparate application logic. Portlets interact with the core portal functionality to deliver personalized content and virtually seamless application integration.*
- Maintenance. *Maintenance is the effort required to maintain and to add new functionality to the portal through portlets.*

For each area of functionality, comparisons are made between the use of WebSphere Portal (using appropriate development approaches) and developing a custom portal application.[1] Development time for the custom approach is determined by examining the efforts in three customer case studies (see Appendix 1) in which portlets of similar functionality were developed. The comparative effort to build or maintain each capability is summarized and time savings are tallied for each case. By multiplying the time savings from each portlet type by the number of various portlet types normally deployed in a typical, enterprise-portal implementation, a framework can be established to estimate the value of WebSphere Portal. This white paper includes a spreadsheet that helps calculate the savings with the number of portlets of each type that will be deployed.

An overview of portal technology

Initially, portals provided a centralized access point for enterprise-related functions, such as e-mail, company information, workgroup systems and commonly used business applications. However, modern portals have evolved to do much more. Portals today offer valuable capabilities, such as security, search, collaboration and workflow. A well-designed portal can provide a common user interface and content base that can be integrated and leveraged across all portal applications (for example, portlets), to deliver a unified, collaborative workplace. Undeniably, portals are the next-generation desktop, delivering on demand business applications to many varieties of client devices, and allowing users to customize their individual interfaces. In the future, every conceivable Web application might need to be customizable by the user to support a true on demand, dynamic workplace.

IBM customers using IBM WebSphere Application Server have long had a scalable and flexible platform to develop and deploy Web applications. With the addition of WebSphere Portal software, they can also use the portal to provide commercial off-the-shelf (COTS) functions to develop many of the components that are reused from application to application. The core infrastructure and portlet application programming interface (API) in WebSphere Portal deliver the platform and the tools to support virtually all of the core requirements of a modern business portal.

In building a portal business case, various contributors to return on investment (ROI) can be listed in five main categories:

- *Cost savings*
- *Revenue generation*
- *Operational efficiency*
- *User satisfaction*
- *Strategy and transformation*

This paper focuses on the cost savings benefit. By investigating the time needed to develop and maintain portal features in both the *build* and the *buy* scenarios, this white paper demonstrates that using the WebSphere Portal platform can help significantly lower costs when compared to using a custom-built portal or creating Web applications using traditional Web development tools without WebSphere Portal.

The core functionality of WebSphere Portal

The core functionality of WebSphere Portal software is the foundation for the portal itself. It provides a wide range of services to connect people to the information they need. These services include the ability to support single sign-on, profiling, content management, personalization, messaging and administrative capabilities. This base functionality is where most customers see the largest ROI, because custom coding these services would normally take a significant amount of time. A commercial development tool, such as IBM VisualAge® for Java or the IBM WebSphere Studio Application Developer toolkit and API is an efficient way to reproduce these services in a custom-built portal application.

Because not all customers use the complete range of functionality, this white paper focuses on a few of the most common services. Figure 1 shows the WebSphere Portal architecture, with blue areas depicting core services.

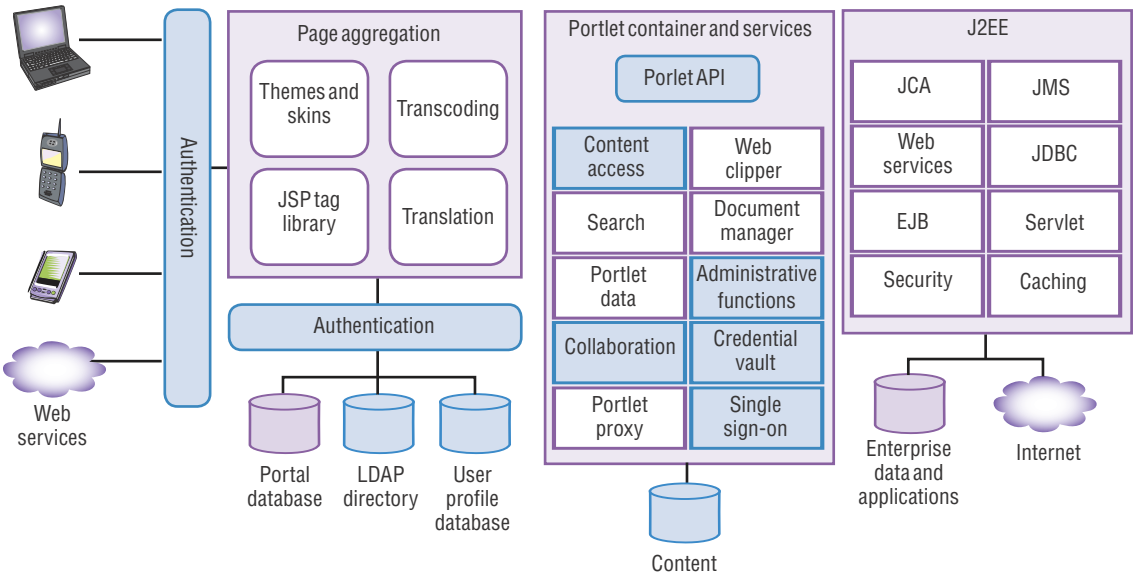

Figure 1. WebSphere Portal architecture

Single sign-on capabilities

One of the main objectives in building a portal is to enable application integration, including nearly seamless authentication with back-end systems. This authentication synchronization, called single sign-on is one of the most important capabilities of a portal. Applications that are exposed through a portal must submit credentials (for example, a user ID and a password) to the back-end remote programs; otherwise, the user is forced to enter this information for each application encountered through the portal.

Buy: Single sign-on with WebSphere Portal

If the remote applications use the same credentials as those used by WebSphere Portal, the portlets can reuse the same credentials as well. Portlets can use the Java Authentication and Authorization Service (JAAS) API to extract these credentials, and submit them to remote applications.

However, it is not always possible for a remote application to use the same credentials as those used by WebSphere Portal. Therefore, to enable a single sign-on experience for the user, WebSphere Portal provides a credential vault mechanism that portlets can use to set and retrieve credentials securely.[2] You spend no development time to develop this credential vault mechanism.

Build: Single sign-on with a custom portal application

The effort to duplicate the functionality of the credential vault functionality has been estimated for a custom portal application. The tasks and their associated times are listed in Table 1.

Task	Time (hours)
Plan and build data table for credential vault	16
Build credential vault core capability	440
Build credential vault interface	220
Test and document	65
Total	**741**

Table 1. The tasks and associated times required to duplicate the credential vault functionality for a custom portal application.

Profiling

In a profiling portal, users can specify attributes about themselves to allow portal content to be targeted to their needs and interests (also known as *personalization*).

Buy: Profiling with WebSphere Portal

WebSphere Portal includes a base profile system that enables users to self-register and identify information about themselves, such as name, e-mail, address, interests and other attributes that personalize their individual experiences. The profile system might be extended by the portal administrator to include as many elements as needed in the profile database. Alternatively, the profile can be extended by the developer, by editing an XML file to map the desired attributes to fields in a Lightweight Directory Access Protocol (LDAP) directory.

One customer, a government entity (See "Customer case study 2" in Appendix 1), extended the profile to two pages to gather more attributes to personalize the user's experience. The information was stored in WebSphere Portal internal databases. The times taken to complete the tasks are listed in Table 2.

Task	Time (hours)
Extend registration page to two pages and add additional fields	25
Build confirmation page	12
Support for error-checking	7
Extra modifications for self-care page	10
Test and document	12
Total	**66**

Table 2. The tasks and associated times required to build a base profile system using WebSphere Portal.

Build: Profiling with a custom portal application

Prior to using WebSphere Portal, the IBM portal (see "Customer case study 3" in Appendix 1), included a registration page to gather information about the user to determine which content to display. The profile page is shown in Figure 3.

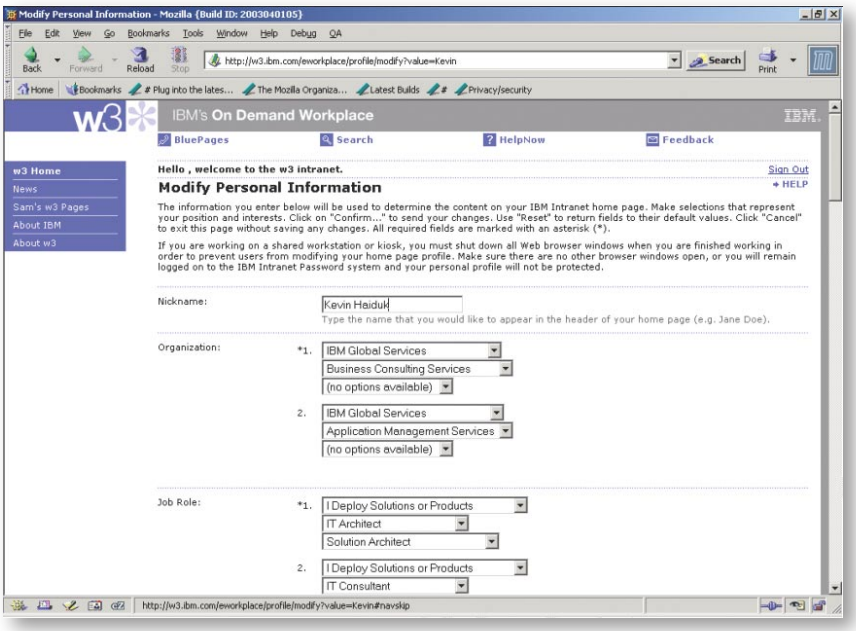

Figure 3. Example of a customized portal profile page

The amount of information gathered in this scenario is similar to the modified profile database in the *buy* scenario in Figure 3 (differences are outlined in Appendix 2). The profile page built in this scenario consisted of the following tasks listed in Table 3.

Task	Time (hours)
Plan and build data table in IBM DB2®	5
Build profile page	230
Build confirmation page	128
Tie into main rules engine	230
Test and document	24
Total	**617**

Table 3. The tasks and associated times required to build a profile page.

Content management

Pertinent, accurate and timely content is key to the value realized by portal users, and content can be targeted to specific users by the inclusion of metatags. This section compares IBM WebSphere Portal Content Publisher (included in IBM WebSphere Portal Extend and IBM WebSphere Portal Experience) to custom content-management applications.

Buy: WebSphere Portal Content Publisher

WebSphere Portal Extend and WebSphere Portal Experience include a content-management tool called WebSphere Portal Content Publisher. This tool closely integrates with WebSphere Portal, allowing users to contribute and approve content to the portal through an integrated workflow system.

Through the WebSphere Portal Content Publisher User and Content models, developers can create content templates, based on publication rules and workflow. Portal content can then be submitted and approved through a portlet that exposes these templates. Figure 4 illustrates a sample of a simple content template.

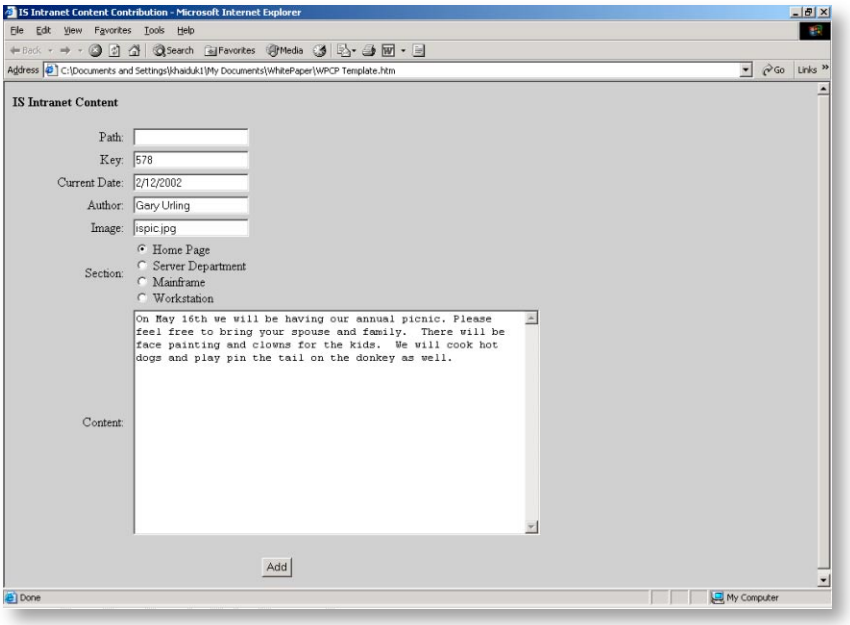

Figure 4. The WebSphere Portal Content Publisher input template

During a recent customer project (see "Customer case study 2" in Appendix 1), WebSphere Portal Content Publisher was used to contribute content to a state government's intranet. The content passed through a simple workflow consisting of a requestor, a contributor and an approver. This was performed for three workflow cases. The times taken to accomplish the tasks to build this setup with WebSphere Portal Content Publisher are listed in Table 4.

Task	Time (hours)
Plan and build data tables in DB2	3
Build user resources from tables	16
Build content resources from tables	16
Build templates from resources	60
Test and document	8
Total	**103**

Table 4. The tasks and associated times required to contribute content to an intranet through WebSphere Portal.

More robust content management tools are available that could potentially cut this time even more. These tools offer greater benefits to the content creators and approvers by providing additional functionality. These tools, which include IBM Lotus® Workplace Web Content Management and third-party tools from Interwoven, Inc., Stellent, Inc. and Fatwire, are preintegrated into WebSphere Portal and, therefore, can be added easily.

Build: Content management with a custom portal application

Within a custom portal application, developers might still opt for off-the-shelf content-management software, or they might choose to develop their own. This white paper examines IBM's own content-management system prior to WebSphere Portal. The content-management system supported only a small subset of the functionality that is now available in WebSphere Portal Content Publisher.

IBM developers based this proprietary content-management system on IBM Lotus Notes® software. The system allowed end users around the world to add content to the portal. The content was published on an hourly basis to the portal, and an on demand publishing agent was also available. The content was published as an XML file, and tagged with metadata that allowed it to be targeted to the correct audience. The tasks involved to build the application and the associated times are shown in Table 5.

Task	Time (hours)
Build base Notes database	190
Build support for seven different portlets	428
Build support for meta-tagging	140
Build publishing agent	76
Test and document	52
Total	**886**

Table 5. The tasks and associated times required to build a content-management application.

These estimates assume the use of Lotus Notes as a tool. Without this application platform, which eases development of templates, workflow and versioning, the time to create a content-management system truly from the ground up could be much higher. The key differences between this custom-developed content-management approach and WebSphere Portal Content Publisher can be found in Appendix 2.

Personalization

Personalization, one of the major strengths of a portal platform, works with profiling, to allow end users to receive relevant content aimed at personal attributes, as specified in their user profiles. The user profile specifies the attributes (such as job, title, geographic location and so on) that the personalization service uses to direct relevant content.

Buy: Personalization with WebSphere Portal

The services available in WebSphere Portal allow personalization to be abstracted to the appropriate business level. Administrators and developers can change the personalization rules with configuration changes and without writing code. Figure 5 is an example of how a developer might configure a rule.

Figure 5. Configuring a personalization rule.

A customer (see "Customer case study 1" in Appendix 1) personalized three portlets using what is known as a *spot*, which uses the rules in portlets to display content. The times to implement these portlets are listed in Table 6.

Task	Time (hours)
Build base portlets	34
Add spot to portlets	6
Build rules	1
Test and document	4
Total	**45**

Table 6. The tasks and associated times required to implement personalization portlets.

Build: Personalization with a custom portal application

In another customer study, the developers created a custom content-management system to tag content with metadata. The portlets were written to include Structured Query Language (SQL) calls to pull the pertinent data. (An inherent problem with this approach is that code must be changed whenever rules change. See Appendix 2 for key differences in this custom approach compared to the WebSphere Portal approach.) The tasks and their associated times are listed in Table 7.

Task	Time (hours)
Build base portlets	18
Add logic and rules to portlets	88
Test and document	14
Total	**120**

Table 7. The tasks and associated times required to build a custom content-management system.

Messaging between portlets

In a portal application it is often necessary for one portlet to share data with another portlet. For instance, one portlet might show a customer's list of accounts with some contact data, and another portlet might show account balance data for a specific account. Rather than always keying in the account number to the second portlet, it can be more efficient to automatically send

the account number to the second portlet, and refresh it with the current data for that customer whenever an action is taken in the first portlet. The user can then click the icon for an account, and the second portlet shows the balance. This capability is called *portlet messaging*.

Buy: Messaging between portlets with WebSphere Portal

Messaging between portlets in WebSphere Portal is accomplished through a mechanism called *click to action*. With click to action, developers can easily place messaging capabilities in portlets. To use the click-to-action feature, developers must declare it in the portlet descriptor and add a few lines of code in the portlet JavaServer Pages (JSP). Figure 6 shows an example of a code snippet to show a received message.

```
<td>
   <C2A:encodeProperty namespace="http://hostname/c2a"
         type="UserIdType" value="<%= object.getUserId() %>" />
   <%= object.getUserId() %>
</td>
```

Figure 6. A portlet messaging code fragment.

The account balance portlet example described before was used to determine development time for this effort. Actual back-end systems were not integrated; a simple call to a database was used. The tasks and their associated times are shown in Table 8.

Task	Time (hours)
Build base portlets	22
Add click to action capabilities to portlets	1
Test and document	10
Total	**33**

Table 8. The tasks and associated times required to build an account balance portlet.

Bowstreet Portlet Factory, a third-party tool, also supports portlet-to-portlet communication through the WebSphere Portal messaging API (click to action, see Figure 7). This is accomplished by first adding a WebSphere Portal event-declaration builder to each model.

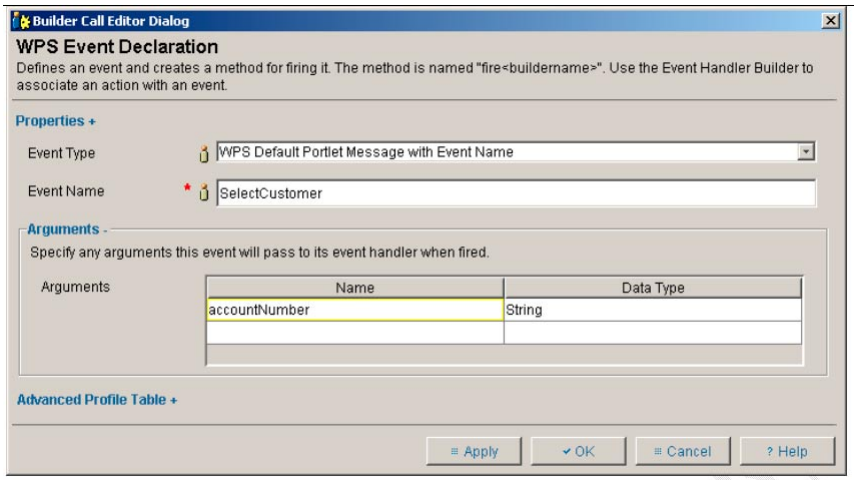

Figure 7. Bowstreet approach to a click-to-action portlet.

The portlet that handles the event also requires an event-handler builder to specify an action to be called when a message was received. The portlet that sends the message uses a link builder to trigger the event and pass the customer number as an argument.

Build: Messaging between portlets with a custom portal application

As a result of the open, expandable portlet functionality and API in WebSphere Portal, portlets are separated into specific applications and do not share the same session. In custom portal applications, on the other hand, all portlets normally reside in the same application, and so they all share the same session. Messaging between portlets can be as simple as putting an object in the session for another portlet to retrieve; thus, methods to transmit messages between portlets in custom portal applications might at first seem quite easy to develop. However, complications can arise from naming when the same portlet is placed on a single page multiple times, which happens often.

For the purposes of this paper, the account balance scenario described is built in a custom portal application, adding the support for messaging to the individual portlets, thereby creating a one-off situation in each portlet. If this were an actual portal, best practices would be followed and the messaging functionality would probably have been added to the base portal itself, an effort estimated at over 500 hours. Again, actual back-end systems were not integrated; a simple call to a database was used.

Task	Time (hours)
Build base portlets (noninclusive)	(20)
Add additional base infrastructure support for portlet messaging	20
Add send and receive messaging capabilities to portlets	6
Test and document	8
Total	**34**

Table 9. The tasks and associated times required to build a custom portal application.

Administrative functionality

A well-structured portal enables required administrative tasks, such as user administration, rules administration, security administration and presentation administration to be performed inside the software.

Buy: Administrative functionality in WebSphere Portal

WebSphere Portal uses its own portlets to deliver administrative functions. Therefore, there is virtually no development effort involved to support this function in WebSphere Portal, unless customizations are necessary. Figure 8 shows one such administrative function, the portlet with which the administrator applies security to pages and portlets.

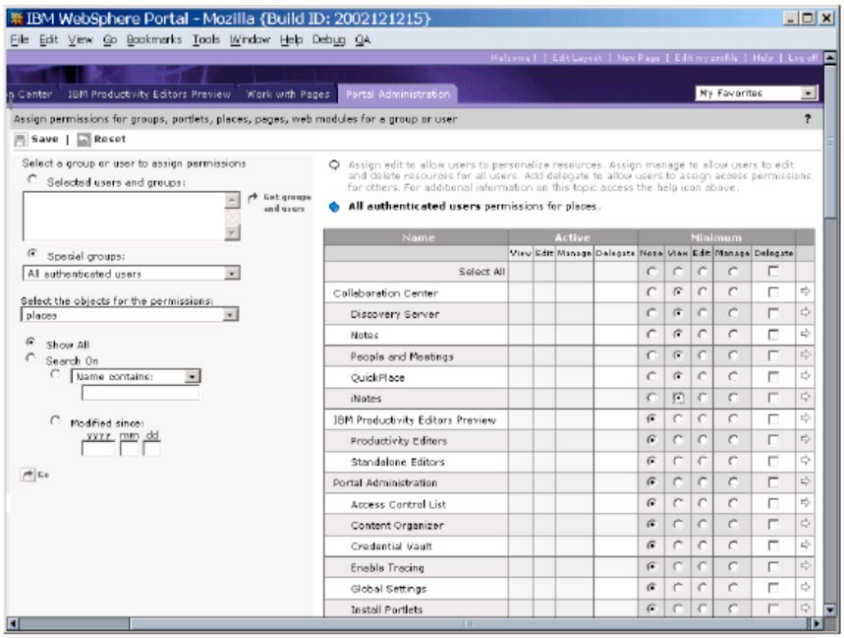

Figure 8. Administrating portal security.

Build: Administrative functionality in a custom portal application

For comparison, this white paper examines the case of delivering the same security in a portal as shown above. One customer managed portal security in a simple XML file. To change security, the XML file had to be updated manually and the portal restarted to read in the changes. (See Appendix 2 to view the key differences in this functionality compared to the functionality that is delivered in WebSphere Portal software.)

Task	Time (hours)
Administrating a portlet[3]	250
Administrating a user	180
Administrating security[4]	110
Test and document	44
Total	**584**

Table 10. The tasks and associated times required to provide administrative functionality.

Core functionality summary

After exploring the main categories of core functionality involved with a portal, the times needed to build the functionality highlighted in this white paper are summarized in Table 11.

Functionality	Time (custom portal)	Time (WebSphere Portal)	Time savings
Single sign-on	741	0	741
Profiling	617	66	551
Content management	886	103	783
Personalization	120	45	75
Administrative functionality	584	0	584
Portlet messaging	34	33	1
Total	**2982**	**247**	**2735**

Table 11. A summary of the times needed to build core portal functionality.

Building portlets in the portal

WebSphere Portal develops the window (through which the actual content or application is seen) in portlets, which are small portal applications, usually depicted as small boxes in a Web page. Portlets are reusable components that provide access to applications, Web-based content and other resources. Web pages, Web services, applications and syndicated content feeds can all be accessed through portlets.

Portlets can be very simple or very complex, and can connect to back-end systems. This white paper examines the full spectrum—from basic functionality portlets, to complex application integration and productivity portlets. For comparison, assume that the custom-built base portal is exactly equivalent to WebSphere Portal Server.

There are several different approaches to building portlets, each appropriate in a different scenario. Where applicable, this paper looks at as many as four different approaches: using ready-made portlets from the IBM WebSphere Portlet Catalog[7]; development of portlets on IBM WebSphere Portal Server with IBM WebSphere Studio Application Developer; development of portlets using Bowstreet Portlet Factory; and by raw code development.

Basic functionality portlets

This example compares the deployment of two different simple portlets from the portlet catalog – the Links portlet and the Web clipping portlet – with the functionally similar, custom-built portlets that were used internally in IBM prior to the release of WebSphere Portal software.

The function of the Links portlet (see Figure 9) is to show a set of links based on user profiles, and to allow the users to further customize the links themselves.

Figure 9. Essential Links portlet.

Web clipping allows portal administrators to build portlets that consume external content and filter it to present a view of only the relevant portions of that content. Examples include weather feeds and stock feeds.

Buy: Basic functionality portlet with WebSphere Portal using available portlets

Companies using WebSphere Portal can create their own portlets or select from a catalog that includes hundreds of portlets created by IBM and IBM Business Partners. Many basic functionality portlets are included with WebSphere Portal and are regularly updated and available through the Portal Catalog. In the cases of both the Links portlet and the Web-clipping portlet, no development is involved because they are provided with WebSphere Portal; administrators simply deploy the portlet to users. The only tasks required are deploying and configuring the portlet, and the total time to do either is approximately 30 minutes, under normal circumstances.

Build: Basic functionality portlet with a custom portal application.
Writing the code required for the Links portlet involves accessing various business objects and user profiles. The tasks and the associated time to complete this portlet are shown in Table 12.

Task	Time (hours)
Build portlet	15
Build edit page	35
Build core logic	30
Test and document	18
Total	**98**

Table 12. The tasks and associated times required to build a custom portlet.

Building the Web-clipping portlet functionality is a more complicated task, because it must navigate all security functions, such as cookies, proxies, firewalls and so on.

Internally, IBM deployed a portlet that supported a subset of a Web-clipping portlet. The portlet had the ability to clip part of a page with the limitations of not being able to specify a clip on the fly, no proxy support or support for cookies. The tasks and the associated times are shown in Table 13.

Task	Time (hours)
Build portlet (display and edit)	50
Add additional support for functionality in Web-clipping portlet	298
Test and document portlet	40
Total	**388**

Table 13. The tasks and associated times required to build a custom portlet.

Application integration portlet

The case of a customer relationship management (CRM) portlet is a good example of a midlevel portlet that connects to a back-end enterprise system. The example for this case is that of connecting to PeopleSoft doftware to obtain a customer contact list.

Buy: WebSphere Portal CRM portlet using available portlets

WebSphere Portal has many out-of-the-box CRM portlets for several leading CRM vendors. For this case, by using the PeopleSoft CRM suite of portlets, there is no development needed for these portlets beyond simple configuration and deployment. Figure 10 shows an example of a user's PeopleSoft contact list.

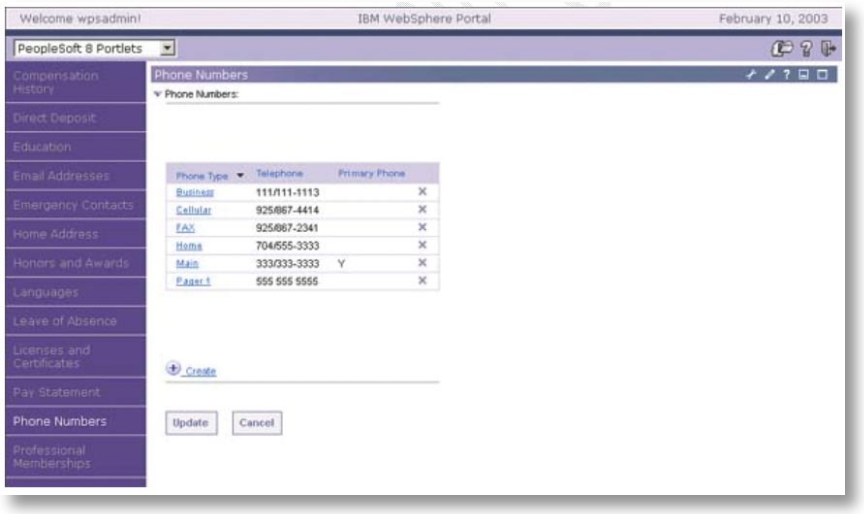

Figure 10. PeopleSoft CRM portlet.

The tasks and times associated to configure and deploy the portlet are shown in Table 14.

Task	Time (hours)
Configure portlet	0.5
Deploy portlet to users	0.2
Total	**0.7**

Table 14. The tasks and associated times required to configure and deploy a custom portlet.

Buy: Creating a WebSphere Portal CRM portlet using WebSphere Studio Application Developer

Not all versions of all CRMs are supported by the portlets in the catalog. Customers using not supported products and versions must build their own portlets. The WebSphere Portal platform ships with WebSphere Studio Application Developer and a portal toolkit. Together, they allow developers to easily build, modify and maintain portlets by using a portlet wizard to quickly create the foundations for various portlets.

Building the CRM portlet with WebSphere Studio Application Developer involves several tasks: creating an edit page (so that an administrator can configure the display of the data), the actual display page itself, and a Simple Object Access Protocol (SOAP) call to the back-end system.

This approach involves several calls using PeopleSoft EIP Web services to obtain the needed information through the PeopleSoft enterprise integration points feature.[6]

The tasks and their associated times to build this portlet from the ground up are shown in Table 15.

Task	Time (hours)
Build base portlet	10
Build portlet edit page	14
Integrate appropriate Web services calls	15.5
Test and document portlet	18
Total	**57.5**

Table 15. The tasks and associated times required to build a custom portlet.

Buy: Creating a WebSphere Portal CRM portlet using Bowstreet Portlet Factory

Customers can also use Bowstreet Portlet Factory for WebSphere to augment WebSphere Portal with tools and technology that are used to rapidly create, customize, deploy and maintain portlets (see Figure 11). By pulling together a sequence of highly adaptive, reusable software components called *Builders*,

novice Java developers can assemble these Builders into models, similar to the way they would build a spreadsheet model by snapping together formulas. These models are then started at run time to dynamically generate application code, including JSP, Java classes and XML documents, as well as all of the low-level artifacts that make up the portlet application.[7] This way, developers can quickly and easily create multiple, highly customized portlets from one code base, without requiring additional code changes or redeployment.

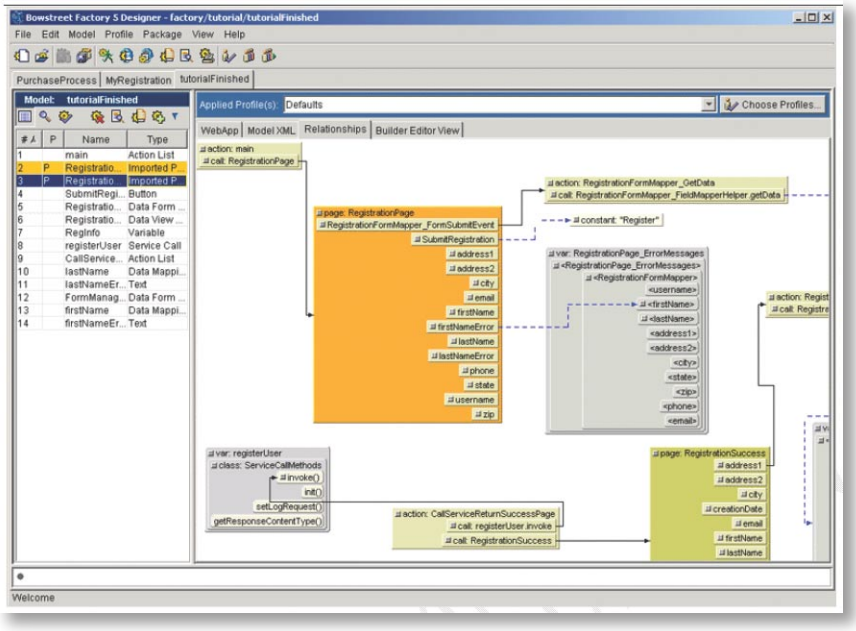

Figure 11. Bowstreet Portlet Factory.

The CRM portlet is straightforward, using the Bowstreet tool. Developing this portlet involves using the Data View Portlet Builder to call the SOAP Web service, and to generate the JSP for presentation in the portlet.

The Portlet Customizer Builder was used to create an edit page that allows the end user to personalize the columns that appear in the table and the number of rows to see in the table.

The tasks and their associated times are shown in Table 16.

Task	Time (hours)
Create portlet using Data View Portlet Builder	0.5
Build edit page	0.5
Test and document portlet	15
Total	**16**

Table 16. The tasks and associated times required to create an edit page.

Build: Creating a WebSphere Portal CRM

Unlike the scenario before, if the portal is custom-built, few or no tools are available for developers. In this case, all portlet code must be written. The tasks and their associated times to complete the portlet are shown in Table 17.

Task	Time (hours)
Build base portlet	40
Build portlet edit page	44
Integrate appropriate Web services calls	25.5
Test and document portlet	20
Total	**129.5**

Table 17. The tasks and associated times required to build a custom portlet.

Appendix 2 describes key differences in the capabilities of the custom portal when compared to those available in WebSphere Portal.

Productivity portlet

Productivity portlets normally support desktop applications, such as IBM Lotus Notes or Microsoft® Exchange in a portal environment. This can be very useful and can help realize the portal as the desktop. It can also be helpful as traveling employees access their e-mail and calendars through kiosks. People awareness, or the ability to know who is currently available, using online methods, (also referred to as *contextual collaboration*), is an essential feature of a portal. One of the leading instant messaging tools is Lotus Instant Messaging. The ability to tie this capability with a portal adds high value.

Buy: Productivity portlet with WebSphere Portal using available portlets
Included in the many productivity portlets shipped with WebSphere Portal is the Lotus Notes Mail portlet (see Figure 12). There is virtually no development necessary to deploy this portlet, and little time (0.3 hours total) to configure and deploy it. If you also deploy Lotus Instant Messaging, the mail portlet is preconfigured to show people awareness.

Figure 12. Lotus Notes e-mail portlet

Buy: Creating a productivity portlet with WebSphere Portal
The effort required to create the Notes Mail portlet is estimated by using the time it took to build a portlet for a view onto a Notes database, abstracting only certain information and building a custom input form.

In WebSphere Portal, adding people awareness to any portlet is straight-forward. WebSphere Portal Extend and WebSphere Portal Experience ship with Lotus Instant Messaging, Lotus Team Workplace and Lotus Collaborative Components. These components, also known as *collaborative services (CS)*, provide Java API methods and tags for JSP. Application developers can use collaborative components to design and implement portlets that incorporate the features of IBM Lotus Domino®, IBM Lotus QuickPlace® and IBM Lotus Sametime® software. Figure 13 shows the ease of use in adding people awareness to a portlet.

```
<%@ page import="com.lotus.cs.*" %>
<%@ taglib uri="/WEB-INF/tld/people.tld" prefix="sametime" %>
<%@ taglib uri="/WEB-INF/tld/menu.tld" prefix="menu" %>

<table border="1" cellpadding="1">
<tr><th>User Name</th></tr>
<tr><td><sametime:person>Kevin Haiduk</sametime:person></td></tr>
<tr><td><sametime:person>Roger Doerman</sametime:person></td></tr>
<tr><td><sametime:person>Thomas Degeest</sametime:person></td></tr>
<tr><td><sametime:person>Gary Urling</sametime:person></td></tr>
<tr><td><sametime:person>Joey Bernal</sametime:person></td></tr>
```

Figure 13. Adding people awareness capabilities to a portlet

Table 18 shows the tasks and their associated times.

Task	Time (hours)
Build base portlet with view of Notes data	22
Build customized input form	24
Build core functionality	22
Test and document	4
Total	**72**

Table 18. The tasks and associated times required to build a custom portlet.

Buy: Creating a productivity portlet with Bowstreet Portlet Factory

Building a portlet to access an existing Lotus Notes database is quick and easy with Bowstreet Portlet Factory. A new portlet is created to access any Lotus Notes database by configuring the Domino View & Form Builder. The Builder generates a page displaying the View data, and also pages to view, edit, create and delete documents in the View (see Figure 14). The same amount of work must be done to implement people awareness as before. All of these pages are rendered within the portlet, without launching IBM Lotus Domino.

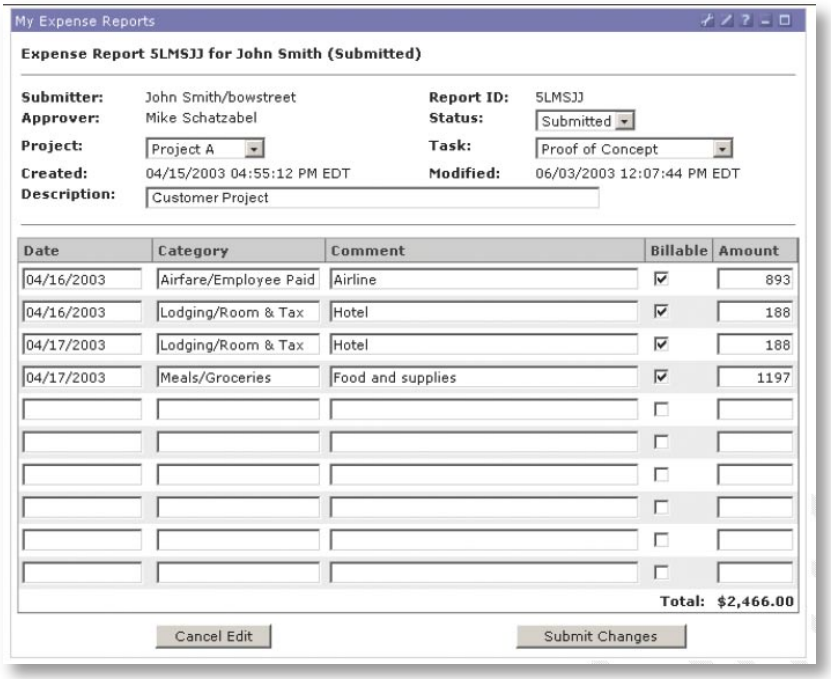

Figure 14. Building a Domino View portlet

These pages can then be customized to create a custom portlet that matches the look and feel of the portal. Table 19 shows the tasks and the associated times.

Task	Time (Hours)
Build base portlet with view and forms	0.1
Customize HTML for document view, edit, and create pages	1
Test and document	3
Total	**4.1**

Table 19. The tasks and associated times required to customize a portlet.

Build: Creating a productivity portlet with a custom portal application

Implementing a Lotus Notes e-mail portlet in a custom portal application involves working directly with the Lotus Notes API through Domino Internet Inter-ORB Protocol (DIIOP). This type of implementation involves a medium amount of coding. Without Lotus Instant Messaging, which is bundled with some versions of WebSphere Portal, achieving people awareness in this portlet would be a formidable task.

Because Lotus Instant Messaging is easy to integrate with other programs, this example includes the time it takes to integrate Lotus Instant Messaging into this custom application.

Note: This does not comply with the previous rules regarding custom development.

Table 20 shows the tasks and the associated times needed to code a portlet supporting Lotus Notes e-mail in a custom portal application.

Task	Time (hours)
Build portlet display	80
Build portlet edit page	82
Build core logic	154
Test and document portlet	12
Total	**328**

Table 20. The tasks and associated times required to code a portlet.

Custom portlet application: A simple example

In most portal applications, there are no out-of-the box portlets to solve certain problems. Inevitably, portlets must be built from the ground up. In a recent customer project, a county government (see "Customer case study 2" in Appendix 1) deployed several custom-built portlets that read abstract content from a database and displayed it according to rules established on the edit pages.

This section examines this customer's news portlet. The news portlet reads the title and abstract of news articles from an IBM DB2® database, which then obeys the rules established on the edit page to determine the display, filtering and sorting rules. Figure 15 shows the news portlet as seen by a user, and Figure 16 shows the actual edit page of the portlet available to administrators.

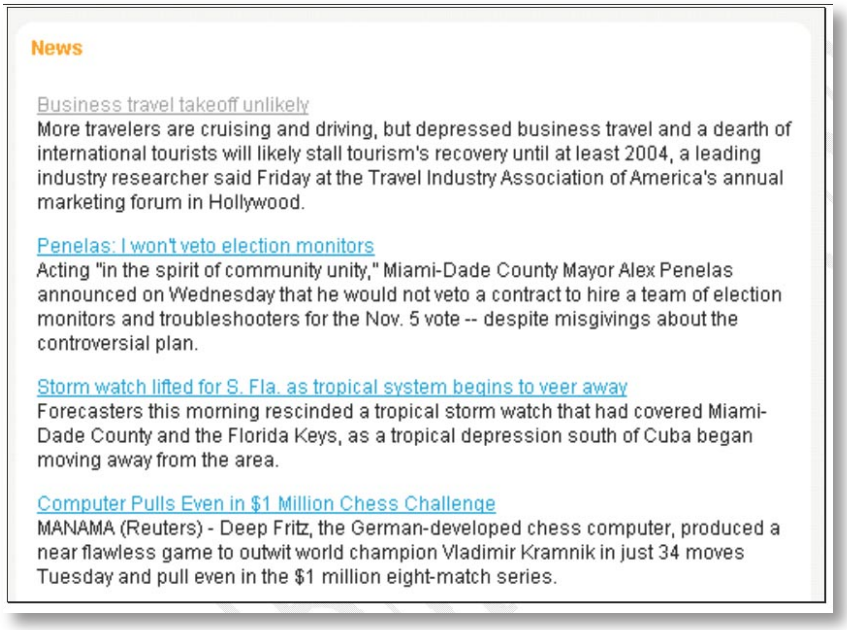

Figure 15. News portlet seen by users.

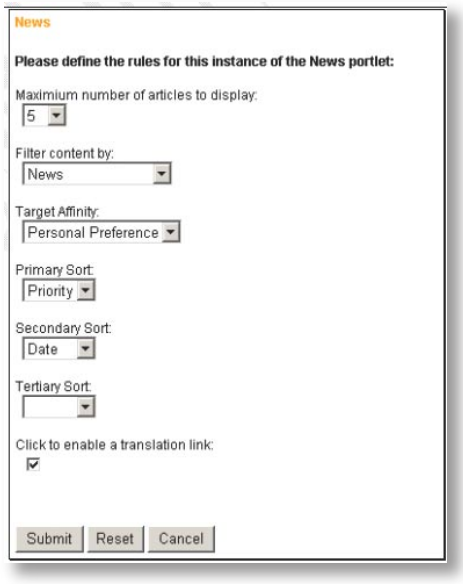

Figure 16. Edit page of news portal available to administrators.

Administrators can configure the portlet using the following parameters:

- *Number of articles to display (1 to 25)*
- *The type of content (10 different types of content)*
- *Affinity (business, employee, visitor, resident or, if the user is logged in, which of these four affinities the user chose in the user profile, denoted as Personal Preference)*
- *Three levels of sorting*
- *Ability to show access translation services (using IBM WebSphere Translation Server)*

The edit page shows one of the major strengths of the portlet. It is easily configurable and can be used on many pages of the portal using the same code base. The portlet was simply placed on many pages throughout the portal and configured for the page instance.

Buy: Simple custom portlet with WebSphere Portal

This portlet was built in WebSphere Studio Application Developer software using the portal toolkit plug-in. Data read in from the outside database was automatically saved to the portal databases with simple API calls, so that it could be placed on any of the pages, using configuration parameters to retrieve the appropriate news stories.

See Table 21 for the tasks and the associated times to complete this portlet.

Task	Time (hours)
Build portlet edit page	4
Build logic and display page	10
Build action listeners	8
Test and document the portlet	10
Total	**32**

Table 21. The tasks and associated times required to build a custom portlet.

Buy: Simple custom portlet with Bowstreet Portlet Factory

To build this portlet in Bowstreet Portlet Factory, a SQL call builder was used to select headlines from the database. The query's result set data was mapped to a tabular presentation using a data page builder. The statement parameters, such as the news category and sort-by information, were exposed for customization in the portlet's edit screen. This edit panel was built using a portlet customizer builder to provide a custom user interface, including drop-down lists with appropriate choices.

The tasks and their associated times are shown in Table 22.

Task	Time (hours)
Build news portlet	0.5
Create custom edit page	0.5
Test and document	8
Total	**9**

Table 22. The tasks and associated times required to build a custom portlet.

Build: Simple custom portlet with a custom portal application

Building the custom news portlet in a custom portal doesn't require much more work than the portal with WebSphere Portal. The major disadvantage of not using WebSphere Portal is that data stored on the edit page in this case is not automatically saved to the portal databases, so the developers must be concerned with mapping the data of a particular portlet instance to a specific table, while keeping track of the portlet instance. For example, this portlet is on the business page so it is identified as such, and it does not read configuration data from the same portlet on the employee page). The tasks and their associated times are shown in Table 23.

Task	Time (hours)
Plan and built database tables	4
Build portlet edit page	33
Build logic and display page	49
Test and document the portlet	18
Total	**104**

Table 23. The tasks and associated times required to build a custom portlet.

Custom portlet: A complex example

This section examines the effort required to build a more complex portlet—pulling information from many disparate systems and aggregating it to a single view. What seems to be a simple, convenient view of information to the end user is, in fact, data from several systems gathered using different protocols from different locations.

In this scenario, the portlet (see Figure 17) displays information to a sales manager, who sees data regarding employees and their related sales. The portlet shows several pieces of key information, aggregated from many different systems by the portlet. The manager can click the table cells to see more detailed information. The data in the columns can also be sorted by clicking the column header.

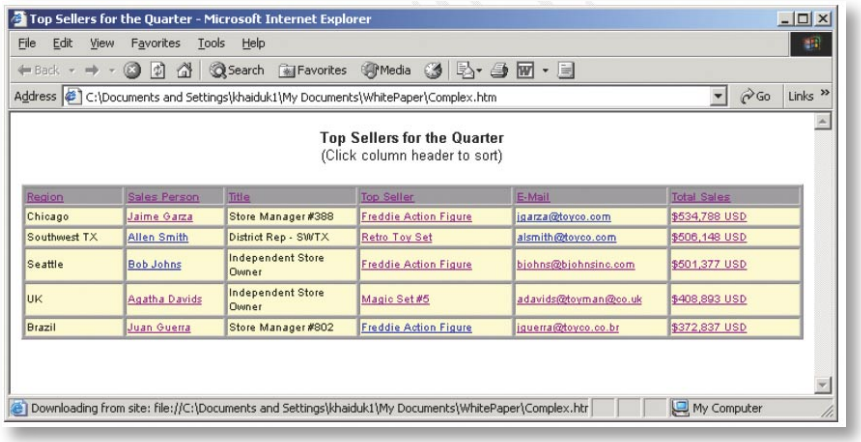

Figure 17. Complex portlet

Table 24 shows the data and the respective originating sources.

Data element	System	Transport
Region	HR system	IBM MQSeries®
Sales professional	LDAP	LDAP
Title	LDAP	LDAP
Top seller	Inventory system	Web service
E-mail	LDAP	LDAP
Total sales	CRM System	JDBC

Table 24. The data and the respective originating sources

Figure 18 shows a logical view of the systems and the respective protocols.

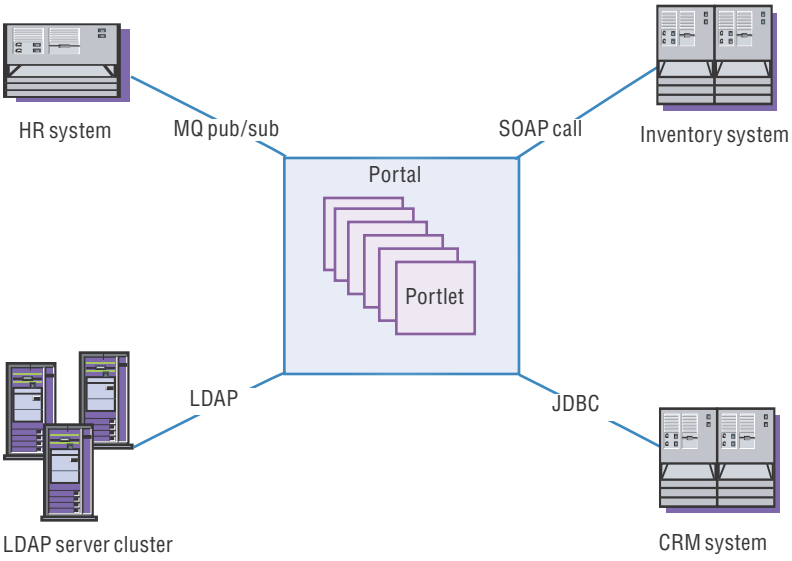

Figure 18. Logical view of systems involved in a complex portlet

Buy: complex custom portlet with WebSphere Portal

Development of this type of complex portlet is normally hindered by the reliance on the disparate subsystems and timely responses, requiring complex caching mechanisms to be written. However, the WebSphere Portal platform enables developers to more efficiently build parts of the portlet by taking advantage of the existing infrastructure.

The tasks involved and their corresponding efforts are shown in Table 25.

Task	Time (hours)
Build base portlet (including sorting action)	55
Create MQ calls	22
Create LDAP calls	8
Create SOAP calls	18
JDBC access calls	6
Aggregation and sorting logic	15
Caching techniques and handling timeouts	65
Testing and documentation	15
Total	**204**

Table 25. The tasks and associated times required to build a custom portlet.

Buy: Complex custom portlet with Bowstreet Portlet Factory

Building this portlet on WebSphere Portal through Bowstreet Portlet Factory requires a similar design to access data in each back-end system. However, Bowstreet Portlet Factory provides a framework for sorting, paging through data and caching, which results in less design work and coding for the developer. Additionally, creating the base portlet and user interface is considerably faster as a result of builders such as Data Page, which can automatically generate a presentation and formatting for the displaying and editing with the data.

Task	Time (hours)
Build base portlet (including sorting action)	10
Create MQ calls	22
Create LDAP calls	8
Create SOAP calls	18
JDBC access calls	6
Aggregation and sorting logic	9
Caching techniques and handling timeouts	45
Testing and documentation	15
Total	**133**

Table 26. The tasks and associated times required to build a custom portlet.

Build: A complex custom portlet with a custom portal application

Management of the portal's cache requires specialized skills and experience. This is compounded in a portal application where several portlets on one page might be making Web service calls and knowledge of different caching techniques is required. Handling these techniques adds to the overall development effort. The tasks and their associated times to develop the complex portlet in an environment other than a WebSphere Portal environment are shown in Table 27.

Task	Time (hours)
Build base portlet (including sorting action)	125
Create MQ calls	26
Create LDAP calls	22
Create SOAP calls	18
JDBC access calls	22
Aggregation / sorting logic	45
Caching techniques / handling timeouts	210
Total	**468**

Table 27. The tasks and associated times required to build a custom portlet.

Portlets summary

Through ready-made catalog portlets, rich development tools and an intelligent infrastructure, the WebSphere Portal platform can provide more efficient portlet development than using a custom portal application. The scenarios and their associated development times described in this section are summarized below.

Portlet	Custom application	WebSphere Portal with WebSphere Studio Application Developer and available portlets	WebSphere Portal with WebSphere Studio Application Developer	WebSphere Portal with Bowstreet Portal Factory	Savings — WebSphere Portal compared to custom portal
Basic functionality	98	0.1	n/a	n/a	**97.9**
Web clipping	288	0.3	n/a	n/a	**387.7**
CRM	129.5	0.7	57.5	16.0	**128.8**
Productivity	328	0.3	72	4.1	**327.7**
Custom portlet - simple	104	n/a	32	9.0	**72.0**
Custom portlet - complex	468	n/a	204	133.0	**264.0**
Total	**1515.5**	**1.4**	**365.5**	**162.1**	**1278.1**

Table 28. A summary of portal development scenarios.

Maintenance comparisons

Maintenance of a portal involves several areas, including maintenance of the base infrastructure, adding new function and upgrading existing function. These areas of the portal can be the most difficult areas for which to quantify an accurate cost-savings realization. This section focuses on adding and maintaining portlet applications in the portal.

Maintaining and adding new portlets

One of the major benefits of WebSphere Portal compared to a custom portal application is the ability to install new portlet applications in a running portal. This can save a vast amount of time in maintenance and zero downtime for users. All code is packaged as Web Application Archive (WAR) files, which saves administrators from deploying multiple code sets (such as Java class files, Java Archive [JAR] files and JSP). In the custom portal application at IBM, it was necessary to update JAR files and JSPs manually, and then restart the application server to read in the packaged code.

Task	Task detail	WebSphere Portal	Custom portal application
Install new and update existing portlet	Add or update portlet	0.1	0.25
Total		**0.1**	**0.25**

Figure 19. Installing or updating a portlet in WebSphere Portal.

Looking at a continuing maintenance and development environment, these seemingly small differences in times quickly add up. The times shown represent the requirements to update ten different portlets per day, in a maintenance environment.

Task	Task detail	WebSphere Portal	Custom portal application
Install new and update existing portlet	Add or update portlet	1	2.5
Total		**1**	**2.5**

Table 29. The tasks and associated times required to update or install 10 portals per day.

Assuming there are roughly 255 working days[8], and maintenance is performed on one-quarter of these days, a time savings of roughly 96 hours per person is possible[9]. If the average development and maintenance team for an enterprise portal consists of on average four people, the estimated yearly savings is 384 hours.

Ongoing portlet development

In all enterprise-portal applications, new functions are continually added to evolve the portal. For example, in a business-to-employee (B2E) portal application, new lines of business can add their own specific functionality to the portal. Depending on the situation, additional portlet development might be compounded proportionally to the growth of the portal. Based on customer case studies and other experience, the number of portlets added for an average enterprise-portal deployment over a one-year period is estimated in Table 30; they are categorized in the portlet scenarios listed in this section.

Portlet	Number of portlets deployed per year	Estimated hourly savings in hours	Total hourly savings in hours (rounded)
Miscellaneous yearly maintenance savings			**384**
Basic functionality	3	47.9	144
Web clipping	4	197.7	791
CRM (application integration)	3	78.8	236
Productivity portlet	2	73.7	147
Portlet messaging	2	21	42
Custom portlet—simple	4	31	124
Custom portlet—complex	3	60	180
Total portlet development hourly savings over a one-year period			**2048**

Table 30. Savings using WebSphere Portal available portlets and coded portlets versus a custom portal application for one-year maintenance period.

Conclusion

Although it is difficult to quantify all of the efforts involved in different development approaches, this white paper compared each scenario using cumulative expertise and experience. When portal decision makers are ready to decide whether to invest in a commercial off-the-shelf portal infrastructure, or to build their own, they must weigh many factors, including development efforts, the pros and cons of each, their own development and deployment environments, the skills and experiences of their development teams, standards in place at their companies and their unique business requirements.

The analysis shows a dramatic time and cost reduction when using the WebSphere Portal platform compared to a custom portal application. This analysis represents an opportunity for lower total cost of ownership (TCO), as well as an opportunity for faster time to value and quicker adaptation of the portal to changing business needs.

The comparisons described in this paper are used to estimate the value of a portal, and to develop and deploy 30 portlets of various types that are typical of enterprise-portal deployments. In this example, detailed in the conclusion of the paper, the savings are dramatic – an estimated five-year savings of US$1.4 million for the portal project that uses WebSphere Portal, compared to building a portal using custom Java 2 Platform, Enterprise Edition (J2EE) technology.

These savings result from the consideration of development time alone and do not include required components of an IT project, such as project management, requirements definition, training and testing. Nor do the savings reflect any incremental value of additional portal function potentially provided in the WebSphere Portal solution, or the value of faster implementation.

Considerations for each scenario

Build: Custom portal approach

+	Allows maximum flexibility for the portal environment and the freedom to build the portlet API and portal framework.
–	Is difficult to maintain and upgrade; IT staff supporting proprietary system may require more of a learning curve for new resources
–	Requires IT to custom-build all portlets because of proprietary API and architecture; no public portlets available
–	Lacks larger technical support community and best practices from other implementation experiences

Buy: Using WebSphere Portal and the Portal Toolkit

+	Delivers reliability and scalability as an enterprise application
+	Offers quick deployment and configuration
+	Provides an open architecture and API; portlets containing a plethora of functionality are available and regularly maintained
+	Delivers excellent support for communities, a function that is hard to duplicate in a proprietary environment
–	Requires reliance on a set API and architecture that might create more work when building unsupported capabilities

Buy: Using a third-party tool (Bowstreet Portlet Factory)

+	Allows users that have little Java experience, and little or no understanding of the portal API, to build many custom portlets
+	Provides many out-of-the-box builders that enable end users to put together functionality instead of writing code
+	Allows easy portlet customization for varying constituencies or contexts without the need to code condition logic or copy or paste new code bases
–	Sometimes requires portions of portlet applications to be developed outside of the tool for complex functionality

Table 31. Pros and cons of a custom portal, WebSphere Portal and a third-party approach.

Summary of efforts involved

Functionality	Custom application	WebSphere Portal with WebSphere Studio Application Developer
Single sign-on	741	0
Profiling	617	66
Content management	886	103
Personalization	120	45
Administrative functionality	584	0
Portlet messaging	34	33
Total	**2982**	**247**

Table 32. A summary of portal functionality development efforts.

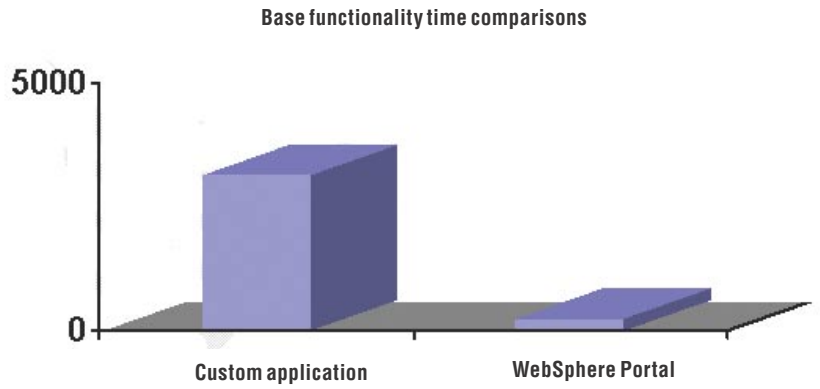

Figure 20. The total time required to build the base infrastructure.

Portlet	Custom application	WebSphere Portal with WebSphere Studio Application Developer and available portlets	WebSphere Portal with WebSphere Studio Application Developer	WebSphere Portal with Bowstreet
Basic functionality	98	0.1	n/a	n/a
Web clipping	388	0.3	n/a	n/a
CRM	129.5	0.7	57.5	16
Productivity	328	0.3	72	4.1
Custom Portlet—simple	104	n/a	32	9
Custom Portlet— complex	468	n/a	204	133
Total	**1515.5**	**1.4**	**152.5**	**162.1**

Table 33. The time to develop the portlets in the scenarios described in this white paper (with best approaches shown in green).

From the data gathered in this study, WebSphere Portal saved 1015.1 hours[10] of time for developing just four portlets over a custom portal application. With just five portlets developed for WebSphere Portal, the Bowstreet Portlet Factory tool saved an additional 205.8 hours. Most portals consist of many more portlets than developed as described in this paper. With the information gathered here, a loose formula for developing an overall savings realization value of using WebSphere Portal rather than a custom portal application can be established (see Table 34).

Portlet	Number of portlets deployed	Estimated hourly savings in hours	Total savings in hours (rounded)
Total base infrastructure hourly savings			**2735**
Basic functionality	5	97.9	490
Web clipping	5	387.7	1939
CRM (application integration)	4	128.8	515
Productivity portlet	5	327.7	1639
Custom portlet-simple	5	72.0	360
Custom portlet-complex	3	264.0	792
Total portlet development savings in hours			**5734**
Total maintenance savings (one-year period)			**2048**
Total savings in hours			**10517**
Assumed cost of development in hours			**75**
Total portlet cost savings — year one			**788775**
Ongoing cost savings (per year)			**153623**
Assumed number of years			**5**
Five-year savings			**1403265**

Table 34. Savings using WebSphere portal available portlets and coded portlets versus a custom portal application.

For more information

For more information on the cost savings that you can realize through WebSphere Portal software, visit:

ibm.com/software/genservers/portal

Appendix 1. Customer case studies

Customer case studies helped support this white paper. The approach involved gathering actual data from customers in various phases of portal deployment. The final data and conclusions were then validated with the customers before appearing here.

Customer case study 1

The customer in the first case study is a large insurance company in the U.S. Midwest where a portal has been deployed in a pilot phase. The development team has been gathering feedback from users to help improve the experience. The customer developed approximately 30 portlets using various methods. The information from the various methods was gathered to help build timelines for each approach. The team has had experience building portlets for WebSphere Portal with both WebSphere Studio Application Developer (using WebSphere Portal Toolkit) and the Bowstreet Portlet Factory tool. The team also estimated how long it would take to build the base infrastructure for the portion of functionality they were using in WebSphere Portal. The customer was not using any of the personalization or profiling features of WebSphere Portal at the time, so these features were not estimated in the time-saving values. WebSphere software with included portlets was used to baseline the effort. Figure 21 shows the cost savings in each scenario evaluated.

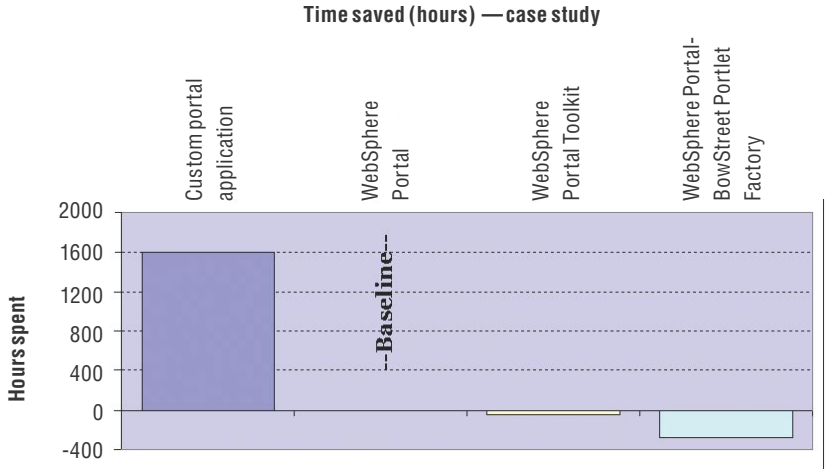

Figure 21. Case study 1 time savings

Customer case study 2

The second customer case study involves a large county government that deployed a personalized portal for constituents, employees and businesses. Users are able to profile themselves and receive personalized content based on their profile. The client also implemented a content management system that integrated with the portal using Interwoven, Inc. software products. This case study focuses on the personalization, profiling and content management in this white paper. Information from the customer was used to gather the effort involved to build a personalized profile in WebSphere Portal, as well as personalization concepts. Figure 22 shows a screen capture of the finalized portal.

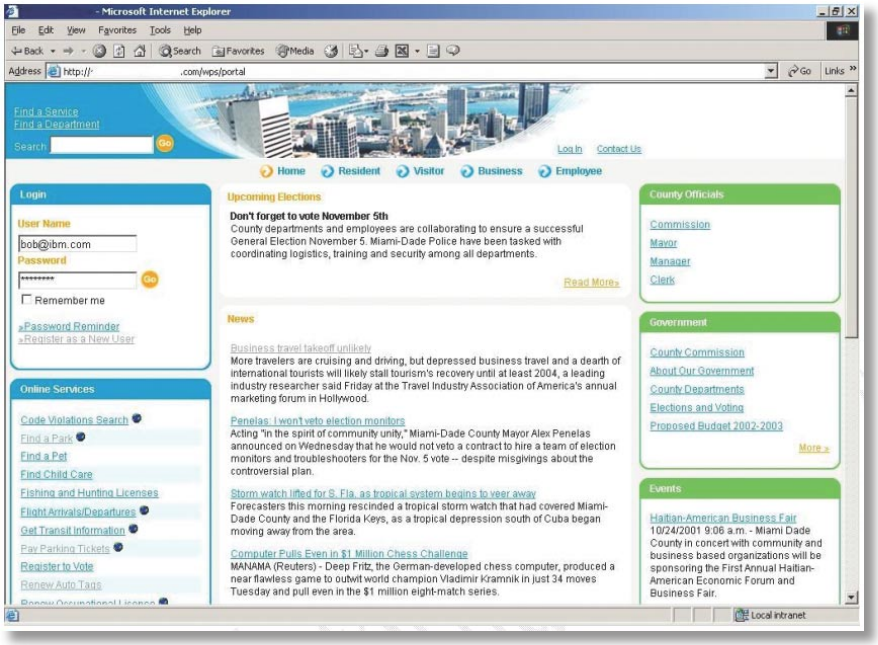

Figure 22. Personalized government portal

Table 34 shows the estimated time savings to provide new functionality and maintain WebSphere Portal in addition to the previous custom portal application for a one-year period.

Task	Time (hours)
Create new portlet applications (10 new portlets per year)	150
Deploy new portlet applications (10 new portlets per year)	100
Maintain users and groups (per year)	260
Administer security	200
Administer look and feel of the portal (per year)	80
Total	**790**

Table 34. Estimated time savings to provide new functionality and maintain WebSphere Portal for a one-year period.

Customer case study 3

The third customer case study involves IBM's own experience in deploying a customer portal application prior to the release of WebSphere Portal. The IBM intranet has been migrated to WebSphere Portal and serves more than 300 000 IBM employees as a dynamic workplace.

The former custom portal included a lightweight portal engine, rules engine and aggregation engine. Additionally, a profiling page and portal layout page was added to support personalization. Figure 23 shows a personalized view of the portal.

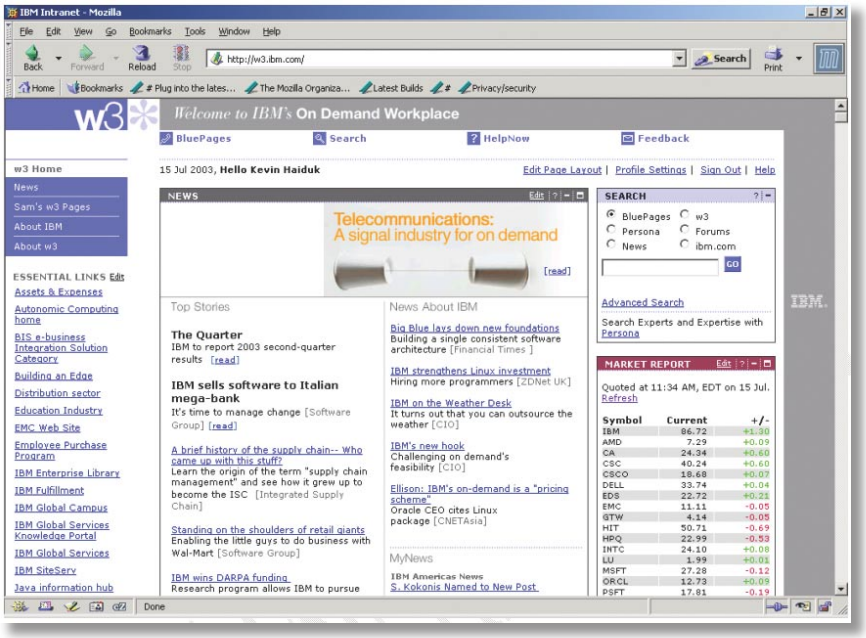

Figure 23. Personalized IBM portal

Table 35 lists the complete times for building the portal and the associated portlets.

Task	Time (hours)
Build main portal engine	540
Build rules engine	100
Build profile page	617
Build portal layout edit page	138
Build portlets (7)	1038
Total	**2003**

Table 35. The time needed to build the IBM Intranet portal and the associated portlets.

Today, the IBM On Demand Workplace is powered by WebSphere Portal software and has truly revolutionized the way people work within IBM.

Appendix 2. Key differences

Key differences in custom-profile page compared to WebSphere Portal profile page

Several differences can be seen in the custom-profile page approach that has been taken within IBM compared to the profiling system that is available in WebSphere Portal. These differences are not drastic from a development standpoint, but do impact maintainability.

- *The profile page is highly extendable in WebSphere Portal. New fields are automatically stored in the database or mapped to LDAP. The custom approach requires manual database changes and cannot be mapped to LDAP as is.*
- *The custom profile page does not provide the ability to have multiple languages.*

Key differences in custom content management system compared to WebSphere Portal Content Publisher

There are some significant differences in the capabilities of the custom content-management system described and WebSphere Portal Content Publisher, which is available with WebSphere Portal. WebSphere Portal Content Publisher provides a robust set of tools for content management and workflow.

- *WebSphere Portal Content Publisher provides unlimited levels of workflow. The custom content-management system provides no workflow.*
- *Extensively more work was required to add functionality to the custom content-management system than would be the case with WebSphere Portal Content Publisher.*
- *Because the WebSphere Portal Content Publisher system is a J2EE application, it is much more scalable.*

Key differences in personalization in a custom portal application compared to WebSphere Portal

The personalization approach in the custom portal application differs in many areas from that of WebSphere Portal.

- *Using the personalization in WebSphere Portal allows the rules of personalization to be extrapolated to a business level. The custom approach requires changing code or property files.*
- *Changes to the personalization rules can be made on the fly in WebSphere Portal. The custom approach requires a restart of the application.*
- *No campaign management is available in the custom personalization approach.*

Key differences in administration of the custom portal application compared to WebSphere Portal

Differences also exist in the administrative functionality available in the custom application portal cited in this white paper when compared to WebSphere Portal. WebSphere Portal provides a rich out-of-the-box set of administrative tools.

- *The custom portal application does not have a facility to add new portlets. Portlets have to be added manually, configured in an XML file and the application restarted.*
- *The custom portal application has no administrative interface to handle the layout of pages, places or portlets.*
- *The custom portal application has no capability to apply skins or themes.*
- *The custom portal application has no interface to apply security to users or groups.*

© Copyright IBM Corporation 2004

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

Produced in the United States of America
08-04
All Rights Reserved

DB2, Domino, IBM, the IBM logo, Lotus, Lotus Notes, MQSeries, Notes, the On Demand Business logo, QuickPlace, Sametime, VisualAge and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.

This information is for planning purposes only. The information contained in this document is provided AS IS. Any person or organization using the information is solely responsible for any and all consequences of such use. IBM accepts no liability for such consequences.

[1] The estimates for development efforts made in this white paper examine only the time for application programming. These estimates:

- Assume that a complete development environment is already installed.
- Do not include portal strategy development, user requirement gathering, design time or project management.
- Assume that developers are trained and moderately experienced on the tools being used.

[2] Using “Credential Vault to Provide Single Sign-on for Portlets.” Sukumar Konduru, September, 2002.

[3] The portlet is to be added to system, added to the XML file and the application server restarted.

[4] Security was administered by manually updating a user database.

[5] The Portal Catalog is available at on the WebSphere Portal Zone at **ibm.com**/wsdd/zones/portal/

[6] PeopleSoft Enterprise Integration Points (EIPs) are Web service connections that allow PeopleSoft applications to work smoothly with third-party systems or software and with other PeopleSoft applications.

[7] Bowstreet delivers RAD for WebSphere portlets.

[8] 65 days per year - 104 (weekend days per year) = 261 – 6 holidays = ~255 working days per year

[9] 1.5 hours saving x (255 days/4)

[10] 780.5 total hours for custom application minus 1.4 hours for deploying available portlets plus an additional 236 hours for developing portlets that are not available out of the box in WebSphere Portal.

G224-7335-00