

OmniPortlet: Publishing Any Data in Your Portal

An Oracle White Paper
May 2003

OmniPortlet: Publishing Any Data In Your Portal

Introduction	3
What is OmniPortlet ?.....	4
Creating Portlets with the OmniPortlet.....	5
Adding an OmniPortlet to a Page	5
Personalize OmniPortlet.....	7
Spreadsheet (CSV).....	8
SQL (Database Access)	9
XML.....	9
Web Services	11
Web Page.....	12
J2EE Connector Architecture (JCA).....	13
Access Secured Data	13
Filtering.....	14
Selecting the Layout.....	15
Selecting the Layout Options	17
From Portlets to Application.....	19
Simple Parameter Form Portlet	19
Packaging.....	22
Conclusion.....	23
What's Next.....	23

OmniPortlet: Publishing Any Data In Your Portal

INTRODUCTION

Enterprise portals are rapidly becoming the de facto gateway to information access and management. Organizations are realizing the benefits of this extensible framework that allows information to be presented in a single, consistent environment to end users. Oracle9iAS Portal V2 is an enterprise portal that combines a rich, declarative environment for creating a Web interface with powerful information publishing and management capabilities, as well as an extensible framework for J2EE-based applications. Using Oracle9iAS Portal V2, e-businesses have the power to connect employees, partners, and suppliers with the information they need, and the flexibility to create views tailored to each community.

One of the challenges faced when implementing an enterprise portal is that the data to be published is usually scattered throughout the organization, and exists in many forms and formats. There are the traditional enterprise data sources such as databases, but the data to be shared is increasingly stored in spreadsheets, XML or Web Services feeds – and the list of data sources continues to grow. Additionally, the quantity of data in each source is often not large enough to warrant setting up and maintaining an enterprise database just to publish it.

Simply accessing the data is not sufficient to make information useful to the end user; this data needs to be personalized. The personalization of the data is made using filtering, user criteria, and having secure access to the data sources. It is also important for an end user to have interaction with the data. For example, they can send filter criteria to one or more portlets, which will enable them transform a set of portlets into a real application that unifies the data into a single enterprise portal.

Another challenge faced when presenting information in a portal is that portlets can be relatively time-consuming to write and maintain, and portlets (either written or downloaded from the Portal Catalog) are often limiting to the portal page designer. Typically, the portal page designer has a specific look and feel identified for his page and would like the portlets on that page to seamlessly blend with it. Unfortunately, in most cases, portlet developers create a single view of the data that page designers simply have to live with.

This white paper describes the OmniPortlet – a new technology available with Oracle9iAS Portal V2, which has been created to address these challenges. The

paper introduces you to the technology, then walks you through the user experience of adding content to a page.

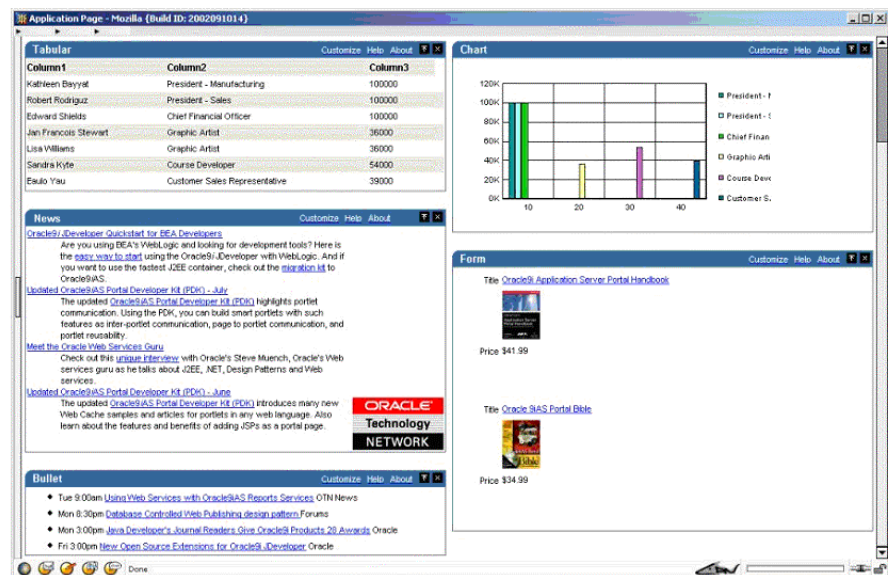
This paper assumes you have a basic understanding of Oracle9iAS Portal. For a tour of basic concepts, refer to *Getting Started with Oracle9iAS Portal*, available from the Documentation page on Portal Center (<http://portalcenter.oracle.com>)

WHAT IS OMNIPORTLET ?

OmniPortlet is a sub-component of Oracle9iAS Portal specifically targeted at enabling “power-users” (a portal page designer or content contributors) with the ability to quickly and easily publish data from various different data sources using a variety of different layouts, using a Web-based wizard.

An OmniPortlet can be based on almost any kind of data source, such as a spreadsheet (CSV), XML, Web Services, database (JDBC), or even application data from an existing Web page. To retrieve personalized data, the page designer can define the parameters for each type of data source to filter the result of a query and the credential information used to access secured data. Out-of-the-box, OmniPortlet provides the most common layout for portlets: tabular, chart, news, bullet and form.

The following is an example of an Oracle9iAS Portal page with OmniPortlets:




OmniPortlets on a Portal Page

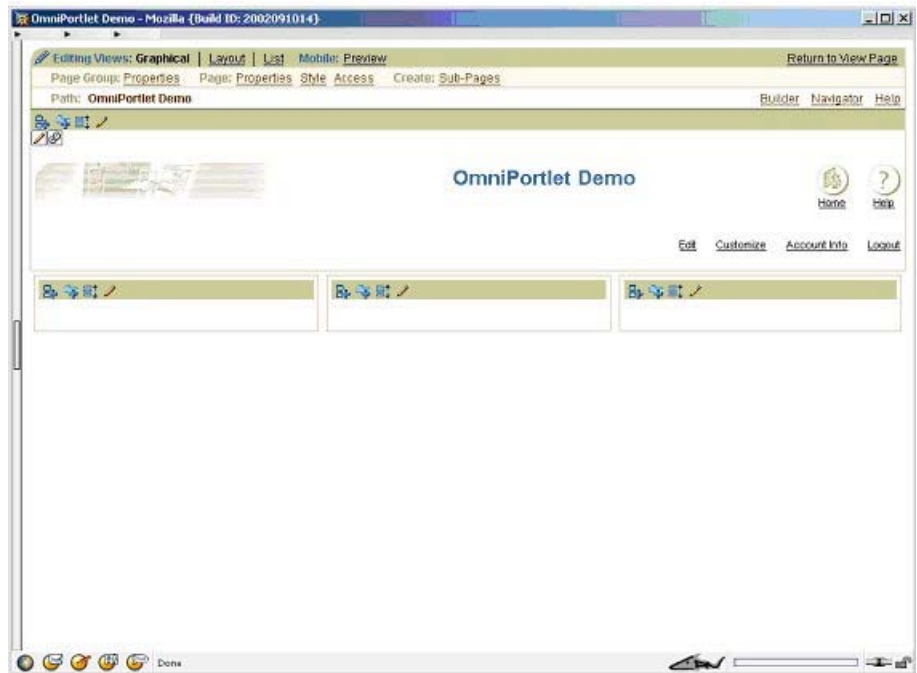
This portal page including its portlets, can be created in a matter of minutes using OmniPortlet wizard. Power users (or developers) are now more productive since they can focus on publishing the content they want using the format they want, without to worry about the low-level details of building portlets.

CREATING PORTLETS WITH THE OMNIPORTLET

This section walks through the user experience when publishing data with the OmniPortlet, and explores each of the features.

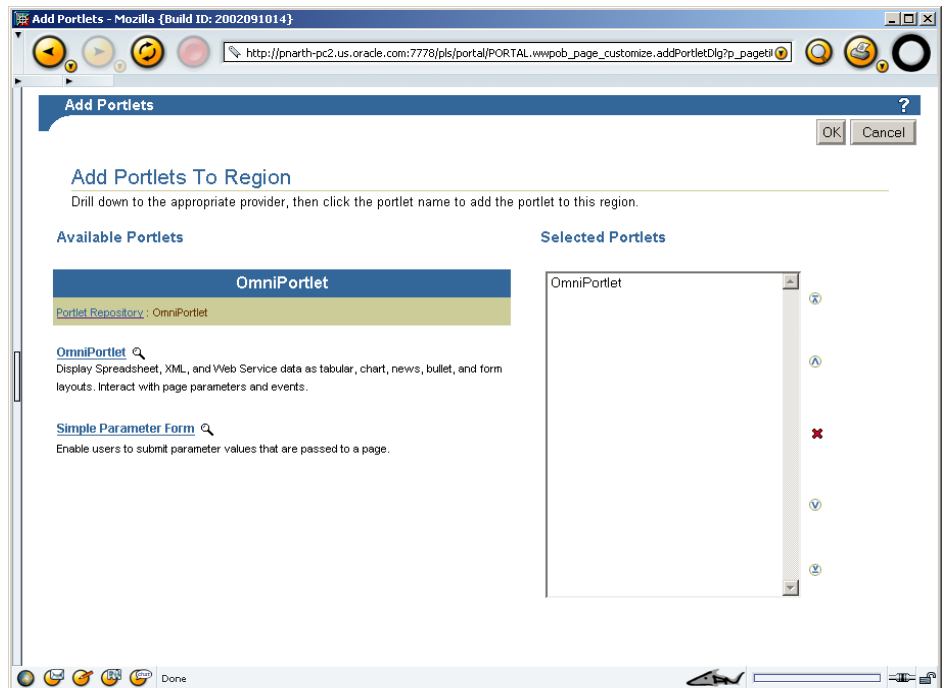
Adding an OmniPortlet to a Page

First, the page designer navigates to the page where they want to publish the data, then clicks **Edit** go into the **Edit Page** view. To add a portlet, the user clicks the **Add Portlet** icon  to display the Portlet Repository. In this image, there is a banner region and three empty regions already defined on the page:



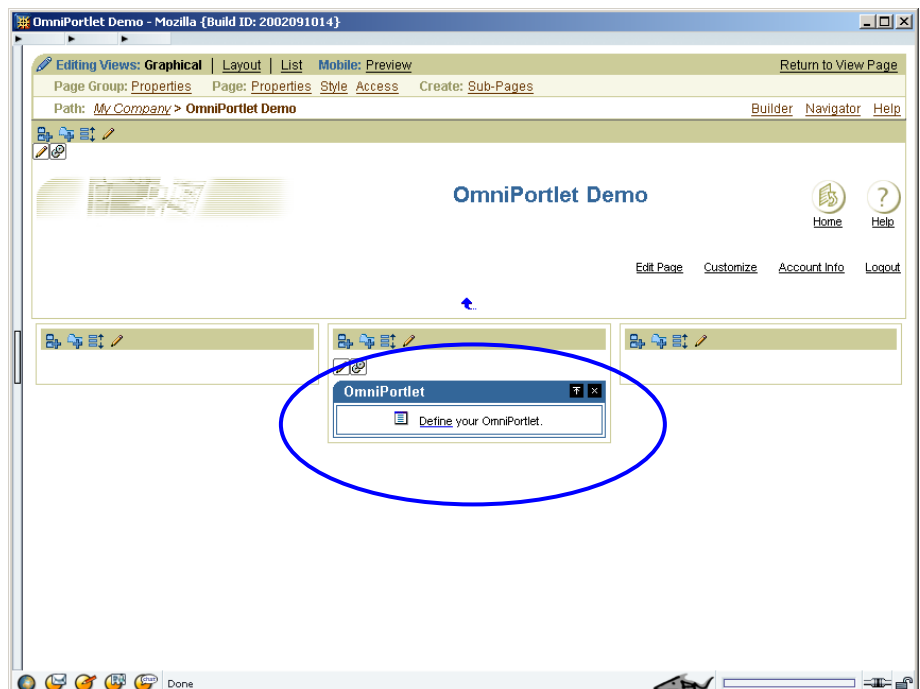
Portal Page in Edit Mode

The Portlet Repository contains all portlets and portlet providers registered with the portal. Since OmniPortlet is supplied within the OmniPortlet provider, the user selects OmniPortlet to see the portlets that are available within it.



Portlet Repository showing the OmniPortlet provider

On the Add Portlets page, they click **OmniPortlet** then click Ok. The Edit Page view displays a default view of the OmniPortlet:



Now that the user has dropped an OmniPortlet onto the page, he needs to define it to display the data in the appropriate format. To do so, he can follow these three simple steps:

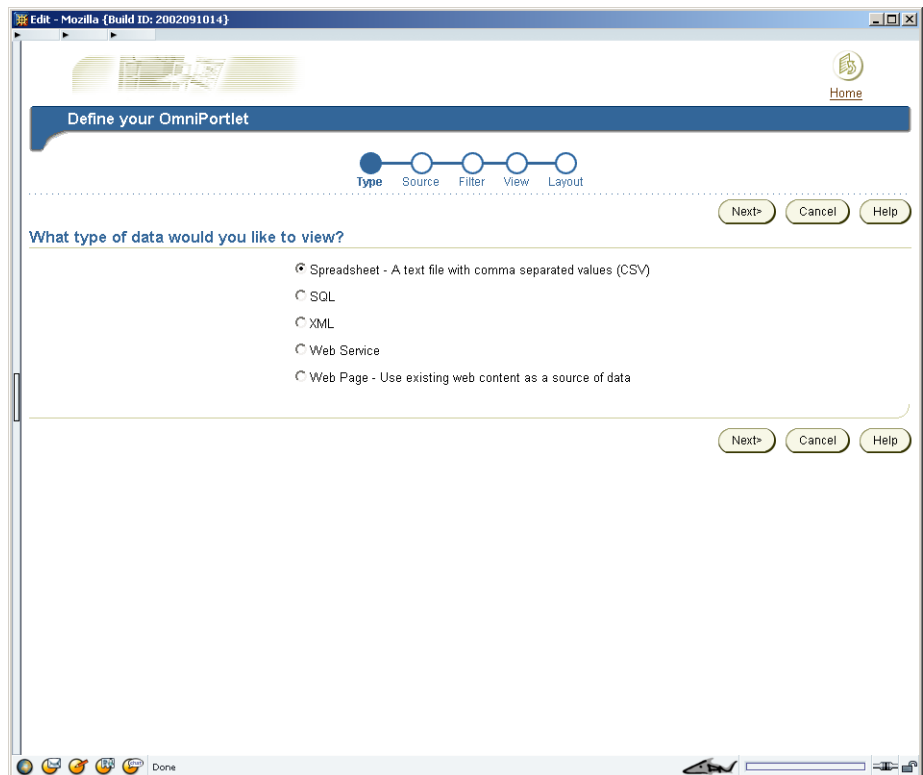
1. Personalize OmniPortlet (data source configuration)
2. Select the layout type
3. Select data to publish (i.e., the columns)

Personalize OmniPortlet

The first step in building an OmniPortlet is to define the data source and configure it to create a personalized portlet. The personalization of the data is done with different ways:

- Choose the type of data
- Configure the access to secure data
- Filter the data

Once the OmniPortlet is displayed in the Edit Page view, the user clicks the **Define** link in the portlet to edit it. The first page of the OmniPortlet wizard display, where the user can define the data source he wishes to use:



The Data Type Selector Page

These five data source types (Spreadsheet, XML, Web Services, SQL and Web Page) are described as follows:

Spreadsheet (CSV)

Spreadsheets are a very common method of storing small data sets. Users often use spreadsheets to meet their own personal publishing requirements, but then decide to share this information across a broader audience. The OmniPortlet makes it easy to share spreadsheets by supporting Comma Separated Values or CSV as a data source. When the user selects the Spreadsheet option on the Data Type Selector page, he can then specify the URL of the spreadsheet (as a CSV file) on the same page. If the spreadsheet is stored on a secured server using HTTP authentication, the page designer can define the different connection information by clicking the Define **Connection Information** button. The security mechanism is explained later in this paper in the “Access Secured Data” section.

The screenshot shows a web browser window titled 'Edit - Mozilla (Build ID: 2002091014)' displaying the 'Define your OmniPortlet' wizard. The wizard has a progress bar with steps: Type, Source, Filter, View, and Layout. The 'Source' step is currently active. Below the progress bar, there are navigation buttons: '<Previous', 'Next>', 'Finish', 'Cancel', and 'Help'. The main content area is titled 'Spreadsheet' and features an icon of a spreadsheet. Below the icon, there is a text input field for 'CSV URL' with the value '/htdocs/omniPortlet/sampleData/employees.csv'. A checkbox labeled 'Use first row of spreadsheet for column names' is checked. A tip section states: 'TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URL. Learn more...'. Below the tip, there is a 'Connection' section with the text 'To access secured data, you will need to provide connection information to the data source.' and a 'Connection Information' dropdown set to '<None>' with an 'Edit Connection' button. Two radio buttons are present: 'Use this connection information for all users' (selected) and 'User must re-enter connection information'. The 'Portlet Parameters' section follows, with a note: 'Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters by editing the Page Properties.' Below this is a table with five columns: 'Parameter Name', 'Default Value', 'Customizable', 'Customize Page Label', and 'Customize Page Description'. The table contains three rows of parameters.

Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1		<input type="checkbox"/>	Param1	Description for P1
Param2		<input type="checkbox"/>	Param2	Description for P2
Param3		<input type="checkbox"/>	Param3	Description for P3

CSV Data Source Page

SQL (Database Access)

The relational database is currently the most common place to store data. OmniPortlet offers an easy way to publish data from a database using the SQL data source. The data source uses standard JDBC drivers, and provides out-of-the-box access to Oracle and any ODBC database.

The user can enter the database connection information or reuse saved connection information to access a specific database. For more information see the “Access Secured Data” section. Once the connection information is correctly entered, any SQL statement can be written and parameterized.

The screenshot shows a web browser window titled 'Edit - Mozilla (Build ID: 2002091014)'. The page is titled 'Define your OmniPortlet' and features a progress bar with five steps: Type, Source, Filter, View, and Layout. The 'Source' step is currently selected and highlighted in blue. Below the progress bar, there are navigation buttons: '<Previous', 'Next>', 'Finish', 'Cancel', and 'Help'. The main content area is titled 'SQL' and contains a diagram of a database cylinder connected to a table icon. Below this, there is a text box labeled 'Statement' containing the SQL query: `select first_name,last_name,employee_id,salary from employees`. A tip box below the statement box reads: 'TIP You can use ##ParamN## (ex: ##Param1##) in place of any text in the SQL Query. You can also use :variable (ex: deptno = :p_dept) to define bind variables. [Learn more...](#)'. Below the tip are two buttons: 'Show Bind Variables' and 'Test'. The 'Connection' section follows, stating 'To access secured data, you will need to provide connection information to the data source.' It shows 'Connection Information' as '<None>' with an 'Edit Connection' button. There are two radio buttons: 'Use this connection information for all users' (which is selected) and 'User must re-enter connection information'. The 'Portlet Parameters' section at the bottom explains that parameters are passed to the portlet from the page when it is displayed and can be mapped to page level parameters by editing the Page Properties. The browser's status bar at the bottom shows 'Done'.

The SQL (database) Data Source Page

XML

Although still a relatively new form of storing data, XML is increasingly used as a method of providing controlled access to data sets over the intranet, and even the Internet. When the user chooses XML on the Type page, the Source page enables him to specify the URL of an XML data source.

The XML data source consumes data that is in a ROWSET/ROW structure, for example:

```
<TEAM>
  <EMPLOYEE>
    <DEPTNO>10</DEPTNO>
    <ENAME>KING</ENAME>
    <JOB>PRESIDENT</JOB>
    <SAL>5000</SAL>
  </EMPLOYEE>
  <EMPLOYEE>
    <DEPTNO>20</DEPTNO>
    <ENAME>SCOTT</ENAME>
    <JOB>ANALYST</JOB>
    <SAL>3000</SAL>
  </EMPLOYEE>
</TEAM>
```

In this instance, the <TEAM> tags delineate the rowset, and the <EMPLOYEE> tags delineate the rows. If the XML feed does not exactly match this format, the user can specify the URL to an XML style sheet (XSL) to format the data as it is being fetched. The OmniPortlet will also introspect the XML to determine the column names, which will then be used to define the layout. If a user wishes to specify this information himself, he can supply a URL to an XML Schema that describes the data. If the XML document is stored on a secure server using HTTP authentication, the page designer can define the connection information. The security mechanism is explained later in this paper in the “Access Secured Data” section. These options are shown on the following screen:

The screenshot shows a web browser window titled "Edit - Mozilla (Build ID: 2002091014)". The main content area is titled "Define your OmniPortlet" and features a progress bar with five steps: Type, Source, Filter, View, and Layout. The "Source" step is currently selected. Below the progress bar, there are navigation buttons: "<Previous", "Next>", "Finish", "Cancel", and "Help". The "XML" section is active, displaying an icon of an XML document and a text input field for the "XML URL" with the value "httpdocs/omniPortlet/sampleData/departments.xml". Below this, there are two optional sections: "Optionally enter an XSL filter to transform the data" with an "XSL Filter URL" input field, and "Optionally enter an XML Schema to describe the data" with an "XML Schema URL" input field. A tip section follows, stating: "TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs. [Learn more...](#)". The "Connection" section is next, with the text "To access secured data, you will need to provide connection information to the data source." and a "Connection Information" dropdown set to "<None>" with an "Edit Connection" button. Below this are two radio buttons: "Use this connection information for all users" (selected) and "User must re-enter connection information". At the bottom, there is a "Portlet Parameters" section. The browser's status bar at the bottom shows "Done".

XML data Source Page

Web Services

Web Services are a new way of publishing information over a network. A Web Service is a discrete business service that can be programmatically accessed over the Internet using standard protocols such as SOAP and HTTP. They are non-platform and non-language specific. Web Services are typically registered with a Web Service broker, which users may search to find a Web Service that meets their requirements. When the user finds the Web Service, he needs to obtain the URL to the WSDL (Web Service Description Language) that describes the Web Service, specifies the methods that are available to be called and the parameters that are expected, and a description of the data that is returned.

There are two main types of Web Services: Document and RPC. OmniPortlet supports both types. If the Web Service is secured using HTTP authentication, the page designer can define the connection information. The security mechanism is explained later in this paper in the “Access Secured Data” section.

Once a WSDL is supplied to the Web Service data source, it is parsed, and the user is presented with the available methods to be called. The user can pick a method, and then supply values for parameters required by that method.

As with the XML data source, data is expected in a ROWSET/ROW format. If it is not returned in this format, users may specify an XSL file to format it. Also similar to the XML data source, column names are determined by introspecting the WSDL, but users can specify an XML Schema to describe the returning data set.

Define your OmniPortlet

Type Source Filter View Layout

Web Service

WSDL URL Show Methods

Web Service Methods

Available methods for this Web Service OTNDeptEmp.getDeptInfoXML Show Parameters

Enter values for the method parameters

deptno Test

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the method parameters. [Learn more...](#)

Optionally enter an XSL filter to transform the method output
This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL Test

Optionally enter an XML Schema to describe the method output
This is useful when the XML data doesn't have data for all fields or to override what is defined in the XML data.

XML Schema URL

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs. [Learn more...](#)

Web Service Data Source Page

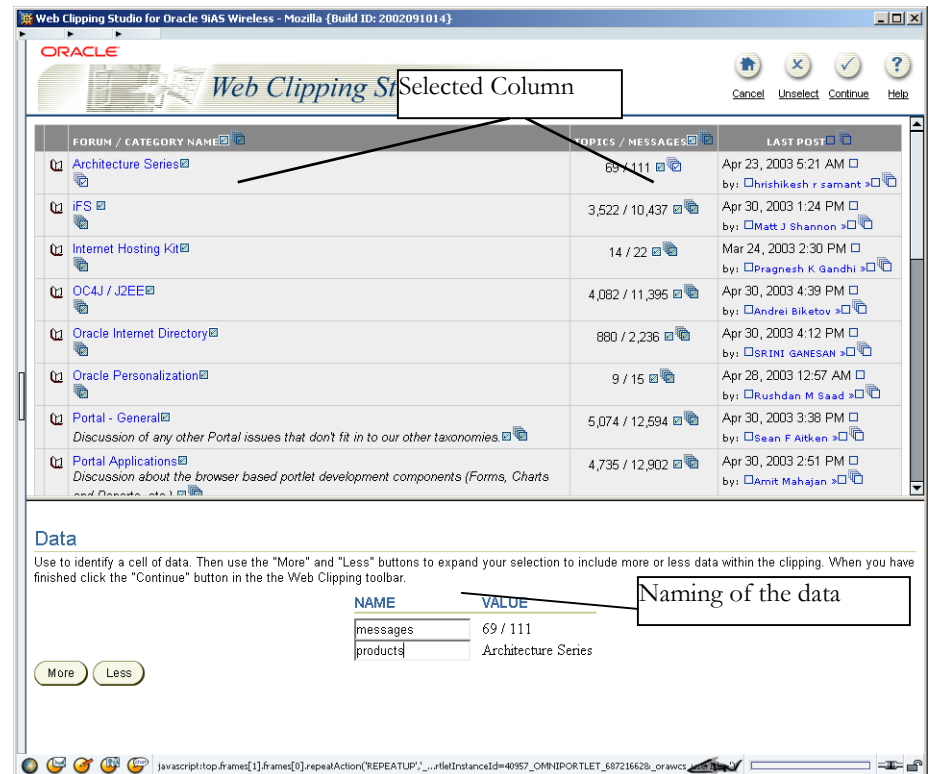
Web Page

Since many applications are now Web based, information needed by the end user is often already accessible from the browser. The challenge is to leverage the existing application and repurpose this application into a portal.

The first answer to this challenge is the Web Clipping portlet, released in November 2002. Web Clippings are pieces of existing Web content that can be published as portlet. For more information about the Web Clipping portlet, visit the **OmniPortlet and WebClipping** page on the Portal Center Web site (<http://portalcenter.oracle.com>)

The Web Page data source enables the user to capture the data of a Web application and easily integrate it into a portal page. This data source use the Web Clipping Studio to “surf” the application and while retaining the business logic, extract the data and publish them into a portlet.

The following image shows how the page designer can select the data to publish



Web Clipping Studio Page in the Web Page Data Source

The Web Clipping Studio allows the user to check the columns of any HTML table, from where the data should be extracted, then the user can name the column returned to the OmniPortlet.

J2EE Connector Architecture (JCA)

Although not displayed on the Type page of the OmniPortlet wizard, a J2EE™ Connector Architecture (JCA) 1.0 adapter is also available. JCA provides a mechanism to store and retrieve enterprise data such as that held in ERP systems (Oracle Financials, SAP, PeopleSoft, etc). Using JCA to access an ERP is akin to using JDBC (Java Database Connectivity) to access a database. Due to the current limitations of the JCA 1.0 Standard, it is not possible to provide a declarative user experience for specification of the JCA data for the portlet (which is why it is not displayed in the OmniPortlet wizard). JCA 1.0 does not support object introspection (querying of a data source to determine the objects available). This feature will be available as part of a revision of the standard in the very near future. At this point, the JCA adapter with the OmniPortlet will be upgraded to be exposed in the user interface. Use of this adapter with the OmniPortlet may be achieved today by editing the metadata definition of the portlet (held in the `providers.xml` file).

Access Secured Data

In a production environment, most of the data is secure. In some cases, the authentication information is not only used to protect the data but also to retrieve personalized data. A typical example is a mail application, where the username and password that the user enters is used to retrieve his mailbox only. Portal, as the gateway to the information, supports access to secure data, but also needs to offer an easy way to send the user information to retrieve the right data. OmniPortlet data sources support connection to secured data. The most common way is the SQL data source, where the user enters the login information. OmniPortlet also supports data source protected by HTTP basic authentication. Since SSL is often used to protect the communication between the server and the client, OmniPortlet supports the usage of HTTPS.

To protect the data and the connection between the Web server and the client (in our case the OmniPortlet) OmniPortlet data sources support the usage of HTTPS.

The page designer manages the data source security during the definition of the OmniPortlet, and the process is the same independent of the data source type. The page designer can decide if the credential information needs to be shared by all users or customizable for each user.

The following image shows the Connection Information page for the SQL data source:

Define your OmniPortlet

Connection Information

OK Cancel Help

Name

A connection name is optional and provides access to connection information that can be used by multiple portlets. Changing the parameter values of a named connection will change the values for all portlets that use this name.

Connection Name

Parameters

Username

Password

Connection String

Driver Name

Test

Customizable

☐

☐

☐

☐

✓ TIP You can use the format ##ParamN## (ex : ##Param1##) in place of Username, Password or Connection String. Sample Username, Password and Connection String would be "scott", "tiger" and "mymachine:1521:iasdb" respectively. [Learn more...](#)

✓ TIP The Connection String format depends on the driver chosen. For "Oracle-thin" driver, the format is Host:Port:SID. [Learn more...](#)

OK Cancel Help

Connection Information Page

The Connection Name is used to identify a set of connection parameters. This value will be added to the list of known named connections. The user can also browse to select an existing named connection. The information specified for a named connection will be used for all portlets on a page that use the same named connection. Thus, when the user enters the appropriate credentials for one portlet that uses this named connection, he will be authenticated for all portlets that use the same named connection.

Filtering

After the user has selected a data source, he can further refine the retrieved data by filtering and sorting it. He can select specific data by using a filtering option that restricts the data based on user-defined criteria.

OmniPortlet supports filtering of data in two areas:

- In the data source itself by specifying what data is retrieved from the actual data source. This is specified on the Source page
- In the middle tier; this is specified on the Filter page

It is always more efficient to filter and sort the data at the data source, since the user can streamline the process by retrieving only the necessary data. The best

practice is to only use the middle tier filtering when the data source does not support any filtering or sorting capability, for example, a spreadsheet.

The following image illustrates the Filter page of the OmniPortlet wizard:

Define your OmniPortlet

Type Source **Filter** View Layout

<Previous Next> Finish Cancel Help

Filter

Use this page to filter and order the data that appears in your portlet.

Conditions

Specify the conditions that the data must meet in order to appear in your portlet. Click the plus sign to specify conditions for additional columns.

☒ Match all ☐ Match any

Column	Operator	Value	Actions
Department_No	Equals	30	+ x
Vacation_Days	Greater than	10	+ x

Order

Determine how the data should be ordered.

Column	Order
Order by	<None> Ascending
Then by	<None> Ascending
Then by	<None> Ascending

Limit

If desired, limit the number of rows that appear in the portlet.

☒ Do not limit results ☐ Limit to 10 results

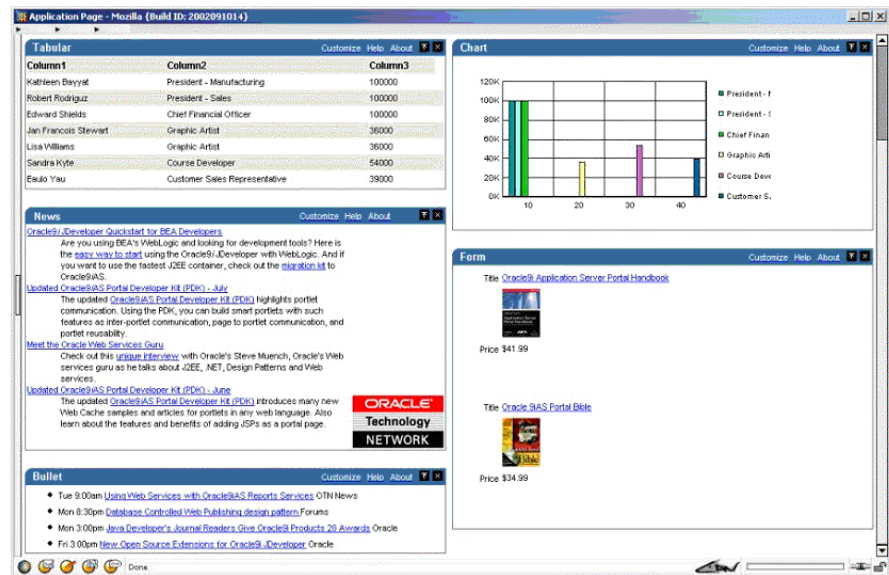
Filter Page

Selecting the Layout

Once the data source is specified, the layout and the actual objects to publish need to be determined. The OmniPortlet provides several out-of-the-box layouts:

- Tabular
- Chart
- News
- Form
- Bullet

The OmniPortlet layouts are based on the Oracle9iAS Portal styles, which means that all OmniPortlet instances will be consistent with the rest of the page. If the page designer changes the style at the template level, the OmniPortlet will automatically reflect the change. Each of the layout formats are shown in the following image:



Different Types of OmniPortlet Layout

The OmniPortlet maintains a separation between data and layout. That is, users may select a data source and select a layout, then may later change the layout without having to rebuild the portlet. For example, data may be rendered initially as a tabular and then changed to a chart, simply by re-entering the wizard and changing the layout type. This flexibility means that users can easily choose the method of publishing their data that is right for them.

Selecting the Layout Options

On the View page, the user can set the title, header and footer text for the portlet, and pick the layout style they wish to use. The following image shows the page that allows the user to select a layout:

Define your News

Type Source Filter **View** Layout

<Previous Next> Finish Cancel Help

Header and Footer Text

Title: OmniPortlet

Header Text:

☒ Show Header Text

Footer Text:

☒ Show Footer Text

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the text, and ##Column Name## (ex: ##DEPTNO##) to pass the column values into the text. [Learn more...](#)

Layout Style

Select a layout for the portlet data.

☒ Tabular ☐ Chart ☐ News ☐ Bullet ☐ Form

Data View Customize Help About F x

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguez	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyle	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

Caching

If page level caching is not used, then the page will always have to wait for this portlet to generate if the portlet is not cached. This can adversely effect page performance.

Layout Type Selector Page

Once the layout style has been selected, the layout options need to be selected, which include several different, layout-specific options, such as:

- For the tabular layout: select whether to see alternating colors for each row.
- For the news layout: select whether to use an image (typically the image of the news provider), or show the news into an automatic scrolling zone.
- For the bullet layout: select the bullet format (disk, number, letter, Roman numerals, etc.).
- For the form layout: select whether the form fields are horizontal or vertical.
- For the chart layout: select the type of chart (e.g., bar, pie, line) and if the data is already grouped (if so, by what column).

One important feature that the OmniPortlet supports is drill-down hyperlinks in chart. That is, a user can program the chart so that when an end user clicks on a

specific part of the chart, an action can occur (e.g., jump to another URL and pass data with the URL).

For the other layout styles, the user can define each column to be displayed in a specific format such as plain text, HTML, image, button or field. An example of this would be if the user selected a data source that contains a URL to an image. To see this image, the power user can select “Image” for the display of this column. Each column may also raise an action (similar to chart drilldown hyperlinks). Such actions will be covered in more detail in the next section.

From Portlets to Application

By publishing various portlets on a single page, the end user is given a consistent and unified view of the data. The next step is to integrate portlets to create a useful application.

Page parameters and events can be used with OmniPortlet to create a cohesive application. Page parameters allow the page designer to declare a data input interface to a page. This interface can then be used to transmit data from the page to one or more portlets in a standardized method directly supported by Portal. Portlet events model the page navigation and the data output of a portlet when a user clicks on a link or submits a form. When an event is sent, a payload is also sent with it – which can be one or more parameters.

The OmniPortlet is capable of receiving up to five parameters and raising up to three different events. Each of the events can send one or more parameters as the payload. In the example application below, the OmniPortlet chart on the right can raise a page event when an end user clicks on one of the bars on the chart. The event that is raised also sends the data that the chart rendered (e.g., the department number) as the payload. The event sets the value of a page parameter, which the OmniPortlet reacts to. So when the page refreshes after the user has clicked on the bar chart, he sees all the information specific to that department.



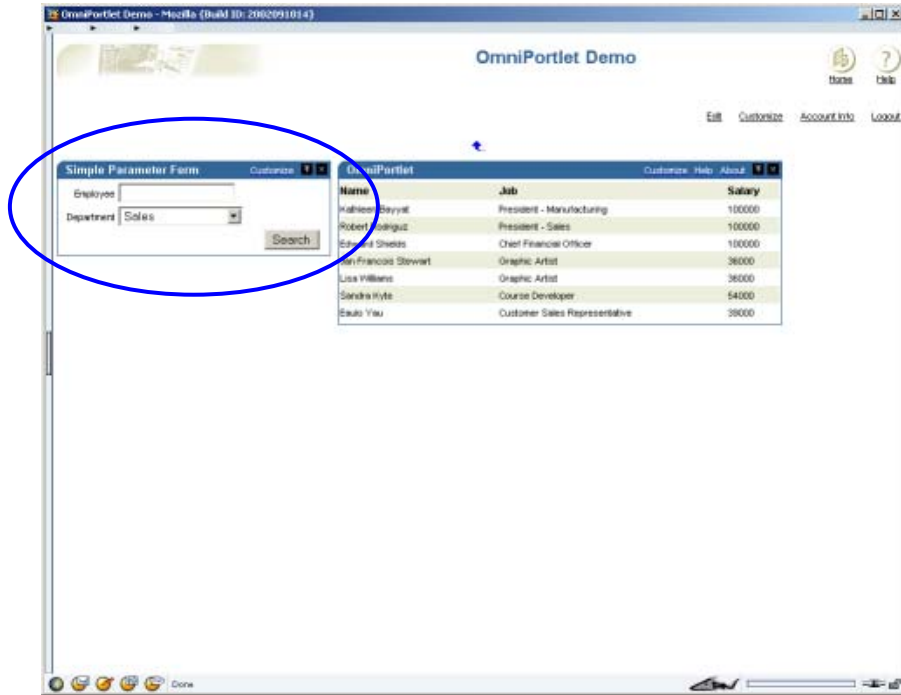
Example of Graph controlling another OmniPortlet

Simple Parameter Form Portlet

In addition to the layout types supported, the OmniPortlet Web provider contains another special type of portlet: the *Simple Parameter Form Portlet*. This portlet allows the page designer to create simple parameter forms where end users may enter data

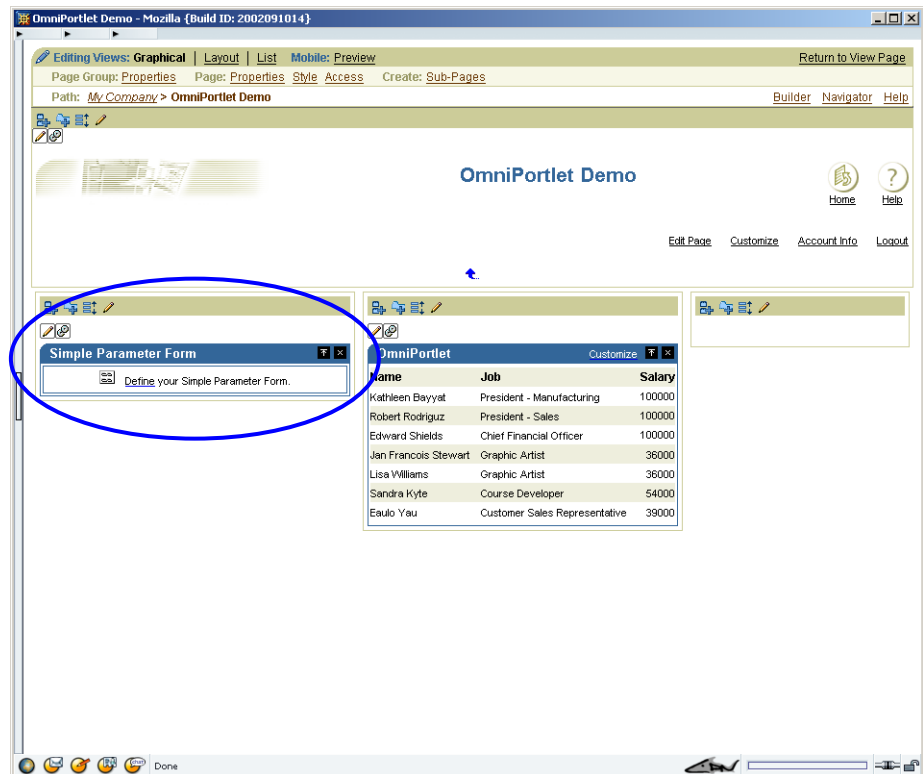
or pick from a list of predefined options, then easily set Oracle*9i*AS Portal page parameters and events.

The following image shows an example simple parameter form that allows an end user to perform a search for an employee or department name:



Defined Simple Parameter Form Portlet

Once the simple parameter form portlet is added to a region, page designers click on the **Define** link to configure the portlet:



The Simple Parameter Form portlet has fewer steps and options than the OmniPortlet, since page designers only need to specify whatever it should have a horizontal or vertical appearance, what input parameters it should accept, and what data should be sent when the end user submits the data.

Define your Simple Parameter Form

View Layout Parameters

<Previous Next> Finish Cancel Help

Form Style

Vertical Horizontal

Data View Customize Help About

Sample Data:

Name: Kathleen Bayyat
 Job: President - Manufacturi
 Salary: 100000
 Commission: 100000

Submit

Entry Fields

Select which parameters to display in the form.

Name	Label	Parameter	Alignment	Display As	Width	List of Values
Field1	Employee	Param1	Left	Field	20	
Field2	Department	Param2	Left	List of Values	20	10:Sales, 20:Purcha
Field3	Param3	Param3	Left	Hidden	20	
Field4	Param4	Param4	Left	Hidden	20	
Field5	Param5	Param5	Left	Hidden	20	

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the field label and list of values. [Learn more...](#)

TIP If you select Display As Text Box, enter the text box's width and height separated by a comma or enter the width only in which case the height is defaulted to '1'. (ex: '20, 2' or '20')

TIP If you select Display As List of Values, enter a list of comma separated values in the List of Values field (ex: '10:Sales, 20:Purchasing, 30:Human Resources, etc.' or 'Sales, Purchasing, Human Resources, etc.').

Submit Action

LAYOUT DEFINITION OF THE SIMPLE PARAMETER FORM

PACKAGING

The OmniPortlet is available with the Oracle9iAS Portal Development Kit (PDK), which you can download for free from Portal Center (<http://portalcenter.oracle.com>). It is implemented as a Web Provider, which means that it is:

- J2EE™ compliant
- Distributes the load by allowing the OmniPortlet code to be located either locally with Oracle9iAS Portal, or remotely
- Will respect the emerging standard specifications (once implemented by the PDK):
 - Oasis Web Service for Remote Portlet (WSRP)
 - Java Community Process Portlet Specifications (JSR 168)

In addition, the component technology that the OmniPortlet has been developed with has been pushed down into the core PDK. This means PDK developers will soon be able to leverage new Java API's and JSP tag libraries when creating their own portlets and seamlessly access OmniPortlet functionality. For example:

- Reuse a data source in their own java portlet
- Create a multi-tab Customize or Edit Defaults page for their portlet

CONCLUSION

The new OmniPortlet greatly enhances the page designer's experience with publishing content to Oracle9iAS Portal. It permits the user to choose between several different data sources (Spreadsheets (CSV), XML, Web Services, Databases, Web Page and JCA), and to render their data as a tabular, bulleted list, chart, news, form. Users can also select different layouts and change them independently of the data source, giving maximum flexibility. By combining the OmniPortlet, the Simple Parameter Form portlet, and the Portal page parameters and events model, users can create simple but powerful applications that unified the information from different sources. By providing a flexible architecture that allows users to extend both the data sources and the renditions, OmniPortlet puts publishing control where it belongs: with the page designer. OmniPortlet is a critical next step in the evolution of Portals.

What's Next?

To continue to enable the page designer to build competitive and powerful portals, OmniPortlet will provide even more data source and layout capabilities. The future release of OmniPortlet will provide a way to developer and register custom layouts and introduce an Oracle9iAS Integration data source to leverage the different adapters.

Custom Layout

The layouts provided out-of-the-box represent the most common portlet styles used when rendering data. In future releases of OmniPortlet, the page designer will be able to extend these existing layout styles to accommodate the portal requirements. The extensibility of the layout will be based on the J2EE standard for the Web Page development: JavaServer Pages (JSP), and JavaServer Pages Standard Tag Library (JSTL).

Oracle9iAS Integration data source

Today, end users must be able to access information via their portals from various systems, such as ERPs and legacy systems. The current version of OmniPortlet can access these systems using XML and Web services, but these ERP and legacy systems may not have an XML or Web service interface. Oracle9iAS Integration provides native access to ERPs (SAP, PeopleSoft, JDEwards, Siebel) and legacy system (CICS, VSAM, Tuxedo) using adapters based on JCA. Future releases of OmniPortlet will provide a data source that leverages these adapters to declaratively access these systems.



OmniPortlet: Declarative Unification of Your Data in Portal

May 2003

Author: Tugdual Grall

Contributing Authors: Paul Narth, Vanessa Wang

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Copyright © 2003, Oracle. All rights reserved.

This document is provided for information purposes only
and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to
any other warranties or conditions, whether expressed orally
or implied in law, including implied warranties and conditions of
merchantability or fitness for a particular purpose. We specifically
disclaim any liability with respect to this document and no
contractual obligations are formed either directly or indirectly
by this document. This document may not be reproduced or
transmitted in any form or by any means, electronic or mechanical,
for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective owners.