


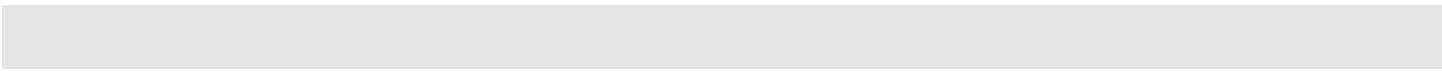


# Portal Integration with SAP Applications

*10 March 2006*

Version 2.0





The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided “as is” without warranty of any kind, express or implied. In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

© Copyright IBM Corporation 2005

Produced in the United States  
August 2005  
All Rights Reserved

IBM, the IBM logo, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries or both.

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

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

BEA and WebLogic are trademarks of Sun BEA Systems, Inc. in the United States, other countries or both.

JBoss is a registered trademark and servicemark of JBoss, Inc. in the United States, other countries or both.

SAP, ABAP and NetWeaver are trademarks or registered trademarks of SAP AG in Germany and several other countries.

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

# Portal Integration with SAP Applications

*A comparison of IBM WebSphere Portal 5.1 and SAP Enterprise Portal 6 with respect to integrating with back-end SAP applications*

## **Executive Summary**

One of the key decision criteria in evaluating portal solutions is the ability to integrate with critical business applications and data. This is especially true for customers who have made significant investments in SAP applications.

This study consisted of looking at portal integration from two different perspectives: 1) user interface integration, and 2) application-level integration. With user interface integration, the objective is for the portal to render the existing presentation stream of the SAP application in a corresponding portlet (or iView for SAP Enterprise Portal). In the case of application-level integration, the goal is to use the application's native programming interface (BAPI, RFC, etc.) in order to access specific application functionality or data and render it within a portlet or iView. This option typically provides portlet/iView developers with more control over the presentation of the information. It also permits data from multiple sources to be aggregated and presented to the user with a more unified view.

A common misconception among customers is that SAP's Enterprise Portal offering is better suited, indeed, required in order to integrate well with these SAP back-end systems. The results of our study show that IBM's WebSphere Portal 5.1 product is equally adept at integrating with SAP environments, whether at the user interface level or the application level. This opens the door for customers to consider other factors in their enterprise portal selection, such as broad support for accessing non-SAP applications and data, portal and collaboration standards, performance/scalability, virtual portal support, user process integration support, portal content management and personalization, among many others. In these areas, IBM offers sizable advantages over SAP EP.

## Introduction

One of the hallmarks of enterprise portals is the ability to integrate multiple disparate applications “at the glass” in order to improve the overall productivity of users. This integration usually takes two forms: 1) User Interface-level, and 2) Application-Level (Figure 1).

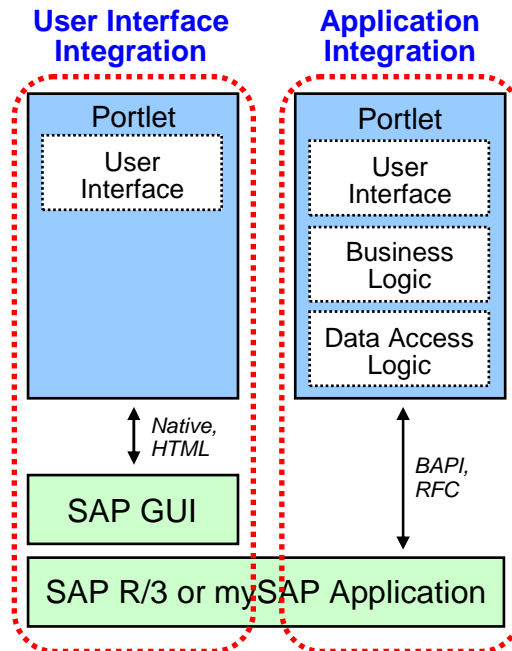


Figure 1. Types of Integration with Back-End SAP Applications

With **User Interface** integration, a portlet encapsulates the presentation stream of the existing application and displays it inside a portlet window. For non-Web applications (e.g. Windows-based GUIs), third party solutions like those from Citrix can provide a means for portals to include these native GUIs inside a portlet. For Web applications, this is easily accomplished using the HTML iFrame standard. Most portal vendors (including IBM and SAP) offer iFrame-based portlets that allow the end-user to specify the URL of the Web application as part of the portlet configuration step. The resulting Web interface of the application is then rendered in a portlet window with no coding required. While quick and easy, this approach does have its drawbacks. Most Web applications are designed to operate in full-screen mode, not inside a portlet window a fraction of that size. As a result, iFrame-based portlets can require the user to scroll up or down and side-to-side in order to navigate the entire application interface. In addition, many Web applications run on servers remote to the portal. By not having control over the entire application and just capturing the presentation stream, the portal is unable to leverage all of its underlying infrastructure services. One example would be portlet-to-portlet messaging. Another might be internationalization support, where the portlet content is rendered in the language-of-choice. Yet another could be the ability to personalize content so that the user is shown only those items that have attributes that closely match those from his login profile or past usage patterns.

In order to realize these benefits and exercise maximum control over what is presented to a user and how it is rendered, many developers use “native” portlets or those that use the **Application-Level** of integration. With this approach, the portlet has its own presentation, business, and data access logic. It uses the API of the back-end application (or database) to retrieve the desired information and, ideally, renders it so that all the contents can be viewable without scrolling. Because this portlet application runs inside the portal, it has full access to all the infrastructure services available to it. Both IBM and SAP offer several solutions to facilitate Application-Level integration with back-end SAP applications. Some involve the use of wizards and pre-defined input and output templates to minimize or eliminate the need for programming skills. Others involve the use of visual drag-and-drop tools to define inputs and outputs while permitting customization of how the information is displayed on the screen. Regardless of the implementation method chosen, the end results are essentially the same for both vendors: portlets

that can access SAP applications and data in a manner consistent with the user’s needs and the role that they perform within their company.

The tools used by both IBM and SAP to handle User Interface and Application-Level integration are as follows:

	IBM WebSphere Portal v5.1	SAP Enterprise Portal v6.0
User Interface Integration	SAP Integration Portlets, Web Clipper portlet	SAP Content Studio
Application-Level Integration (user-driven)	IBM WebSphere Portlet Application Integrator (WPAI), Bowstreet Portlet Factory	SAP Visual Composer
Application-Level Integration (developer-driven)	IBM Rational Application Developer v6	SAP NetWeaver Developer Studio

## User Interface Integration

Like many application vendors, SAP provides both native and Web-based user interfaces for their back-end applications. The SAP Client-Server GUI is a “thick” client interface that must be installed on each user’s desktop in order to render and interact with SAP back-end applications. This remains true even if the customer wishes to use a portlet that also renders this same interface within the portal. As its name implies, the SAP HTML GUI renders the user interface of the back-end application via a Web browser.

Over the years, SAP has used different presentation technologies across their SAP application portfolio. These include the following:

- **Business Server Pages (BSP):** server-side ABAP technology used in CRM and other mySAP applications
- **Business Warehouse Explorer:** browser-based reports created against SAP’s data warehouse
- **Internet Application Components (IAC):** precursors to iViews, these are Web-based “mini-applications” that access related back-end SAP functions or data. Examples include early renditions of the Employee Self-Service Business Package.
- **MiniApps:** precursor to Internet Application Components, they provide access to a set of related back-end SAP functions and or data. They were used also in some early iView Business Packages.
- **Transaction:** returns the results of a given transaction ID to either a Web browser or the SAP “thick” client.
- **Web Dynpro:** proprietary build-time and runtime Web-based application environment that provides a higher level of abstraction on top of J2EE

SAP EP utilizes **Content Studio** to enable users/administrators to create iViews for each of these technologies. Content Studio is a component of the SAP EP Web-based administration console and provides a wizard-based approach in building iViews out of existing content. In each case, the user specifies the unique artifact (e.g. transaction ID, BSP, Business Warehouse report, IAC/MiniApp, etc.) to be included in an iView and the tool automatically generates and deploys it into the SAP EP runtime. No coding is required and the user interface displays unmodified in an iView window (iFrame-based).

**IBM WebSphere Portal** offers similar capabilities through its **SAP Integration Portlets** and the Web Clipper (iFrame) portlet. The SAP Integration Portlets are actually a set of three configurable portlets that support different User Interface types:

- SAP HTML GUI
- SAP Business Warehouse
- SAP EP iViews

Customers can utilize the SAP HTML GUI portlet to specify a specific transaction ID, or the URLs associated with an IAC or MiniApp in order to render the user interface of these applications in a corresponding portlet. Similarly, customers can specify which report they wish to display using the SAP Business Warehouse portlet. Finally, existing iViews can also be rendered in WebSphere Portal using the SAP EP iViews portlet. The latter option is important for customers who have already invested in SAP Business Packages like Employee Self-Service (ESS) or Manager Self-Service (MSS) to front-end their back-end SAP Applications. SAP Business Packages are a collection of related iViews.

In the case of Business Server Pages and Web Dynpro applications, both browser-based, customers can utilize IBM's Web Clipper (iFrame) portlet to specify the URL for the BSP or Web Dynpro application in order to have its contents displayed in the portlet window.

In some instances, customers may wish to expose an SAP "thick" client GUI in WebSphere Portal. To do this, they can utilize the Citrix portlet available for download in the WebSphere Portal Portlet Catalog in conjunction with the Citrix Presentation Server (purchased separately).

The fact that SAP, as well as IBM, relies on the HTML iFrame standard for handling User Interface integration within the portal means that there is very little differentiation between the two vendors in this area. Thus, there are no inherent advantages to using SAP EP over IBM WebSphere Portal for User Interface integration (Figure 2).

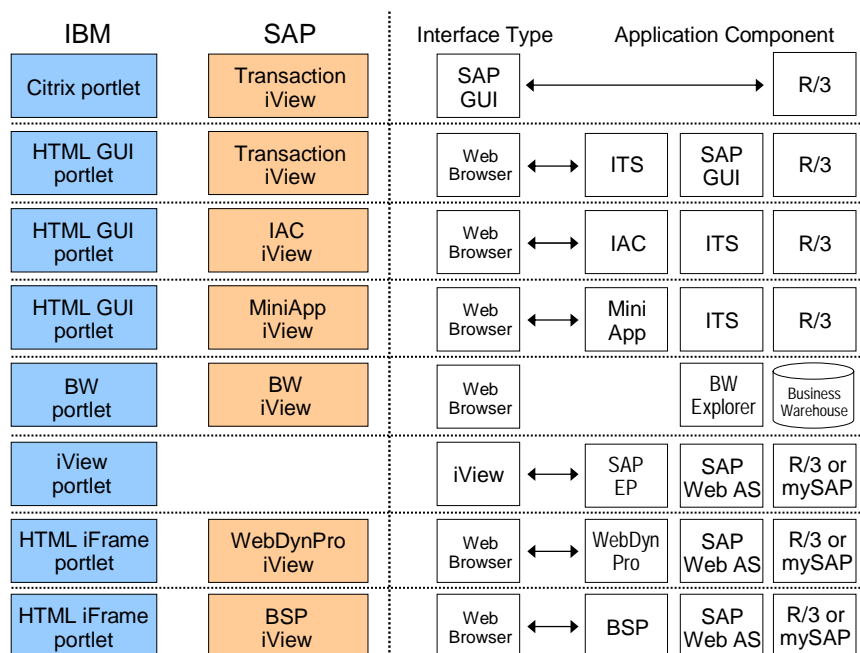


Figure 2. IBM and SAP User Interface Integration Solutions

## Application-Level Integration

The monolithic structure of most existing enterprise applications typically requires navigation through multiple user screens before the desired data is located. In addition, there is often a need to combine data across different application functions in order to present a more holistic view (e.g. Customer detail). This requires customers to build "native" portlets that have their own presentation, business, and data access logic. There are different tools to accomplish this, depending upon the skills available and the degree of customization required. Some are aimed at the business user, while others are targeted towards the professional developer.

### User-Driven

SAP is promoting the use of **Visual Composer** for tech-savvy end-users who are business experts but lack general programming skills. As its name implies, it features a highly visual, drag-and-drop interface for defining inputs and outputs (Figure 3), as well as graphically building input and output forms with which the end-user interacts (Figure 4).

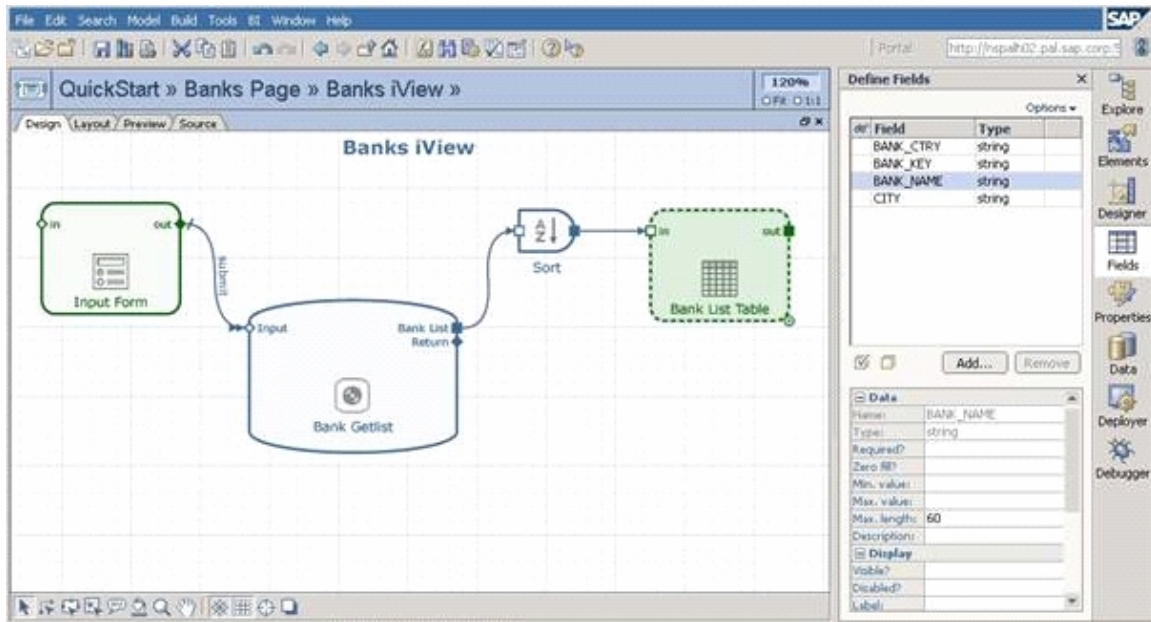


Figure 3. SAP Visual Composer workspace for drag-and-drop inputs and outputs

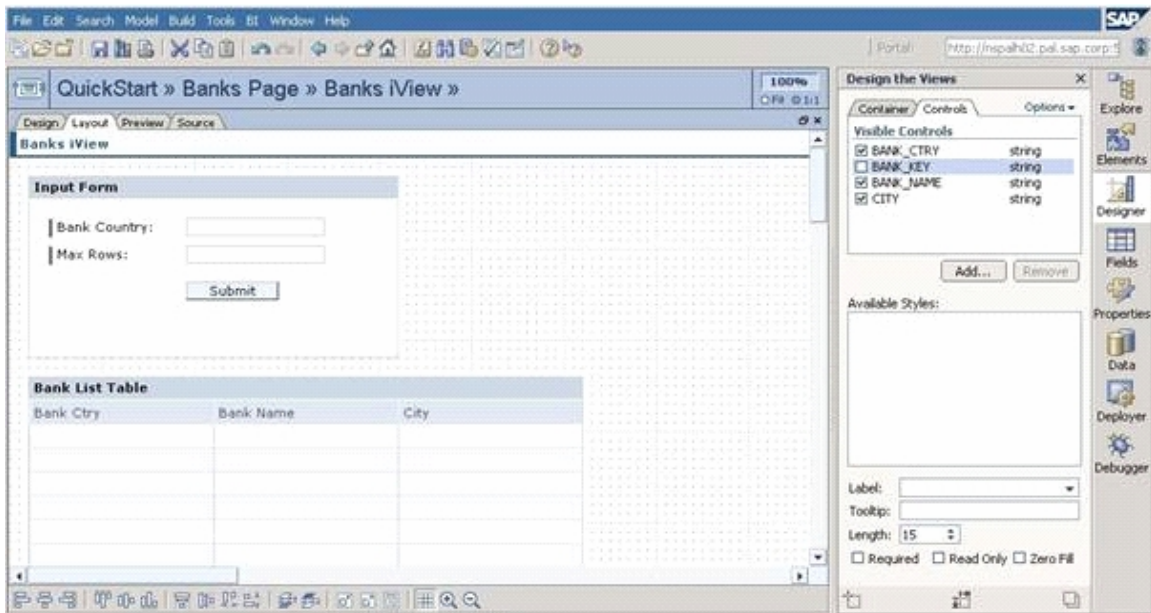


Figure 4. SAP Visual Composer workspace for defining input and output form layouts

Although Visual Composer is browser-based, it requires third party plug-ins and requires a Microsoft-centric server-side infrastructure that front-ends SAP EP in order to operate (Figure 5). For example, the server-side components require Windows Server 2000, Internet Information Server (IIS) 5.0, and SQL Server 2000, which the customer must purchase/acquire separately. In addition, the Internet Explorer (IE) browser must be at the 6.0 release level and must have the Microsoft XML Parser 4.0 and Adobe SVG Viewer 3.0 installed as well.

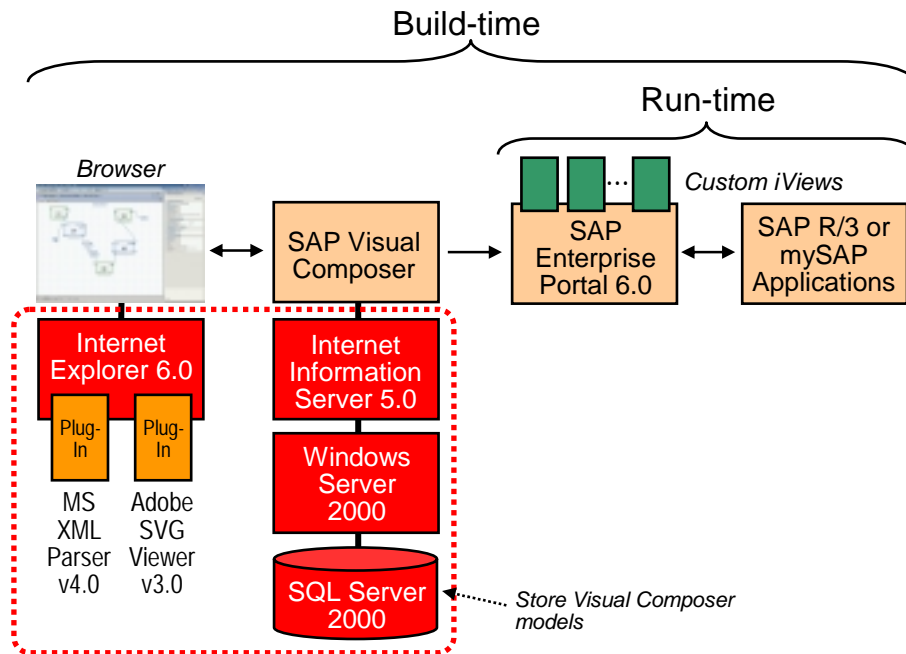


Figure 5. SAP Visual Composer Build-time and Runtime Components

To create new iViews that access back-end SAP applications and data, the following steps are required:

1. The user logs in to SAP EP and browses/searches for the desired BAPI or RFC. Once found, the user drags the BAPI function onto his/her workspace to create a corresponding data service. The user then tests it out using the BAPI test dialog to verify that it provides the desired information.
2. The user then defines the search (input) form by dragging and dropping from the input port of the previously defined data service to an empty location on the workspace. The Field Tasks panel is used to define which fields to submit as input to the data service.
3. The outputs are defined in a similar manner, with the user dragging and dropping from the output port of the data service to another empty section of the workspace. Likewise, the Field Tasks panel is used to define which data is displayed.
4. The next step is to customize the layout of both the input form and output table. A default layout is generated based on the selected BAPI or RFC from earlier steps, which the user can then manipulate using WYSIWYG techniques.
5. The “preview” mode allows the user to see what the iView will look like prior to being deployed. The actual deployment process is a one-click exercise that is performed entirely within Visual Composer. The Deployer Tasks panel collects the necessary information to process the deployment.

IBM’s **WebSphere Portlet Application Integrator (WPAI)** is a portlet-based tool provided within WebSphere Portal that is used to accomplish the same purpose – all with no coding required. While it does not use the visual drag-and-drop development metaphor, it does utilize easy-to-follow wizards to lead the user through the portlet creation process. One of the distinguishing features of WPAI is its generic approach to Application Integration. Application-specific adapters can be built and easily plugged into the WPAI framework. The ones that ship with the product include support for SAP, Siebel, PeopleSoft, JDBC-based data sources, and Domino databases.

The two major components of WPAI are the Portlet Builder and the Business Object Editor, both of which are portlets themselves (Figure 6).



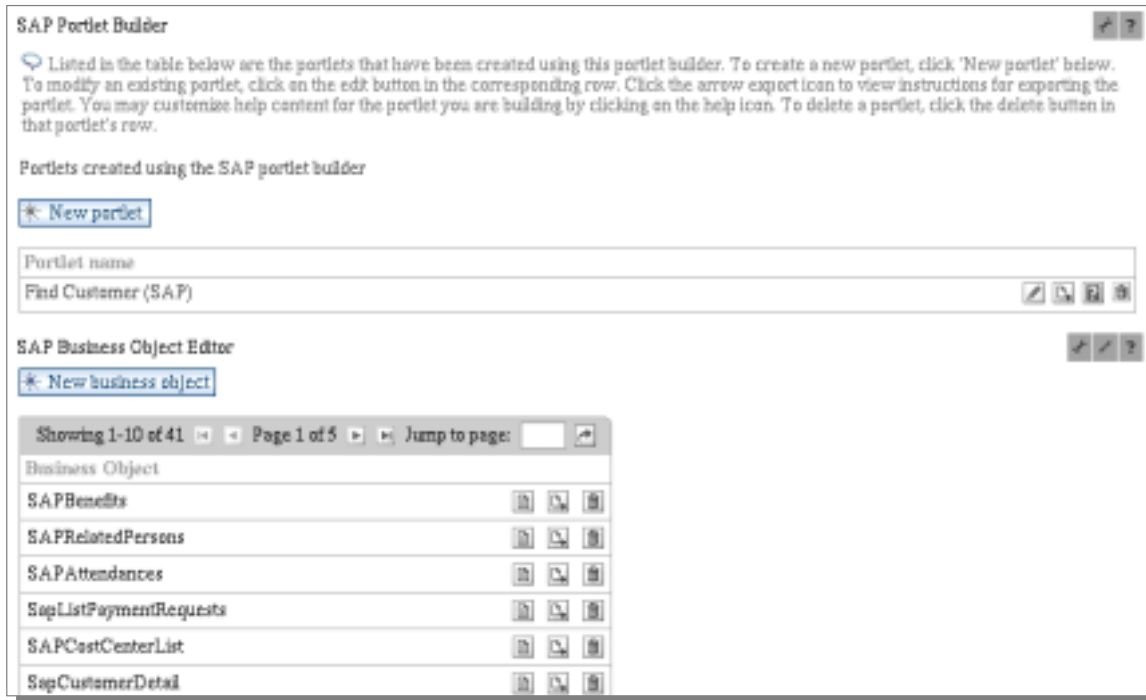


Figure 6. IBM WebSphere Portal Application Builder: SAP Portlet Builder and Business Object Editor

For this particular exercise, we used the SAP versions of both portlets. The steps to create an SAP portlet in WPAI are depicted in Figure 7 and are described below:

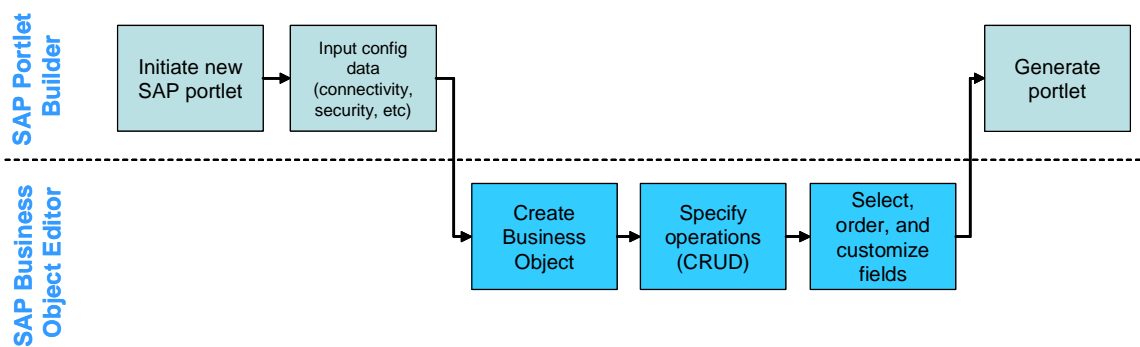


Figure 7. Steps for Building an SAP Portlet Using the SAP Portlet Builder and Business Object Mapping Tool

1. The Portlet Builder wizard starts by collecting information about the target SAP host, connection parameters, and security credentials.
2. In order to select the appropriate SAP application fields for display in the portlet, a new business object is first created with the Business Object mapping tool that is included with WPAI. The Business Object mapping tool allows the user to first search and display BAPIs defined for a given SAP server. It also introspects and displays specific configuration information and parameters unique to the SAP BAPI selected by the user. Using this data, the user specifies the operations this particular Business Object should support (Search, Update, Create, Delete) and then saves it. (see Figure 8 for example).

- The user proceeds to use the SAP Builder and selects defined business object created with the SAP Business Object mapping tool. Next, the user specifies the output markup (HTML, cHTML, WML) and other options, including Click-to-Action to enable inter-portlet data exchange, SAP Table display, people awareness, image links and other options that may be desired in the portlet application. This completes the creation of the Business Object.
- The Portlet Builder is then called upon to wrap the resulting Business Object in a portlet wrapper and deploy it to the portal server

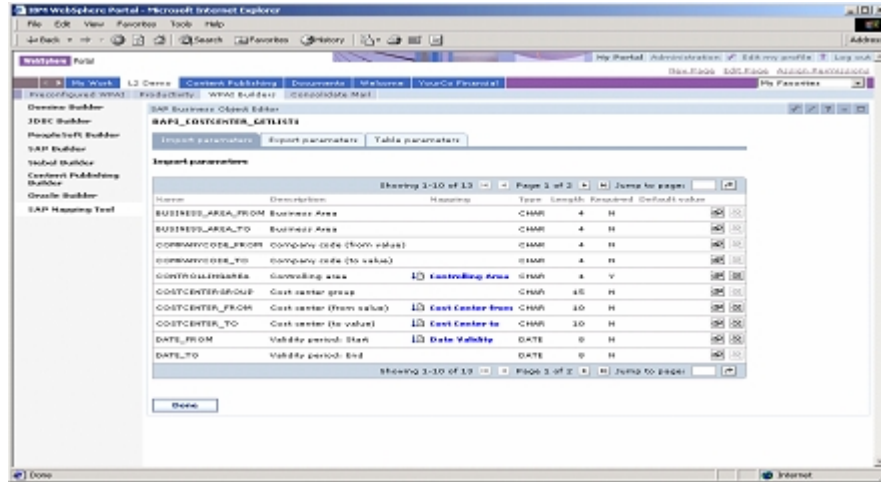


Figure 8. WPAI SAP Business Object Mapping Tool

Although both SAP Visual Composer and IBM WebSphere Portlet Application Integrator are aimed at the business-level user with little to no coding experience, their implementations are very different and make a direct comparison between the two difficult. Nevertheless, a table outlining their similarities and their differences appears in Figure 9. The net effect, however, is the same: both products produce portlets that can access back-end SAP applications and data.

	IBM WebSphere Application Integrator (WPAI)	SAP Visual Composer
Developer license required	No	Yes
Visual tooling	Yes (wizard-based)	Yes (drag-and-drop)
Portal/portlet-based tooling	Yes	No
Multiple operations in a single portlet (e.g. create, update, delete)	Yes (via Business Objects)	Yes (sequentially only)
Introspection of SAP back-end	Yes	Yes
Work with customized SAP system	Yes (with modification of XML mapping files)	Yes
Customize input, output forms	Limited (output form only)	Yes
Built-in support for inter-portlet communications	Yes	No
Extensible through programming	No	No
Built-in support for non-SAP systems	Yes (PeopleSoft, Siebel, JDBC, Lotus Domino)	Limited (JDBC, Web services)

Figure 9. IBM WebSphere Portlet Application Integration and SAP Visual Composer Comparison Matrix

Portlet Factory provides perhaps the most powerful user-driven application integration option. This product was originally developed by Bowstreet, Inc. and introduced as Bowstreet Portlet Factory in 2002. Bowstreet was acquired by IBM on December 20, 2005 and the product has been re-branded as **WebSphere Portlet Factory** as of March 2006. Portlet Factory is a framework and a set of tools for rapidly creating and maintaining customized portlets. Developers can rapidly build portlets by pulling together a sequence of highly adaptive, reusable software components called Builders. Users assemble these Builders into Models, similar to the way in which they would build a spreadsheet Model by snapping together formulas. These Models are executed at runtime to dynamically generate application code, including JSP's, Java classes, and XML documents, as well as all of the low-level artifacts that make up the portlet application. In this way, developers can capture and automate the process of building dynamic portlets instead of explicitly coding each portlet. Furthermore, developers can easily and quickly create multiple highly customized portlets from one code base, without requiring additional code changes or redevelopment.

WebSphere Portlet Factory includes an easy-to-use graphical tool (Designer) for creating, viewing, and running portlets. The Designer plugs seamlessly into the IBM Rational Application Developer 6 development tool as well as the open source Eclipse IDE.

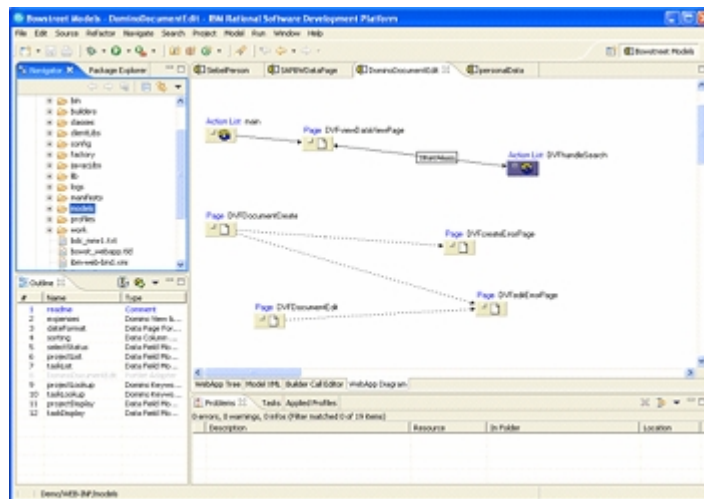


Figure 11. WebSphere Portlet Factory – Designer Tool

WebSphere Portlet Factory combines several key objects to create a portlet application:

- **Model:** A Model is the container that holds the ordered list of Builder calls. Typically, when a developer wants to create a new portlet, they will create a new Model and then add the appropriate Builders to the Model.
- **Builder:** Builders are software automation components that capture design intelligence and automate the creation of code. Similar to customizable robots in an assembly line, Builders perform specific software automation tasks based upon inputs or parameters specified by developers. Portlet Factory includes over 120 Builders that automate a wide range of tasks, such as creating HTML from a schema or integrating with common back-end systems (like Domino, SAP, Siebel, databases, etc.). Builders have easy-to-use, wizard-like user interfaces, which make it both fast and easy to develop portlets. They are much more powerful than wizards, however, since Builders can be used iteratively throughout the entire development process. Design flexibility and improved productivity is provided as portlet builders can easily change a Builder's input values and have the entire portlet application update instantly. Behind the scenes, a Builder is made up of a Java class that performs the appropriate automation task (like creating the JSP for the button) and an XML document that defines the Builder's characteristics.

- **Profile:** A profile contains a set of parameters that vary the way an application behaves. A profile feeds values into Builders based on user identity or other contextual information (such as language). Using profiles, different variations of a portlet can be automatically generated (from the same model) for different users, customers, or situations.
- **Regeneration:** When a Model is regenerated, each Builder in the Model executes in sequence and assembles components of the resulting portlet, such as JSP Pages or Java methods. During regeneration, profiles can feed different inputs to Builders based on the user or situation, automatically creating custom portlets “on the fly”. Regeneration is similar to the process of “recalculating” a spreadsheet, the difference being that Portlet Factory regenerates the portlet code, rather than a set of numbers.
- **WebApp:** The WebApp is a profile-specific instance of a portlet application that is dynamically created by the Portlet Factory regeneration engine. Each Builder, when called during regeneration, creates the artifacts that make up this WebApp or run-time portlet application, such as pages, forms, variables, Java Objects, and methods. The regeneration engine creates the WebApp by regenerating a Model with a unique instance of profile data. The generated WebApp code is then processed by the Factory’s execution engine to instantiate the executable J2EE application session(s).
- **Application server:** Portlet Factory leverages the application server's HTTP stack as well as all of the services from the application server, such as clustering, failover, J2EE security and session management.
- **Portal servers:** Portlets created with the Designer automatically plug into WebSphere Portal as well as any JSR 168 compliant portal server. Bowstreet Portlet Factory integrates with Portals via the Bowstreet Portlet Adapter. This is a standard Portlet WAR that includes the Portlet Factory classes (JAR files). When a request comes in from the portal, the Bowstreet Portlet Adapter invokes the Portlet Factory code at the layer below the servlet layer (the WebAppRunner class).

WebSphere Portlet Factory provides out of the box builders to quickly create portlets for SAP applications without programming. Builders specific to SAP applications include:

**SAP Function Call** - Establishes a call to a remote-enabled SAP function and creates a Java method that can be called to get data from the SAP system. This Builder also creates Model variables that contain the data returned from the function. (This Builder’s functionality is incorporated into the SAP View & Form builder.)

**SAP Help Values** - Provides access to the “Help Values” associated with the fields present in an SAP function. Help Values are a list of valid choices for a given function field. This Builder can be used to populate a Bowstreet Select input control from which a user can select an input to the SAP function.

**SAP View & Form** – Creates a Search Page (input form), Results Page and, for functions that have tabular data for the results, a Detail Page from a remote-enabled SAP function call (see Figure 10). Each of these pages can interact with the selected SAP function.

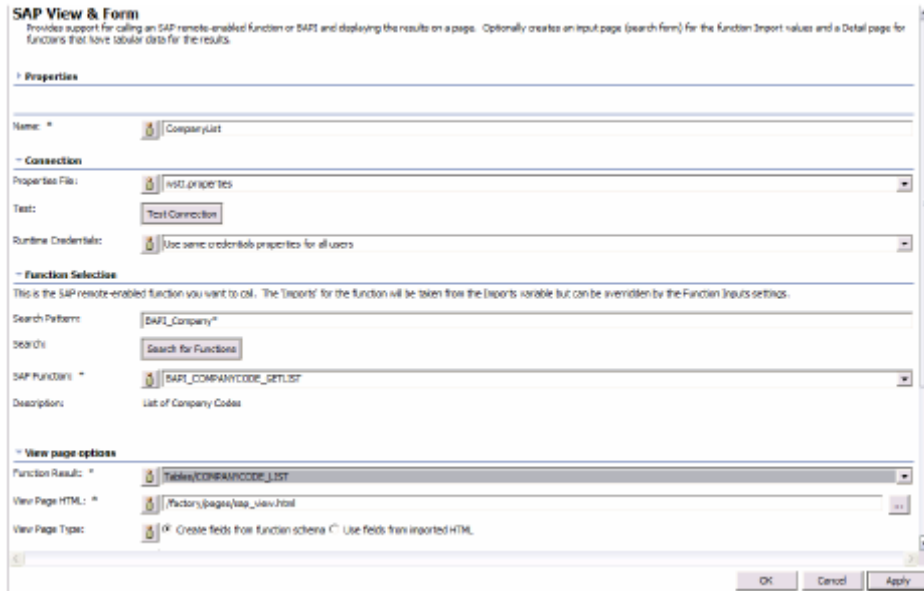


Figure 10. View and Form Builder, WebSphere Portlet Factory

**SAP Batch Builder** - Executes an exported SAP batch input. Using features in the Builder you can parameterize values of the batch process. Additionally, values of the batch can be designated User Inputs.

The following steps are completed to generate a portlet to an SAP application using WebSphere Portlet Factory:

1. Create a Model by launching the new Model wizard and choose a wizard to work with. There are several data integration wizards available that represent the most common portlet types. These wizards automate the Model creation process by adding the necessary Builders for you. An empty Model wizard is provided to permit users to manually add Builders. From the Bowstreet perspective within Eclipse or Rational Application Developer 6, create a new, empty Model and give it a name.
2. From the Builder Palette, select the SAP Function Call Builder.
3. Populate the Builder's inputs by selecting the appropriate parameters, input fields, and desired operations (Create, Read, Update, Delete). Then select the operation or method to call, specify the desired paging behavior, and then apply an appropriate stylesheet.
4. Save and close the Builder call by clicking OK.
5. Save the Model by clicking the 'Save' icon to complete the portlet creation steps.

To incorporate more functionality, the user would simply repeat the process using other builders and link them together using the wizard-like dialog panel.

Overall, WebSphere Portlet Factory eases the entire portlet development process by enabling developers to create, test, debug, run, and modify portlets as fully functioning standalone web applications - outside of the portal framework. This ability to run portlets standalone has huge benefits, since developing and debugging in a standalone setting is both efficient and familiar for developers.

When compared against SAP Visual Composer, WebSphere Portlet Factory offers users far greater flexibility and functionality for building SAP portlets without the user having write any code.

## Developer-Driven

For maximum control and flexibility in building “native” portlets, both vendors provide toolsets aimed at the professional developer. These portlets are typically built using a “**tops-down**” approach, where the developer extends a generic portlet class or object and adds the appropriate presentation, business, and data access logic. After testing and debugging, the resulting files are then packaged up and deployed to the production portal server.

**SAP** provides **NetWeaver Developer Studio (NWDS – Java)** and the **Portlet Development Kit** in order to create these “tops-down” portlets. The generic portlet class, however, adheres to a proprietary API and is not compliant with the emerging industry standard JSR 168 Portlet API. In addition, the development process is largely a manual one due to the lack of visual editors to help minimize the amount of handcrafted code that must be written. **IBM Rational Application Developer (RAD) v6**, on the other hand, utilizes various wizards and visual drag-and-drop style editors to minimize the amount of hand-written code. RAD v6 includes an SAP back-end adapter, as well as a set of SAP-specific Service Data Objects (SDO) from which to accelerate the process of building SAP portlets. When combined with the JavaServer Faces (JSF) standard for presentation development, quantum leaps in developer productivity can be expected compared to more manually driven tools. The differences in productivity can be seen in a sample portlet application that was developed using both sets of tools. The scenario involved developing an input form requesting a Customer number, which upon clicking on the Submit button would retrieve details about that customer from the back-end SAP application and provide the response in an output table. The results from this productivity study appear below (Figure 10).



	IBM Rational Application Developer v6		SAP NetWeaver Developer Studio (Java)	
	Tool/Technology	Time	Tool/Technology	Time
Application Set-Up - Define SAP R/3 system	RAD Config. Tools	10 min	SAP System Landscape Dir	2 min
Create Application - Create portal app project	Portal wizard	1 min	Portal wizard	1.5 min
Create Proxy for Back-End Access	SDO wizard	3.5 min	wizard	4 min
Code Application	JSF + SDO (drag-n-drop)	10 min	JSPDynPage + JSPs (manual)	48 min
Make Application Available to Portal	RAD Deploy. Tools	2 min	SAP Content Studio	4.5 min
<b>TOTAL TIME</b>	<b>26.5 min</b>		<b>60 min</b>	

Figure 10. IBM and SAP “Tops-Down” Portlet (iView) Development Productivity Comparison

As one can see, the task of writing the actual application logic (presentation, business, and data access) is significantly faster (~5X) using IBM RAD v6 than SAP NWDS (Java) while the overall productivity advantage is a little more than 2X.

**SAP** prefers, indeed, recommends that customers use the newer, non-standard **Web Dynpro** development environment (also packaged with the Eclipse-based NetWeaver Developer Studio) and a “**bottoms-up**” approach to building iViews for running in SAP EP. “Bottoms-up” refers to first developing a Web application irrespective of whether or not it will be used in a portal, then turning the application into an iView (portlet) “after-the-fact”. In SAP’s case, this is done using the Content Studio tool just like before when building User Interface integration iViews. Thus, the User Interface of the Web Dynpro application is simply captured in an iFrame-based iView like

any other application. Ordinarily, this would limit the corresponding iView's ability to leverage all of the portal services available to it. Because Web Dynpro applications are compiled into J2EE artifacts, they can run on the same server as the portal. Thus, services like iView-to-iView messaging are still accessible and can be leveraged accordingly. Other issues, however, such as the need to scroll from top-to-bottom, or side-to-side within the iView, remain unless careful layout planning is done in the initial design phase.

Web Dynpro provides a higher level of abstraction on top of the J2EE development model. It consists of a build-time and a runtime environment, both of which are proprietary. It closely follows the classical Model-View-Controller programming paradigm and has a far richer set of visual tools and editors that improve developer productivity when compared to the standard NetWeaver Developer Studio (Java) toolset. When the same portlet application described above is built using Web Dynpro, the task of writing the presentation, business, and data access logic was roughly equivalent to that of IBM Rational Application Developer v6 (Figure 11).

	IBM RAD v6		SAP NWDS (Java)		SAP NWDS (Web Dynpro)	
	Tool/Technology	Time	Tool/Technology	Time	Tool/Technology	Time
Application Set-Up - Define SAP R/3 system	RAD Config Tools	10 min	System Land. Dir.	2 min	System Land. Dir.	2 min
Create Application - Create portal app proj.	RAD Portal wizard	1 min	Portal wizard	1.5 min	Wizard	1 min
Create Proxy for Back-End Access	RAD SDO wizard	3.5 min	wizard	4 min	Wizard	1.5 min
Code Application	JSF + SDO (drag/drop)	10 min	JSPDynPage + JSP (Manual)	48 min	Wizards + visual tools	8.5 min
Make Application Available to Portal	RAD Deploy Tools	2 min	SAP Content Studio	4.5 min	Web Dynpro admin console, SAP Content Studio	6.5 min
<b>TOTAL TIME</b>	<b>26.5 min</b>		<b>60 min</b>		<b>19.5 min</b>	

Figure 11. IBM and SAP Portlet Development Productivity Comparison

When it comes to doing Application-Level integration with existing back-end SAP applications (R/3 or mySAP), IBM and SAP both offer an array of options. Although differing in their implementations, they achieve the same objective of enabling end-users to access SAP data via a portal. IBM has chosen to adopt a more standards-driven path, with its support for JSR 168 (Portlet API), Web Services Remote Portlets (WSRP), JSR 170 (Content Repository API), JavaServer Faces (JSF), Service Data Objects (SDO), and more. SAP, on the other hand, has bet their hand on leveraging proprietary technologies like Web Dynpro, the iView API, content repository API, etc. in order to garner similar results.

## Pre-Built Portlets and iViews

Many customers prefer to 'buy' rather than 'build' portlets that integrate with SAP applications and data. For them, solutions like SAP Business Packages can be an attractive alternative to custom portlet or iView development. As long as customers are willing to use these collections of iViews "as-is" and without modification, they can be an acceptable solution. In this vein, SAP EP essentially becomes just another User Interface for SAP applications, a trend that is expected to increase in the future as SAP evolves their back-end application suite around the NetWeaver platform. As such, SAP EP will take on more of an integrated UI role for SAP applications than a true enterprise portal with broad support for heterogeneous systems and data. IBM WebSphere Portal, for its part, can run these same iViews, as well as offers more comprehensive support for third party applications, data, and other content. Customers are encouraged to compare the [IBM Workplace Solutions Catalog](#)<sup>1</sup> against SAP's Portal

Content Portfolio (part of the SAP Developer Network subscription offering) to see first-hand the difference in non-SAP integration solutions available.

For companies needing to modify the iViews in current Business Packages so that they more accurately reflect the needs of the business, this can be a two-edged sword. Some iViews may not have their original source code available to import back into NetWeaver Developer Studio for customization. Others may have been written around a spectrum of different programming artifacts and technologies, forcing developers to learn multiple toolsets in order to make the necessary changes. Thus, customers will need to balance carefully the overall value they receive in the form of pre-built iViews vs. the time, effort, and skills required to make them more suitable to their business requirements. In many instances, it may be more worthwhile to write new portlets or iViews from scratch.

**Bowstreet's Corporate Performance Suite** provides IBM WebSphere Portal customers with another option for pre-built portlets fashioned around specific business solutions. In addition to providing pre-packaged Executive, Sales, and Operations-oriented “dashboards”, Bowstreet also provides a dashboard framework for building custom dashboards or extending the pre-packaged versions. The advantage to this approach is that it permits rapid customization of portlets and dashboards to more closely align to the needs of the business instead of the “fixed-function” nature of SAP iView Business Packages.

## Summary

The challenge to integrate back-end SAP applications into an enterprise portal is not one to be taken lightly. For obvious reasons, many customers have the perception that SAP Enterprise Portal is better equipped and offers tighter integration with an SAP R/3 or mySAP environment than any other vendor offers. Our own hands-on evaluation and comparison against IBM WebSphere Portal suggests strongly that such is not the case. SAP relies heavily on the use of iFrame technology to achieve User Interface integration with their back-end applications. IBM WebSphere Portal uses the iFrame-based approach as well. Both vendors offer tools for the Business User as well as Professional Developer for building native portlets. Though their implementations differ, the end results are portlets and iViews that access and render the desired R/3 and mySAP application data in a portal. All things being equal in terms of back-end SAP integration, customers are advised to look at other critical enterprise portal requirements such as broad support for accessing non-SAP applications and data, as well as portal and collaboration standards before making a decision. Other factors like performance/scalability, virtual portal support, user process integration support, along with portal content management and personalization are areas where IBM also excels relative to SAP Enterprise Portal.



## References

---

<sup>1</sup> IBM Workplace Solutions Catalog (<http://catalog.lotus.com/wps/portal/workplace>)