

Technical Note

*MANAGE, INTEGRATE, AND PUBLISH YOUR
ENTERPRISE CONTENT INTO YOUR PORTAL*

August 2004

Manage, Integrate, and Publish your Enterprise Content into your Portal

THE NEED FOR A COMPLETE SOLUTION

In most organizations, important business content is spread across a wide variety of content management and storage systems, including file systems, databases, content management repositories, Web sites, email accounts, and applications. Organizations must make this content available throughout the enterprise, to enable better decision making, to inform employees about changes in policies and procedures, to comply with government and industry regulations and quality controls, to share best practices, to avoid reinventing content that already exists, and to prevent unnecessary duplication of content and the potential for outdated copies. At the same time, much of the content must be subjected to rigorous controls to prevent unauthorized access.

But there are many obstacles to unlocking and controlling access to the rich, untapped store of business content. These obstacles include: content management systems that are proprietary, difficult to use, or expensive to license across the enterprise; reliance on imprecise searches to access unstructured content on file systems and Web sites; Web publishing practices that rely on specialists to mark up and deploy content to corporate Web sites and portals; and inconsistent end-user interfaces to the content, where each content source has its own access tools and presentation format.

Early generations of enterprise portals attempted to resolve these problems by providing portlets that acted as "windows" on external content sources. However, this approach has had its own set of problems: each portlet accessing a different content source would offer a different user experience for selecting and presenting content. And, while it has been common practice for content management systems to provide portlets that expose their administrative functions (e.g., workflow, content submission), these systems typically don't offer portlets for easily publishing links to content or displaying the content itself. Even a simple task such as publishing an announcement to a department home page can require a complex set of interactions between the CM system and the portal, often involving custom development and considerable administrative effort. Finally, in many portal products portlets cannot communicate with each other or with the page on which they were placed. So although several portlets could be placed on a page to pull information from different sources, those portlets could not respond to a single selection or customization that would ensure that each portlet displays related information.

Let's look at an example: an IT department would like to publish a page about computer security on the corporate portal. The page will contain announcements about new security threats, links to company policies and procedures on computer security, a list of active computer viruses, and a portlet for getting information on a specific virus. The department also wants end-users to be able to customize the information, for example, to select viruses that are applicable to their specific desktop operating system. Using the company's current portal product, the page would have to be constructed as follows:

- First, a developer must create a Java Server Page that will serve as the skeleton for the computer security page.
- The portal product does not provide a self-service application for publishing text-based content, like announcements. The announcements must be written by the IT department, then handed off to a Web Designer in Corporate Communications who puts the text into an HTML file. The HTML file is submitted to the company's content management system, where it goes through an automated approval process before being deployed to the production file system for the portal. A portal developer must construct a portlet that can read the file, and the portal administrator must add the portlet to the computer security page so the announcements can be displayed.
- The company policies and procedures are managed by the HR department and stored in the company's content management system. However, because of licensing costs, the content management system can only be accessed by a small number of users. To publish the content to a broad audience, the content must be extracted from the content management system and loaded on a public file system. Again, a portlet must be written that displays links to the files to be published. The IT department has written a portlet that can list the contents of a file system directory. The portal administrator must add this portlet to the computer security page and set its properties to display the computer security folder. More advanced selection of content is not possible because the metadata associated with the files in the CMS cannot be stored in the file system. For example, the portlet cannot be set up to display files for a particular author or effective date without additional work to extract and store the metadata outside of the CMS. It is also difficult to place access controls on the documents.
- The list of new viruses is available as a syndicated XML feed from the company's supplier of virus protection software. A custom portlet must be developed to convert the XML feed into HTML that can be displayed on the page and that conforms to the corporate style defined for the portal. Since the XML feed also contains information on products that the company has not licensed, the XML must be parsed to filter out anything but the virus information. Finally, a customization interface must be developed for the portlet so that end-users can select the virus information that is relevant to their desktop operating system (e.g., select viruses specific to various versions of Windows, Mac, and Linux desktops).
- To show detailed information about a specific virus, the IT department would like to automatically search the Internet for information about a virus when the user clicks on the virus name in the virus list portlet. However, the search portlets available from their enterprise search vendor cannot communicate with the other portlet to get the virus name, and the portal vendor does not support inter-portlet communication even for custom-developed portlets. Furthermore, the search portlets only allow users to submit searches – they don't actually show the search results (the user is taken to another page containing the results). Rather than construct another custom portlet, the IT department decides not to implement this feature in their portal, even though it means that less information will be available to users about potentially hazardous viruses.

This example illustrates that some level of content integration is possible with many portal products. But that integration comes at a high price, requiring custom development and cumbersome administrative processes. Since this is a fairly typical scenario for a portal, the portal product could make life much easier by:

- Providing a built-in, self-service content management application for publishing content directly to the portal

- Enabling content managed in an external content management system to be easily deployed to the portal for publishing
- Allowing content that is already published or available in an external source to be easily accessed through the portal. The end-user experience of selecting, filtering, and presenting the content should be consistent, regardless of the underlying content source
- Supporting inter-portlet communication to improve the relevancy of published content by making portlets automatically respond to user interactions

This paper will discuss how Oracle Application Server (OracleAS) Portal satisfies these requirements for managing, integrating, and publishing enterprise content. The following section briefly describes the content management features available within OracleAS Portal. The remainder of the paper discusses the technologies and features that can be used to integrate any content into your portal, no matter where it is managed or where it is stored.

MANAGING CONTENT WITH ORACLEAS PORTAL

For sites that need a single application for managing and publishing content, OracleAS Portal has a built-in, complete set of features for both document and Web content management. With these self-contained features, OracleAS Portal enables users to create, maintain, publish, and search business content without having to leave the portal for an external application. The content management features can also be used to enhance the portal's appearance and user interaction, and to save on infrastructure costs by consolidating multiple corporate Web sites into a single, integrated portal.

The content management features in OracleAS Portal promote document sharing, collaboration, and process automation with built-in controls, such as versioning, check in/check out, automated publishing and expiry, and multi-step approval routing. In fact, any type of content, including content metadata, can be managed in the extensible Portal repository. Content and metadata can be authored in the browser, and support for the Web Distributed Authoring and Versioning (WebDAV) protocol allows desktop applications to interact directly with content stored in the repository.

Hundreds of organizations have used OracleAS Portal's unique and flexible page design features to build their public and customer-facing sites. Because content is not an afterthought in OracleAS Portal, its content management features allow users to create content-rich portal pages that conform to corporate style standards. This is an essential feature for building externally facing portals that are directed at customers, suppliers, and partners, and for consolidating existing Web sites.

With content management built right in, it is also easier to leverage the complete portal framework offered by OracleAS Portal to improve the relevance, accessibility, timeliness, and accuracy of content. Content can be maintained and accessed directly in the context of enterprise applications that are integrated into the portal. Page hierarchies, single sign-on, personalization, search, mobile device support, subscriptions, notifications, collaboration, and globalization make content easier to access. And many of the management and performance features offered by the portal framework are also available to its content management system – features like user provisioning and access control, caching, and usage monitoring.

In our example, the IT department can use OracleAS Portal's content management features to:

- Design and build the computer security page, without requiring that a developer create a custom Java Server Page. The page design capabilities are available through a wizard-driven, browser-based application that requires no knowledge of HTML.
- Author and manage the announcements directly on the page. WYSIWYG authoring of HTML is also accomplished through browser-based wizards, and again no knowledge of HTML is required to create richly-formatted content. Publication of the announcements can be controlled via a multi-step approval process, granular access control, and automated publish and expiry dates.
- Upload and manage the policy and procedure documents into a document library in the portal repository. The documents can be attributed with company-specific metadata to enable precise searching and the query-based publishing capabilities described later in this paper

Overall, having a single application for both managing and publishing content reduces cost and complexity when compared to portal solutions that require integration with third-party content management systems.

INTEGRATING CONTENT WITH ORACLEAS PORTAL

While Portal's content management features can be used to meet many content management needs, there are still a number of reasons why content may be managed elsewhere:

- An organization deploying OracleAS Portal may already be managing content successfully in an independent content management system.
- The organization may have specialized content management needs that are better addressed by another product, such as imaging, catalog management, or compliance with regulatory standards (ISO, FDA, etc.).
- The organization needs to separate the management and the publishing of content to allow the content to be repurposed for multiple applications.

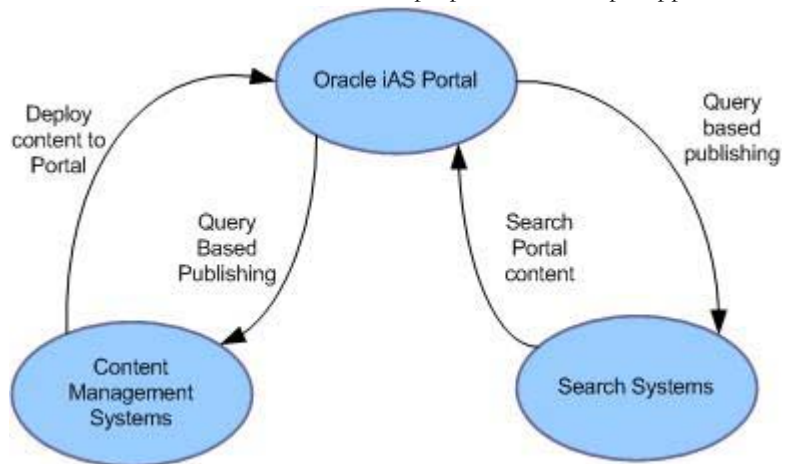


Figure 1 - High Level Flow for Content Integration

As shown in Figure 1, a variety of approaches can be used to integrate content into OracleAS Portal for publishing, regardless of where that content is managed and stored. Conventional search systems can also be used to locate content and publish it directly to a portal, without requiring portal users to define and submit their own search criteria.

The content integration approaches used in OracleAS Portal can be grouped into two types: *Managed Deployment* and *Query-Based Publishing*.

MANAGED DEPLOYMENT

Managed deployment refers to the process of physically copying content from an external source into the Portal repository. From the repository, the content can be published using OracleAS Portal's built-in publishing features.

To support managed deployment, OracleAS Portal users can take advantage of *WebDAV*, and the *Portal Content APIs*.

WebDAV

Web Distributed Authoring and Versioning (WebDAV) is a standards-based protocol for accessing remote content repositories over HTTP. File content can be copied directly to and from the Portal repository through a WebDAV desktop client, such as Microsoft Office, Windows Web Folders, Macromedia Dreamweaver, and Adobe Photoshop, or command line tools like *sitcopy* and *cadaver* (both Open Source).

Portal Content APIs

OracleAS Portal provides a set of PL/SQL APIs for programmatic loading of content to the repository. The APIs enable the building of applications that take advantage of

advanced features of the Portal repository, including non-file content types like text and URLs, extended metadata, translations, and access control.

Managed Deployment Examples

In our example, the IT department could use one or more of these managed deployment features to load content from an external source into the Portal repository. For example:

- A custom application can be used to capture the XML feed and create Portal content items. While this involves some custom programming, the developer is only concerned with the relatively simple task of loading the content into the Portal repository. Business users handle filtering and publishing the content through built-in Portal publishing features.
- Content might be prepared locally or stored in a content management system, but for publishing it might need to be moved to OracleAS Portal. This could be done as described above using custom code against the APIs, but it could also be performed using WebDAV as the transport protocol. Content could be deployed using simple, standards-compliant tools that support WebDAV.

QUERY BASED PUBLISHING

OracleAS Portal's query-based publishing capabilities allow content to be published in the portal directly from a content source. The source may be content in the Portal repository, but will often be content that is stored in an external system. Query-based publishing enables dynamic presentation of content, where the content that is presented in the portal can vary based on end-user preferences or personalization rules. It also ensures that information is always up-to-date, as the content is dynamically queried directly from the source instead of being deployed to an intermediate storage location.

Query-based publishing is achieved with the *Custom Search Portlet*, *OmniPortlet*, and *custom portlets*.

Custom Search Portlet

The *Custom Search Portlet* assists page designers with creating automatic queries of content managed within Portal. Following a wizard-driven interface (see Figure 2), the page designer specifies the page groups that are to be searched and the search criteria, consisting of attribute and keyword values. The page designer can also define the layout and ordering of the search results – the results can be configured to look much like an item region on a Portal page. Finally, the page designer indicates that the search is to be executed and the results displayed automatically when the portlet is viewed. When end users view the page containing the portlet, they see the results of the query as if that content actually resides on the page.

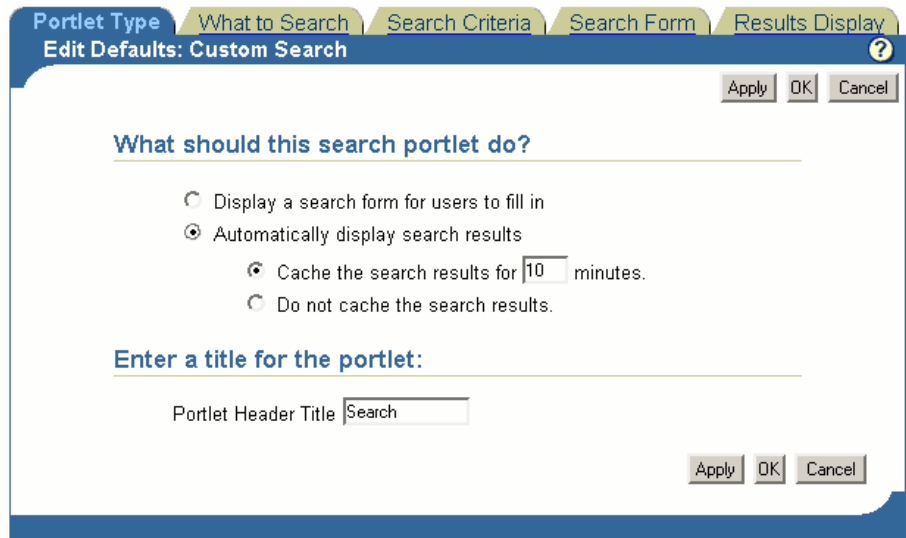


Figure 2 - Custom Search Portlet

The custom search portlet enables you to separate the management and publishing of your portal content. Content can be stored in one page group that is configured for ease of use by users who create and manage content by defining the navigation, layout, security, and behavior that meets their specific needs. For example, the page hierarchy of the page group can be designed to organize the content by functional teams of content contributors. In addition, the layout of the item regions can emphasize the attributes that are important to managing content (e.g., publish dates, expiry dates, authorship, etc.). Finally, the availability of item types and the definition of approval processes can be tailored to the content and the users who manage it.

The content can be published in another page group that is set up for consumers or viewers of the content. The custom search portlet (along with other query-based publishing portlets) is used to aggregate related content from multiple management pages. The portlet can display those attributes that are meaningful to the viewers while hiding attributes that are only of interest to content managers.

OmniPortlet

The OracleAS Portal *OmniPortlet* is a declarative tool used by page designers to create dynamic content publishing portlets. The OmniPortlet framework allows content to be queried from disparate sources through a common interface and published using a set of standardized layouts. The user experience for publishing content with OmniPortlet is consistent, regardless of whether the content is coming from a relational database, XML document, Web service, or other source.

Custom Portlets

The declarative approach found in the custom search portlet and OmniPortlet can offer a great deal of flexibility, but some content publishing requirements may require custom portlet development with the Portal Developer Kit (PDK). However, the majority of content publishing needs can be handled by business users and page designers using the declarative tools, leaving developers free to focus on exceptional or complex publishing applications.

Query-Based Publishing Examples

In our example, the IT department can use these query-based publishing features to publish their content as follows:

- If the department has opted to copy their policy and procedure documents to the Portal repository, as described under *Managed Deployment*, they can use the

custom search portlet to automatically publish any document that has a prescribed set of attributes that are relevant to the computer security page (e.g., Category = "Policy", Topic = "Security", and Department = "IT"). The documents don't have to be loaded onto the computer security page—they can reside anywhere in the repository.

- The OmniPortlet XML data source can be used to access the syndicated XML data feed on virus protection. In our example, the page designer can filter the query to return only virus protection listings (note that OmniPortlet supports filtering of the content even if the content source – in this case, an XML document – does not have a query interface). The page designer can also parameterize the query so that viewers can customize the portlet; for example, if the XML feed contains an element called "OPERATING SYSTEM", the page designer can associate this element with a customization parameter that a viewer can use to restrict the list to viruses that affect the desktop operating system that he or she uses.

The page designer can also set up the OmniPortlet to raise an event when a virus name is clicked in the result list. The event can pass the name of the virus to another portlet on the page, via a page parameter. This second portlet can be another instance of OmniPortlet that queries the same XML feed, but displays more detail about the individual virus selected in the parameter rather than a summary list of all viruses. The second portlet could also be an OmniPortlet that queries another content source, or a portlet provided by a search vendor that searches the Internet for news about the virus. The important thing to note is that portlets from dissimilar sources can communicate with each other via OracleAS Portal's parameter and event feature.

Related Information

Please visit OracleAS Portal Center at <http://portalcenter.oracle.com> for in-depth information on all aspects of OracleAS Portal, including all the resources listed here.

The Web Publishing and Content Management page (<http://portalcenter.oracle.com/publishing>) on OracleAS Portal Center contains a variety of articles on the content management, integration, and publishing capabilities of OracleAS Portal.



{Title - Set in Document Properties}
August 2004

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 © 2004, 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.