Rhythmyx

Connector for BEA WebLogic 8.1 Portal

Version 5.7

Printed on 19 October, 2004

Licenses and Source Code

Other Copyrights

This document was created using ***AuthorIT™, Total Document Creation*** (see AuthorIT Home - http://www.author-it.com).

Percussion Software
600 Unicorn Park Drive
Woburn, MA 01801 U.S.A
781.438.9900
Internet E-Mail: technical_support@percussion.com
Website: http://www.percussion.com

Contents

CHAPTER 1

Portal Connector Overview

The Rhythmyx Connector for BEA WebLogic Portal enables you to use Rhythmyx as your WebLogic Portal's Content Management system.

The Connector allows you to take advantage of the robust functionality of Rhythmyx Enterprise Content Management—sophisticated version management, content reuse across platforms, dynamic workflow, content computerization through Intelligent Relationships, globalization capabilities and much more—while providing:

- an alternative user interface to the CMS through Contribution Portlets;
- single sign-on using your WebLogic Portal as the Content Management authentication provider;
- de-coupled content delivery to the Portal;
- Active Assembly.

The Connector's ***Contribution Portlets*** (see "Understanding the Content Contributor UI" on page 28) ease your users' adoption of content management by allowing authorship and workflow through your enterprise Portal interfaces. These Portlets provide an easy-to-use Content Management interface for infrequent or distributed contributors, and support a lightweight or zero-client install procedure which your system's security may require. You may customize the Contribution Portlets using Rhythmyx Web Services.

On the ***delivery side*** (see "Setting up Delivery" on page 43), managed content benefits from the full advantages of enterprise class content management, and your Portal platform acquires content from the Rhythmyx Content Management system in an optimized fashion (for example, you don't want or need to manage old versions of Portal content on the Portal. Versioning is managed in the Rhythmyx repository, which is distinct from your Portal repository). Only data and metadata that the Portal needs are delivered to WebLogic, and these data are delivered to the file system and/or database repository according to the way you configure publishing.

Rhythmyx can pre-assemble output, or deliver content as fielded data that your own delivery Portlets can consume and fully manage dynamically at runtime, or use any combination of these approaches together. Without intervention from technical staff, users can publish content to the correct file system in the static tier and metadata to the correct WebLogic Portal repository for optimal delivery. As a Portal Developer, you can choose from a broad spectrum of options for delivering portal output.

NOTE: Access the BEA documentation referred to in this document at http://edocs.bea.com/ or through the WebLogic 8.1 user interfaces.

Functional Overview

The Rhythmyx Portal Connector for BEA WebLogic Portal enables you to:

- deploy the Rhythmyx business user interface and functionality into your BEA Portal environment so that you can perform Rhythmyx content contribution functions through Portlets;
- deliver Rhythmyx content into the BEA repository;
- optimize delivery of static components to static tiers and dynamic components to the Portal repository;
- use Rhythmyx Dynamic Workflow on in-process content in your portal;
- reuse content originally targeted to other delivery environments in your WebLogic Portal;
- apply Active Assembly and Rhythmyx Intelligent Relationships to content in your Portal environment.

Rhythmyx Portlets perform Rhythmyx content contributor functions such as invoking Content Editors and performing content searches. These functions allow users to access the CMS through the WebLogic Portal to create, edit, assemble, and workflow Content Items. A publisher plug-in gives users the ability to publish Rhythmyx data to the Portal repository or target schema. Using Rhythmyx Web Services and the Portal Connector source code examples you can customize the contribution and delivery functionalities of your Portal Connector to meet the specific needs of your system.

Architectural Overview

The BEA Portal Connector consists of separate components for contribution and delivery.

The contribution functionality of the Portal Connector lets users connect to Rhythmyx through WebLogic Portlets. The components installed to support the contribution functionality are:

- Rhythmyx Contributor Portal and Portlets

  Rhythmyx Portlets in the Rhythmyx Portal let users access the Content Explorer and various Content Explorer functions. See Determining Which Portlets to Use in Your System for more information about the Rhythmyx Portlets.
- Rhythmyx Web Services

  Rhythmyx Web Services allow the Portlets to make requests and receive responses from Rhythmyx. Web Services are installed with other applications on the WebLogic server. The Web Services used in the contribution functionality are Login, Content Type List, Search Request, and Call Direct. These functions are available for you to write your own Portlets.
- a sample tag library

  The sample tag library lets you add functionality to Portlets without having to code it. See ***Extending the Content Contributor Interface*** (see "Extending the Content Contributor UI" on page 35) for information about the sample tag library.
- a Web Service Helper class

  The Web Service Helper class performs functions using the Web Service APIs. See ***Extending the Content Contributor UI*** (on page 35) for information about functions in the helper class.

The delivery functionality of the Portal Connector enables a Rhythmyx publisher to deliver data from the Rhythmyx repository to the BEA Portal repository. The components installed to support the delivery functionality are:

- a custom publisher plug-in
  The custom publisher plug-in lets you use a Rhythmyx publisher to deliver data to the BEA Portal repository. For more information about using the publisher plug-in, ***see Performing and Extending Deli***very (on page 45).
- the Rhythmyx Publisher service
  The Rhythmyx Publisher service allows you to publish content from the Rhythmyx repository. You may install the Publisher service with other applications on the WebLogic server to use its capabilities as a J2EE servlet container.
- the Rhythmyx Article Portlet
  The Rhythmyx Article Portlet is a sample delivery Portlet. You may use it as a template to create your own delivery Portlet. The Article Portlet displays content and metadata delivered to the BEA Portal file system and repository. For more information about the Rhythmyx Article Portlet, see Viewing Published Content.

In addition to the contribution and publishing components, the BEA Portal Connector includes sample Rhythmyx components that perform all delivery functions that the Portal Connector supports. The sample components are installed on the Rhythmyx Server and include publishing support applications and Publisher component registrations. See Step 4 of ***Rhythmyx Setup Steps*** (on page 9) for a list of the components installed.

This document uses the Portal Connector sample components with BEA's Sample Portal to demonstrate how to install and use the Portal Connector. You may use the Sample Portal or another BEA Portal to create your own WebLogic Rhythmyx Portal.

CHAPTER 2

Before Installing the Portal Connector

The minimum recommended memory for running the BEA WebLogic 8.1 portal and the Rhythmyx server is 1 GB.

Note for Windows XP users: If you are working in WebLogic and want to work directly in Rhythmyx, or you are working directly in Rhythmyx and want to work in WebLogic, close all of your Web browsers prior to running the Rhythmyx Server or Rhythmyx may hang.

Prerequisites for Publishing to Oracle databases earlier than Oracle 9:

- in the file `<bearoot>\weblogic81\common\bin\commEnv.cmd`, modify the WebLogic server's class path by changing the statement `%WL_HOME%\server\lib\ojdbc14.jar` to `%WL_HOME%\server\lib\classes12.jar`;
- move the file ojdbc14.jar to another location (but do not remove it from your system;
- in the file `<bearoot>\weblogic81\samples\domains\portal\startWebLogic.cmd`, add the location of the file classes12.jar to the PRE_CLASSPATH.

Prerequisites for DB2 Databases:

In the file `<bearoot>\weblogic81\common\bin\commEnv.cmd`, modify the WebLogic server's class path by adding the following statements:
`C:\Progra~1\SQLLIB\java\db2java.zip;C:\Progra~1\SQLLIB\java\runtime.zip`
.

Recommendation for Users of this document:

The Rhythmyx Setup Steps will instruct you to install the archive file `<Rhythmyx root>/Samples/DeprecatedSamples.pda`; therefore, your installation should not include FastForward since its components may conflict with the archive file components.

Choosing How to Use the Portal Connector

Before you install the BEA Portal Connector, you should decide if you are going to use its components to enable content contribution in your Portal, to enable content delivery to its repository, or to perform both of these functions.

Enabling content contribution in your Portal benefits your users if they typically log in to the Portal home page to access the applications they use most frequently. Portlets displaying your Rhythmyx functions may appear alongside other Portlets that your users access daily, such as their mail inbox or Rhythmyx Portlets may appear on a separate tab. If you want to give users access to the full functionality of Rhythmyx, you may also give them the ability to log into Content Explorer accessed directly from the Rhythmyx Server. System Administrators, Implementers, and Web Masters must be able to access Content Explorer directly from the Rhythmyx Server in order to access the Publishing, Workflow, and System Administration tabs.

The following table outlines the Rhythmyx functionality available in Rhythmyx when accessed directly vs. the Rhythmyx functionality available in the BEA Contribution Portlet.

Rhythmyx Function	**Rhythmyx - Accessed directly**	**Rhythmyx - Accessed through BEA Contribution Portlets**
Edit Content	yes	yes
Dynamic Workflow	yes	yes
Preview	yes	yes
Active Assembly	yes	yes
Impact Analysis	yes	no
Folders	yes	yes (available only in the Content Explorer Portlet)
Active Assembly for Documents	yes	no
Delivery (Publishing) Administration	yes	no
Workflow Administration	yes	no
System Administration	yes	no

If your users are comfortable logging into Content Explorer directly through the Rhythmyx Server or do not use a BEA Portal as a common point for accessing their most frequently used applications, you may choose not to enable Rhythmyx in your Portal environment.

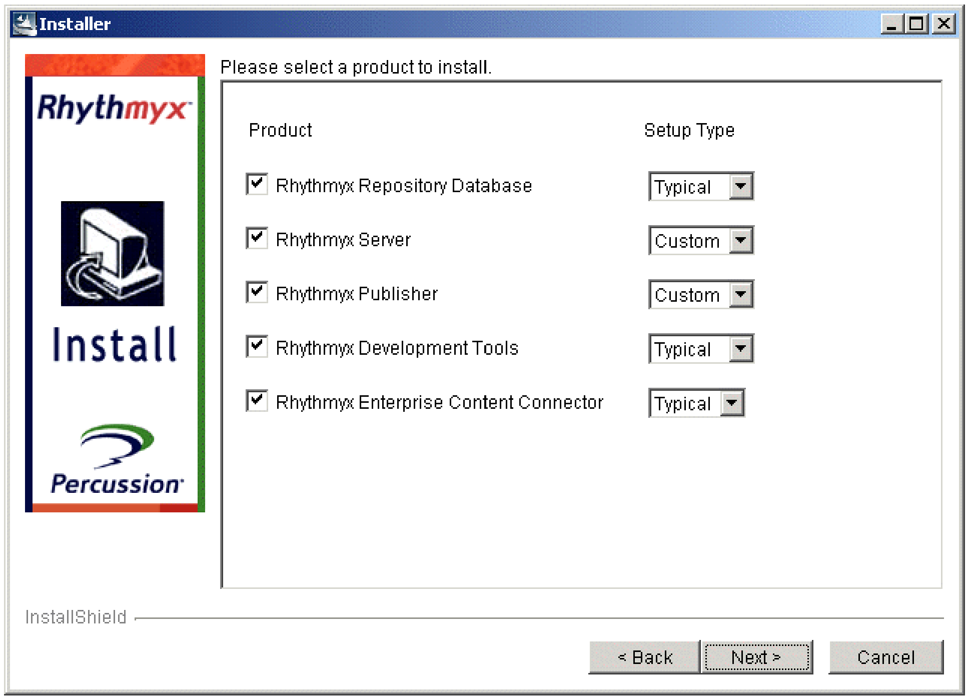
If you want to display data that you enter into Rhythmyx through a BEA Portal, you should use the delivery function of the Portal Connector to publish the required data to the Portal.

Rhythmyx Setup Steps

Setup typically involves installing a Rhythmyx server, deploying Rhythmyx application services to your WebLogic platform, and then installing the sample applications that ship with the Portal Connector. The Rhythmyx Installer will guide you through the process:

To set up Rhythmyx before installing the BEA Portal Connector:

1. Use the Rhythmyx Installer to install Rhythmyx.
2. In the Select a Product to Install dialog, in the Rhythmyx Server drop list, choose Custom.

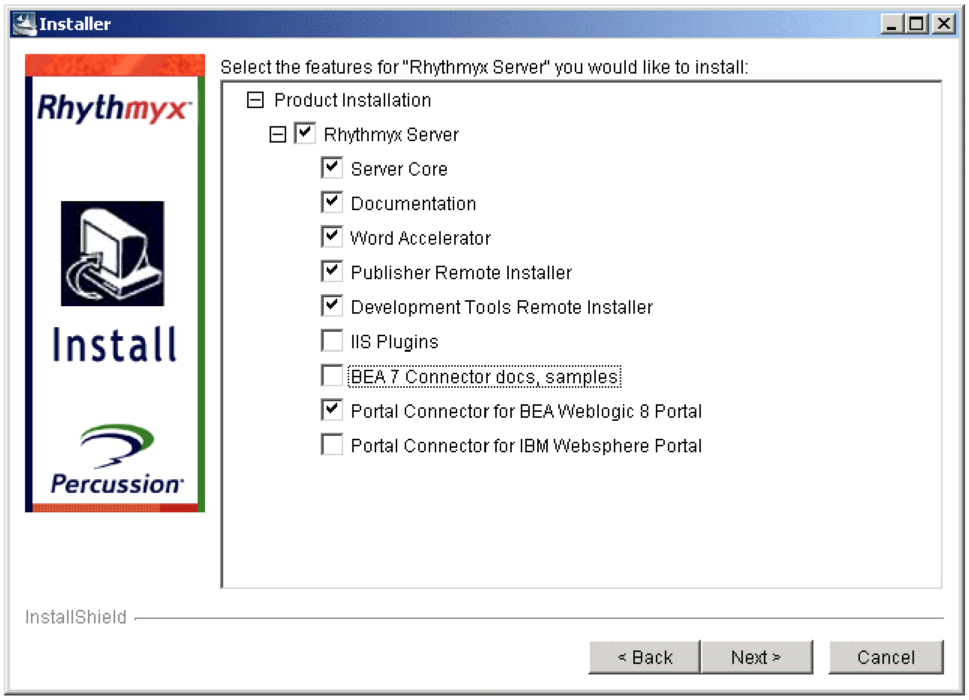


Please select a product to install.

Product	Setup Type
✓ Rhythmyx Repository Database	Typical
✓ Rhythmyx Server	Custom
✓ Rhythmyx Publisher	Custom
✓ Rhythmyx Development Tools	Typical
✓ Rhythmyx Enterprise Content Connector	Typical

Figure 1: Select a Product to Install dialog

3 In the Select the Features for Rhythmyx Server dialog, check *Portal Connector for BEA WebLogic 8 Portal* along with your other selections.

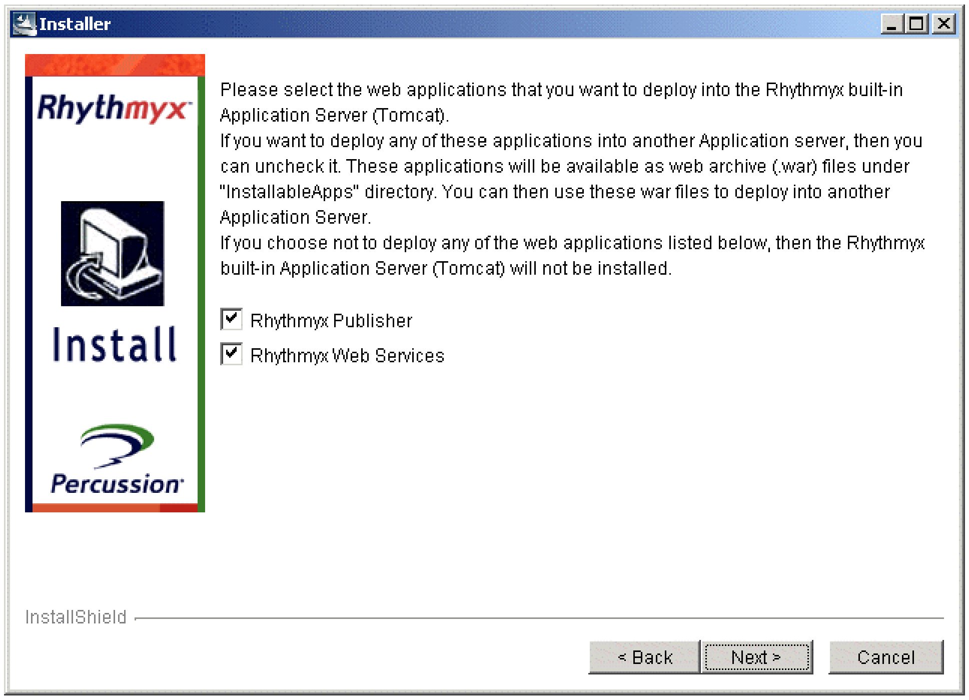


Select the features for "Rhythmyx Server" you would like to install:

- Product Installation
  - [✓] Rhythmyx Server
    - [✓] Server Core
    - [✓] Documentation
    - [✓] Word Accelerator
    - [✓] Publisher Remote Installer
    - [✓] Development Tools Remote Installer
    - [ ] IIS Plugins
    - [ ] BEA 7 Connector docs, samples
    - [✓] Portal Connector for BEA Weblogic 8 Portal
    - [ ] Portal Connector for IBM Websphere Portal

Figure 2: Select the Features for Rhythmyx Server dialog

4 In the Select the Web Applications dialog, check both Rhythmyx Publisher and Rhythmyx Web Services. Although you must deploy these services to the portal, checking them in this dialog ensures that the Installer copies all the necessary files into your Rhythmyx AppServer directory.



Please select the web applications that you want to deploy into the Rhythmyx built-in Application Server (Tomcat).
If you want to deploy any of these applications into another Application server, then you can uncheck it. These applications will be available as web archive (.war) files under "InstallableApps" directory. You can then use these war files to deploy into another Application Server.
If you choose not to deploy any of the web applications listed below, then the Rhythmyx built-in Application Server (Tomcat) will not be installed.

- [✓] Rhythmyx Publisher
- [✓] Rhythmyx Web Services

Figure 3: Select the Web Applications dialog

5 Complete the installation.

6 Install the package containing the Portal Connector examples into Rhythmyx so that the components are available to you from the Rhythmyx interface.

To install the archive packages into the correct Rhythmyx directories use the Rhythmyx Multi-Server Manager. If you have installed your devtools in a location other than your Rhythmyx Server, and you do not have access to the Rhythmyx Server's file system, copy:

```
<Rhythmyx root>/Samples/Portals/BEA8/portalExamples.pda
```

from the Rhythmyx Server to your devtools location.

Open the Multi-Server Manager using RhythmyxMultiServerManager.exe in the directory where you have installed your devtools.

Use the Rhythmyx Multi-Server Manager Help with the following instructions to install the package. The topics referenced are located in the Rhythmyx Multi-Server Manager online and print documentation:

a) Follow the steps in the "Server Registration Maintenance Procedure" section to register and connect to your Rhythmyx server.

b) Since the archive already exists, skip the section "Creating and Editing Archives and Generating Deployment Descriptors." Note that you do not have to map elements, identify ID types, or select dependencies or ancestors.

c) Follow the steps in the "Installing an Archive" section to install the packages into the correct Rhythmyx directories:

- In the Select an Archive and Server dialog:
- In Target Server, choose your Rhythmyx server.
- In Archive Selection, select Server and click […]. Choose portalExamples.pda The file is located in:
  <Rhythmyx root>/Samples/Portals/BEA8/
  or the location that you moved it to.
- In Server Archive Name, enter a descriptive name, such as ExamplesArchive.
- In the Choose Installation Mode dialog, choose Typical.
- In the Add Packages to Install dialog, choose [Add All].
- Proceed to the Summary dialog and finish the installation.

d) Follow the steps in the "Reviewing Installed Packages" section to make sure that your packages installed correctly.

When you have completed the installation:

- In the CMS Publisher tab, the BEA Publisher, Editions, Sites, and Content Lists are registered. See Registering Publishing Components for graphics of the registrations in Rhythmyx.

7 Install the archive file `<Rhythmyx root>/Samples/DeprecatedSamples.pda` into Rhythmyx. This archive file includes former default Rhythmyx components that the BEA 8.1 Connector uses. See the document *Installing Rhythmyx* for help installing this archive.

8 Associate the sample Rhythmyx Site and Variant with Communities. In our sample, we use the Default Community.

9 Add the Role that you added to WebLogic (`rxrole`) to Rhythmyx. In ***WebLogic Setup Steps*** (on page 13), we added `rxrole` as the Portal Connector Role.

a) Create a new Web Server Security Provider named *Basic*. See the topic "Adding a Web Server Security Provider" in the Rhythmyx Server Administrator Help.

b) Add the WebLogic Role to the Rhythmyx Server Administrator.

c) Associate the WebLogic Role with a Rhythmyx Community. In our example, we associate `rxrole` with the Default Community.

d) Then associate the WebLogic Role with the Workflows you plan to use. Also assign the WebLogic Role to specific Workflow States within the Workflows. For help, see the topic "Adding a Role" in the online Rhythmyx Server Administrator Help, the topic "Associating a Role with a Community" in the online Rhythmyx Content Management Help and the section About Workflows in Rhythmyx in the online Rhythmyx Content Management Help.

WebLogic Setup Steps

In our sample, setting up WebLogic prior to installing the BEA Portal Connector involves adding a Rhythmyx user, group, and Role. Your security setup may differ from the sample setup, which uses the default security provider in WebLogic.

To set up WebLogic for using the Rhythmyx sample before installing the BEA Portal Connector:

1 Open the Administration Console by entering `http://localhost:7001/console` in a browser or, in Windows, by clicking the [**Start**] button and selecting *Programs > BEA WebLogic Platform 8.1 > Examples > WebLogic Platform > Server Admin Console,* and log in.

2 In the Navigation pane, expand *Portal > Security > Realms* to open the Portal > Realms dialog, and under Name click *myrealm*.

The Administration Console displays the myrealm page.

3 Add a Portal Connector user:

a) At the top of the page, click the User Management tab, then Manage Groups Within This Security Realm.

b) On the Groups page, click Configure a New Group.

c) Add a new Group for the Portal Connector. In our example, we use the Group `rxgroup`.

4 Add a Portal Connector Role and add this Role to the Portal Connector Group, `rxgroup`.

a) At the top of the page, click myrealm, then click Manage Roles Within This Security Realm.

The Administration Console displays the Global Roles page

b) Click Configure a New Global Role.

c) Add a Role for the Portal Connector. In our example, we add `rxrole`.

d) Click the [**Apply**] button.

e) Click the Conditions tab.

f) Under **Role Condition**, select *Caller is a member of the Group* and click the [**Add**] button.

g) In the **Enter Group Name** field, enter the Portal Connector group (`rxgroup`) and click the [**Add**] button.

h) Click the [**OK**] button.

i) Click the [**Apply**] button .

5 Add a Portal Connector user and add it to rxgroup.

a) At the top of the screen, click myrealm, then click Manage Users within this Security Realm.

b) On the Users page, click Configure a new user.

c) Add a user for the Portal Connector. In our example, we use the user `rxuser` and password `password`.

d) Click [**Apply**].

e) Click the Groups tab.

f) Move `rxgroup` under Current Groups and click the [**Apply**] button.

CHAPTER 3

Installation

Deploying the Rhythmyx Publisher and Web Services

The steps in this section assume that your are using your local host and the default WebLogic port (7001). To deploy the Rhythmyx Publisher and Web Services on the WebLogic server:

1. Start the WebLogic Server Administration Console by entering `http://localhost:7001/console` or in Windows by clicking the [**Start**] button and selecting *Start > Programs > BEA WebLogic Platform 8.1 > Examples > WebLogic Platform > Server Admin Console*.
2. In the Navigation pane, expand portal/Deployments/Web Application Modules.
3. Click Deploy a New Web Application Module.
4. Follow the instructions on the screens to deploy `Rhythmyx.war`. Use the following information:
   - the files are located in: `<Rhythmyx Root>/InstallableApps/FrontEnd/Rhythmyx.war`;
   - leave the Deployment Target as `portalServer` to deploy the file into your WebLogic 8.1 samples directory: `bea8\WebLogic81\samples\domains\portal\`.
5. In the Navigation pane, click Portal/Deployments/Web Application Modules.
6. Click Deploy a New Web Application Module.
7. Follow the instructions on the screens to deploy `soap.war`. Use the following information:
   - The file is located in: `<Rhythmyx Root>/InstallableApps/AllInOne/RxServices.war`.
   - Leave the Deployment Target as `portalServer` to deploy the file into your WebLogic 8.1 samples directory: `bea8\WebLogic81\samples\domains\portal\`.
8. After deployment is complete, run the Rhythmyx and BEA Servers. Open a Web browser and enter `http://localhost:7001/Rhythmyx`. If the Rhythmyx servlet has deployed correctly, the browser displays the Rhythmyx Content Explorer.

Modifying the Rhythmyx Web Application Deployment Descriptors

Before you install your Web applications onto WebLogic, you must set the Session Settings in their deployment descriptors. You may also change the init params in your Rhythmyx.war deployment descriptor if you do not want to use the default values.

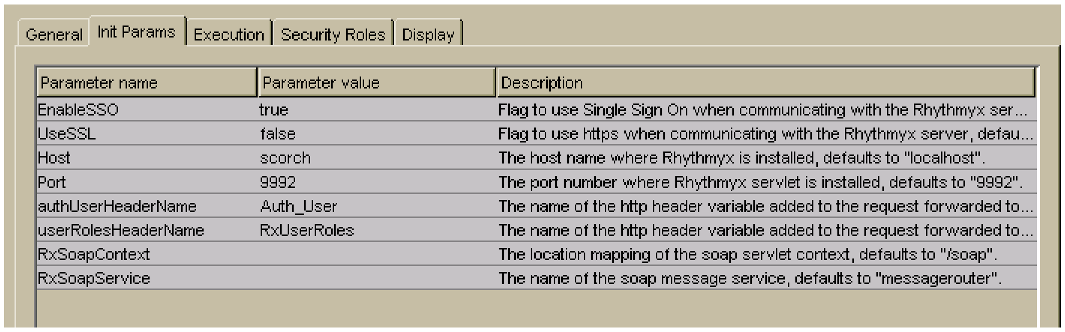
To set your session settings:

1. In your Start Menu, choose *Programs > BEA WebLogic Platform 8.1 > Other Development Tools > WebLogic Builder*.
2. Open `<Rhythmyx root>/Installable Apps/FrontEnd/Rhythmyx.war`.
3. In the Navigation pane, click Session Settings.
4. In the Display pane, click the HTTP Cookie Settings tab.
5. In **Cookie path**, change the value to a forward slash: "/".
6. Change **HTTP cookie name** to *JSESSIONID* if it does not already have this value.
7. Open `<Rhythmyx root>/Installable Apps/AllInOne/RxServices.war` and Repeat Steps 3 through 6.

   To:

   - use non-default values for Authorized User Name and Roles variables;
   - use SSL; or
   - use a non-default Rhythmyx Server or Port;
   - you must make certain edits to the Web application deployment descriptors. To modify the Rhythmyx deployment descriptor:
8. In your Start Menu, choose *Programs > BEA WebLogic Platform 8.1 > Other Development Tools > WebLogic Builder*.
9. Open `<Rhythmyx root>/Installable Apps/FrontEnd/Rhythmyx.war`.
10. In the Navigation pane, click Servlets/Rhythmyx.

11 Click the Init Params tab.

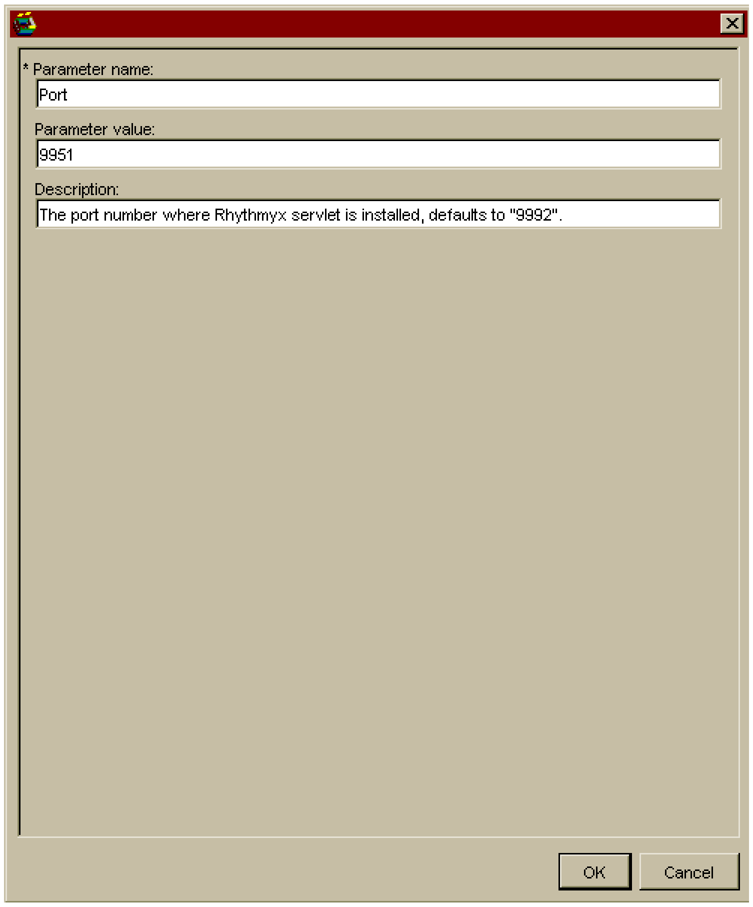


General | Init Params | Execution | Security Roles | Display

Parameter name	Parameter value	Description
EnableSSO	true	Flag to use Single Sign On when communicating with the Rhythmyx ser...
UseSSL	false	Flag to use https when communicating with the Rhythmyx server, defau...
Host	scorch	The host name where Rhythmyx is installed, defaults to "localhost".
Port	9992	The port number where Rhythmyx servlet is installed, defaults to "9992".
authUserHeaderName	Auth_User	The name of the http header variable added to the request forwarded to...
userRolesHeaderName	RxUserRoles	The name of the http header variable added to the request forwarded to...
RxSoapContext		The location mapping of the soap servlet context, defaults to "/soap".
RxSoapService		The name of the soap message service, defaults to "messagerouter".

Figure 4: WebLogic Builder, Init Params tab

12 To change a parameter value, double-click on the **Parameter Name**. In the dialog that opens, change the **Parameter Value** and click [**OK**].



* Parameter name:
Port

Parameter value:
9951

Description:
The port number where Rhythmyx servlet is installed, defaults to "9992".

Figure 5: Init Param dialog

- By default, SSL use is not enabled. To enable SSL use, set the **UseSSL** parameter to `true`.
- By default, the Rhythmyx servlet host is your local host. To use a different host, set the **Host** parameter to the host name or address. Note: You must also change the messagerouter parameter in RxServices.war. See "If you are not using the default Rhythmyx Server or Port" below.

- By default, the Rhythmyx servlet port is 9992. To use a different port, set the **Port** parameter to the port number. Note: You must also change the messagerouter parameter in RxServices.war. See "If you are not using the default Rhythmyx Server or Port" below.
- By default, the Authorized User Name is `Auth_User`. To use another value for the Authorized User Name variable change the value of the **authUserHeaderName** to the name you want to use. You must also change the value of the **Authenticated user header name** field on the Web Server Security Provider Details dialog in Rhythmyx to the same value.
- By default, the User Roles Header Name is `RxUserRoles`. To use a different value for the User Roles Header Name, change the value of the userRolesHeaderName parameter to the name you want to use. You must also change the value of the User role list header field on the Web Server Security Provider Details dialog in Rhythmyx to the same value.
- By default, the SOAP servlet context is `/soap`. To change the default, enter a value in the **RxSoapContext** parameter.
- By default, the SOAP message service is `messageservice`. To change the default, enter a value in the **RxSoapService** parameter.

If you are not using the default Rhythmyx Server or Port:

1. Open `<Rhythmyx root>/Installable Apps/AllInOne/RxServices.war`.
2. In the Navigation pane, click Servlets/messagerouter.
3. Click the Init Params tab.
4. Set the **rhythmyx_url** parameter to use the correct host and port.

Registering SOAP services on the WebLogic server

1. Perform the steps in Deploying the Rhythmyx Publisher and Web Services on the WebLogic Server if you have not already completed them. At this point, Web services are deployed but not installed.
2. Open a command prompt window on the WebLogic server.
3. Change the directory to `c:\<Rhythmyx Root>/InstallableApps/AllInOne` or another location where you have deployed RxServices.war .
4. Unzip RxServices.war.
5. From the command prompt, run:

   ```
   registerPublisher.bat localhost:7001
   ```

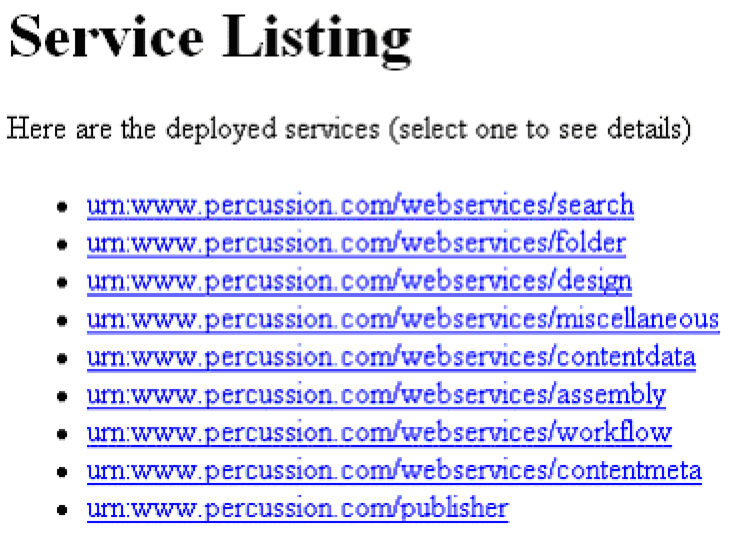
   and then:

   ```
   registerWS.bat localhost:7001
   ```

6. After installation is complete, open a Web browser and enter the URL: `http://localhost:7001/RxServices`.

   The Apache SOAP administration console opens.
7. Click Run the admin client.
8. Click [**List**] to access `http://localhost:7001/RxServices/admin/index.html`.

   If SOAP services have installed correctly, the Service Listing should list all of the following deployed services:



Service Listing

Here are the deployed services (select one to see details)

- urn:www.percussion.com/webservices/search
- urn:www.percussion.com/webservices/folder
- urn:www.percussion.com/webservices/design
- urn:www.percussion.com/webservices/miscellaneous
- urn:www.percussion.com/webservices/contentdata
- urn:www.percussion.com/webservices/assembly
- urn:www.percussion.com/webservices/workflow
- urn:www.percussion.com/webservices/contentmeta
- urn:www.percussion.com/publisher

Figure 6: SOAP Service Listing

Deploying the Rhythmyx Portlets to the Portal

The files for the sample Rhythmyx Portal and Portlets are located in `rxPortal.ear`, which you installed into `<Rhythmyx root>/InstallableApps/Weblogic/`.

To deploy `rxPortal.ear` to the BEA 8.1 Portal:

1. Start the WebLogic Server Administration Console by starting and browser and entering the URL `http://localhost:7001/console` or in Windows by clicking the [**Start**] button an selecting *Programs > BEA WebLogic Platform 8.1 > Examples > WebLogic Portal > Server Admin Console*.
2. In the Navigation pane, click portal/Deployments/Applications.
3. Click Deploy a New Application.
4. On the Deploy a New Application: Select The Archive For This Application page, select *<Rhythmyx root>/InstallableApps/Weblogic/rxPortal.ear* and click [**Continue**].
5. On the Deploy a New Application: Review Your Choices and Deploy page, leave the **Deployment Target** as *portalServer*, and click [**Deploy**].

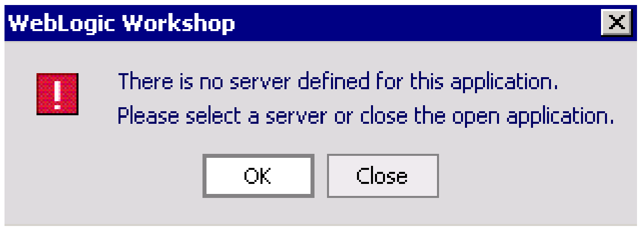
Testing and Modifying the Rhythmyx Portal and Portlets

After you deploy rxPortal.ear, it is located in the sample BEA Portal application directory. Now, you can view the Rhythmyx Portal and Portlets and modify them in the WebLogic Workshop.

To work with the Rhythmyx Portal and Portlets:

1. Open the WebLogic Workshop by clicking the [Start] button and choosing *Programs > BEA WebLogic Platform 8.1 > WebLogic Workshop 8.1.*
2. Choose *File > Open > Application*.
3. In the Open Workshop Application dialog, navigate to `<BEA root>/weblogic81/samples/domains/portal/portalServer/.wlnotdelete/rxportal/rxportal`.work and click [**Open**].

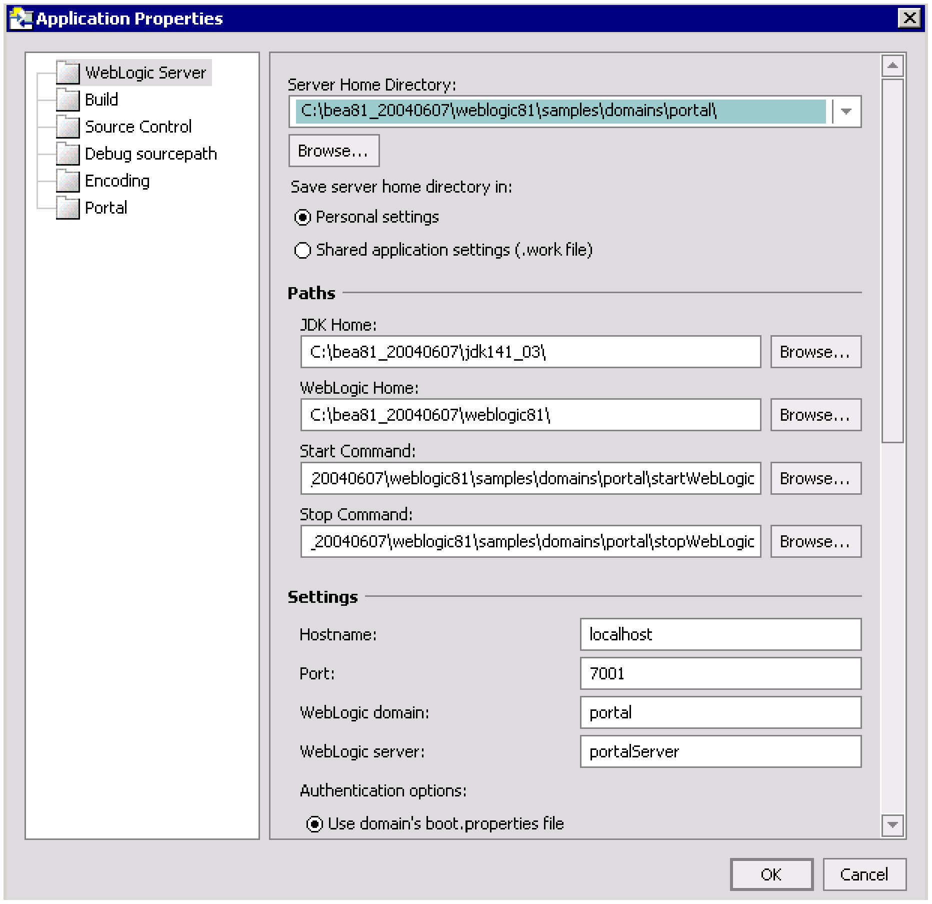
   If you do not have a server home directory defined in BEA, a dialog prompts you to define one:


WebLogic Workshop

There is no server defined for this application.
Please select a server or close the open application.

Figure 7: Select a Server prompt

Click [**OK**] to open the Application Properties dialog:



Application Properties

- WebLogic Server
- Build
- Source Control
- Debug sourcepath
- Encoding
- Portal

Server Home Directory: C:\bea81_20040607\weblogic81\samples\domains\portal\

Save server home directory in: [✓ Personal settings] Shared application settings (.work file)

Paths

JDK Home: C:\bea81_20040607\jdk141_03\

WebLogic Home: C:\bea81_20040607\weblogic81\

Start Command: 20040607\weblogic81\samples\domains\portal\startWebLogic

Stop Command: _20040607\weblogic81\samples\domains\portal\stopWebLogic

Settings

Hostname:	localhost
Port:	7001
WebLogic domain:	portal
WebLogic server:	portalServer

Authentication options: [✓ Use domain's boot.properties file]

Figure 8: WebLogic Workshop Application Properties dialog

Choose the correct home directory in the **Server Home Directory** drop list. The dialog automatically fills in the other fields. Click [**OK**].

4 In the Navigation pane, expand rx/portlets/Rhythmyx.

5 Double-click on rx.portal to view the configuration of the Portal book in a Window. The Portlets are already set up in the Portal, and you do not have to make any modifications to use it; however, you probably want to customize it for your system. In this Window, you can rearrange or remove Portlets from the book.

6 To preview the Portal, start a browser and browse to `http://localhost:7001/rx/`.

7 For complete instructions on modifying Portals in the WebLogic Workshop, adding custom Portlets and applying Portlet code changes to the Portlets in the Portal, choose Help and an appropriate sub-topic in the Menu Bar.

NOTE: The WebLogic Workshop is a testing and staging environment for your Portal. To deploy the Portal to a production environment, choose Help > Help Topics in the Menu bar, and refer to the topic Deploying Applications.

Enabling Portlets for use in the Standard Sample

After you complete the procedure in Adding the Rhythmyx Portlets to your Portal Project, you have included the Rhythmyx Portlets in your BEA Rhythmyx Portal, but have not enabled them for users. To decide which Portlets you want to enable, see Determining Which Portlets to Use in Your System.

To enable Portlets, follow the instructions *Build Your First Portal*: "Create a Portlet and Make it Visible to the Portal" section (http://edocs.bea.com/wlp/docs70/portal1/portal.htm) of the WebLogic Portal documentation. Use the following information along with the instructions in the BEA documentation:

1. Use the **User Name** *Administrator* and the **Password** *password* to open Portal Administrator.
2. In the Portal Management Home Page, choose your Rhythmyx BEA Portal (if you are using the sample, choose Avitek Users (rxgroup)).
3. In the Edit Portlets Entitlements and Attributes Page, select each of the Rhythmyx Portlets and set the attributes as indicated in the BEA documentation.
4. Once you have enabled the Portlets, they are available to use in the Sample Portal. No further configuration or setup is required. View them by entering:

   `http://localhost:7001/samplePortal`. If you have added a separate Page for the Rhythmyx portlets, click the tab for the Page.

Modifying the Look and Feel of your Portal

The Portal Connector does not supply CSS files or skins for determining the look and feel of your Portal. See the BEA documentation for information about applying a default BEA Look and Feel to your Portal or creating your own look and feel.

Using a Different Database

By default, WebLogic uses the PointBase database, but you may use another database if you perform the proper configuration. The BEA 8.1 Portal Connector can publish to Pointbase, Oracle, SQL Server and DB2; however, it can only display content from Pointbase, Oracle, and SQL Server. For instructions on using another database, see the section "Using Databases" in the online *WebLogic Workshop Help*.

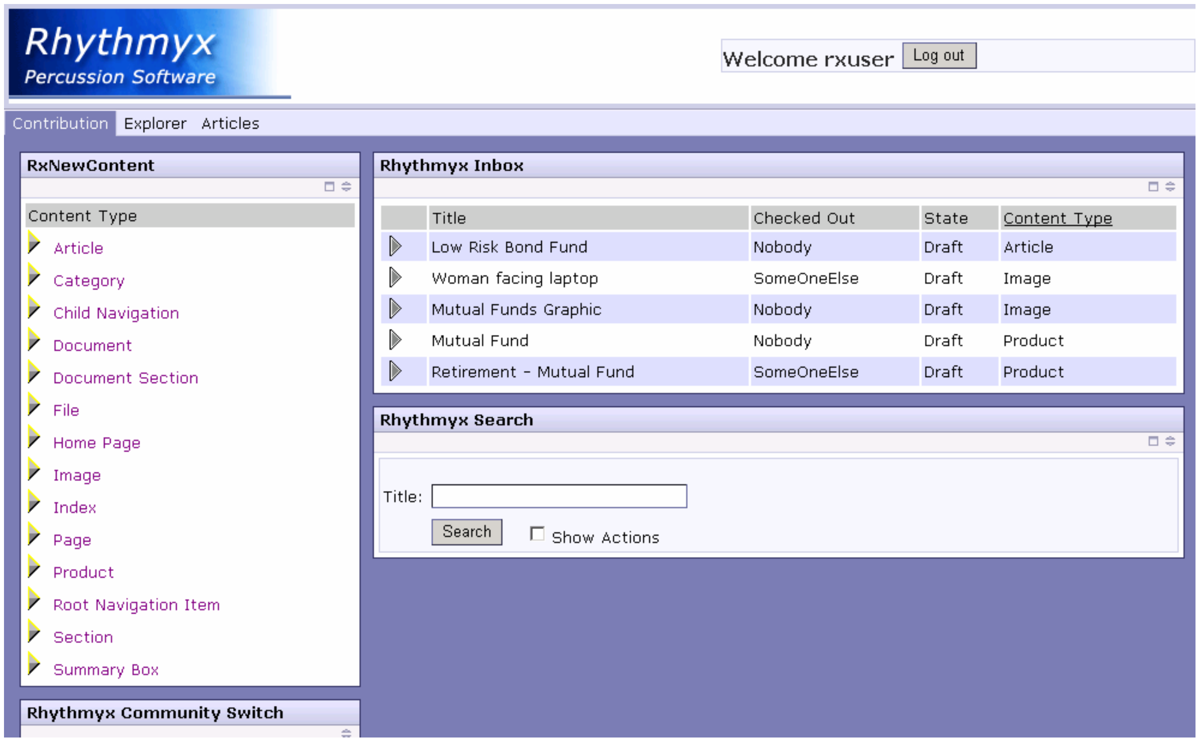
NOTE: If you publish to an Oracle database earlier than Version 9.2.0.3, place the earlier classes12.jar file on the WebLogic classpath before any other Oracle driver jar file.   The classes12.jar file that WebLogic includes by default is not backwards compatible for versions of Oracle earlier than 9.2.0.3, although the earlier jar file works with newer versions of Oracle.

CHAPTER 4

Working with the Content Contributor UI

Understanding the Content Contributor UI

The Rhythmyx Portlets included in the BEA Portal Connector let you display a Rhythmyx Content Explorer interface through your Portal that lets users perform most of the functions available in the Content Explorer interface accessed directly from the Rhythmyx Server.. Users can create content in Content Editors, preview it, and perform Active Assembly on it, and transition it through Workflow. In addition users can perform content searches and change their Communities and Locales. The Rhythmyx Portlets provide a default Inbox View through which users can view and access content. You can easily modify this View to create other Views available in Rhythmyx or to create custom Views.



Content Type
- Article
- Category
- Child Navigation
- Document
- Document Section
- File
- Home Page
- Image
- Index
- Page
- Product
- Root Navigation Item
- Section
- Summary Box

Title	Checked Out	State	Content Type
Low Risk Bond Fund	Nobody	Draft	Article
Woman facing laptop	SomeOneElse	Draft	Image
Mutual Funds Graphic	Nobody	Draft	Image
Mutual Fund	Nobody	Draft	Product
Retirement - Mutual Fund	SomeOneElse	Draft	Product

Title:
Search ☐ Show Actions

Figure 9: WebLogic Portal displaying Rhythmyx Portlets

NOTE: The Rhythmyx Portlets do not provide functionality for using Navigation Tree folders, Impact Analysis, and Active Assembly for Documents, and they do not provide access to the Publishing, Workflow, and System tabs available to Rhythmyx Administrators. Administrators must access Content Explorer directly from the Rhythmyx Server to access the Delivery (Publishing), Workflow and System functions.

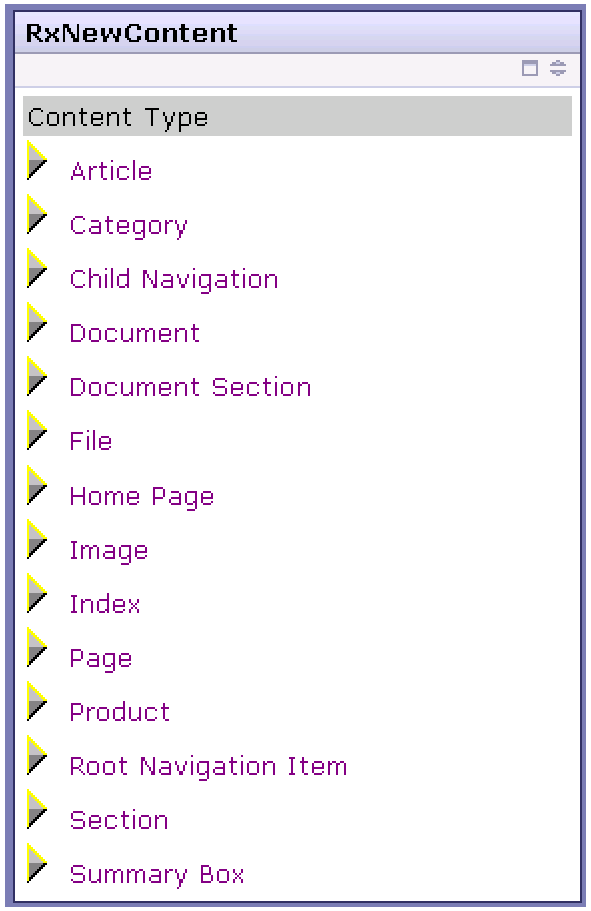
NOTE: A Community is a group of Roles that require access to similar information in Rhythmyx. A Locale is a Rhythmyx login language including its regional variation.

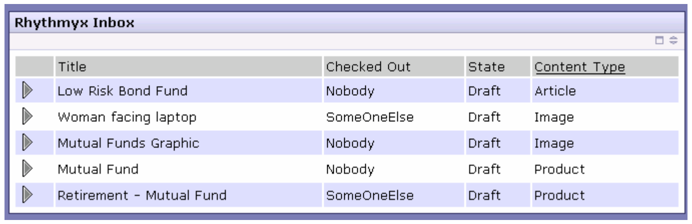
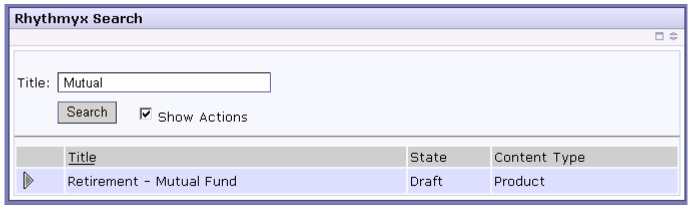
Determining which Portlets to use in Your System

When you first start using the Portal Connector, leave all contribution Portlets in your Portal book so you can see how they function. Most likely, you will want your Portal to perform the functions that these Portlets provide. However, if you customize your WebLogic Rhythmyx Portal, you may choose to replace some of these Portlets with custom Portlets. In this case, you can remove any of the Portlets provided from your Portal.

NOTE: The only Portlet associated with the delivery function of the BEA Portal Connector rather than the Content Contribution function is the Article Portlet.

The BEA Portal Connector installs the following Rhythmyx Portlets:

Portlet	Function	View of Portlet
New Content	Displays a list of all Content Types the login user has access to. The Content Type name is a link that opens the Content Editor for the Content Type in a new window. If you hold the cursor over the arrow in front of a Content Type, a pop-up displays the description of the Content Type.	 RxNewContent Content Type Article Category Child Navigation Document Document Section File Home Page Image Index Page Product Root Navigation Item Section Summary Box

Portlet	Function	View of Portlet
Inbox	Displays a View showing all Content Items assigned to the login user with the Display Format columns Title, Checked Out, State, and Content Type. Clicking on the gray arrow next to a Content Item opens an Action Panel with Content Item metadata and the Action Menu options Edit, View, Preview , Workflow, Active Assembly, and Copy URL to Clipboard. Clicking on any column sorts by that column. Title of sorted column is underlined. Minimized by default.	
Search	Displays a two-part table. Initially, only displays the top row, which has a Title field for entering a string, a Show Actions option for including Action Panels with the results, and a [**Search**] button for initiating a search. When the search is complete, the bottom section displays the results using the same Display Format as the Inbox View. Clicking on the gray arrow next to a Content Item opens an Action Panel with Content Item metadata and the Action Menu options Edit, View, Preview , Workflow, Active Assembly, and Copy URL to Clipboard options. Clicking on any column sorts by that column. Title of sorted column is underlined. Minimized by default.	

Rhythmyx Inbox

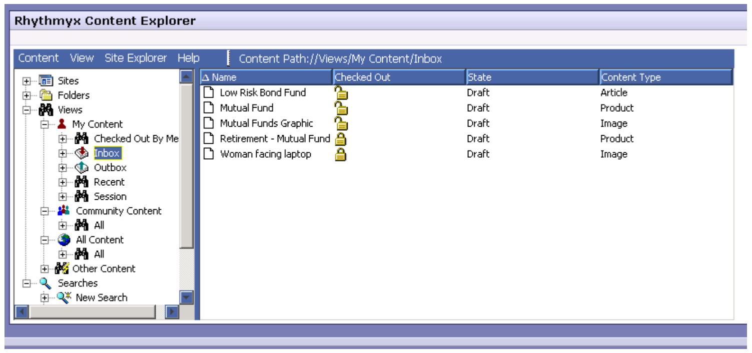
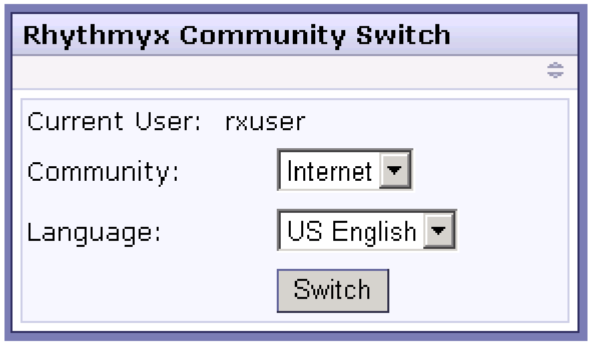
Title	Checked Out	State	Content Type
Low Risk Bond Fund	Nobody	Draft	Article
Woman facing laptop	SomeOneElse	Draft	Image
Mutual Funds Graphic	Nobody	Draft	Image
Mutual Fund	Nobody	Draft	Product
Retirement - Mutual Fund	SomeOneElse	Draft	Product

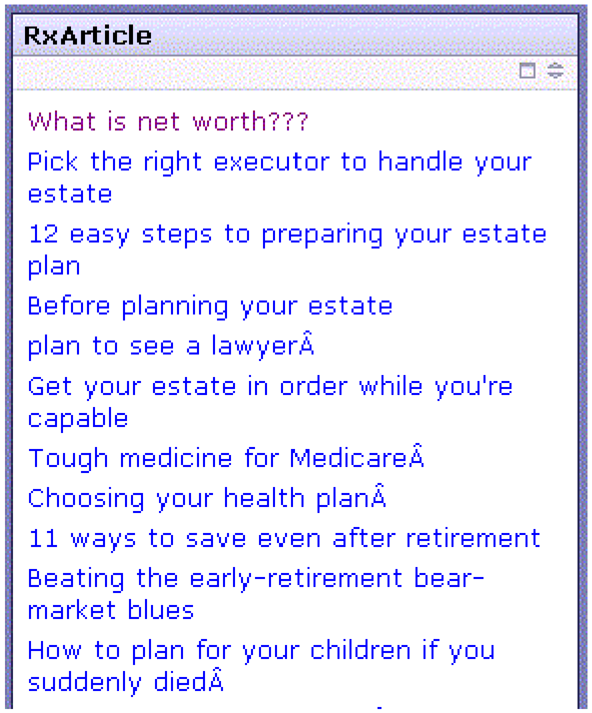
Rhythmyx Search

Title: Mutual

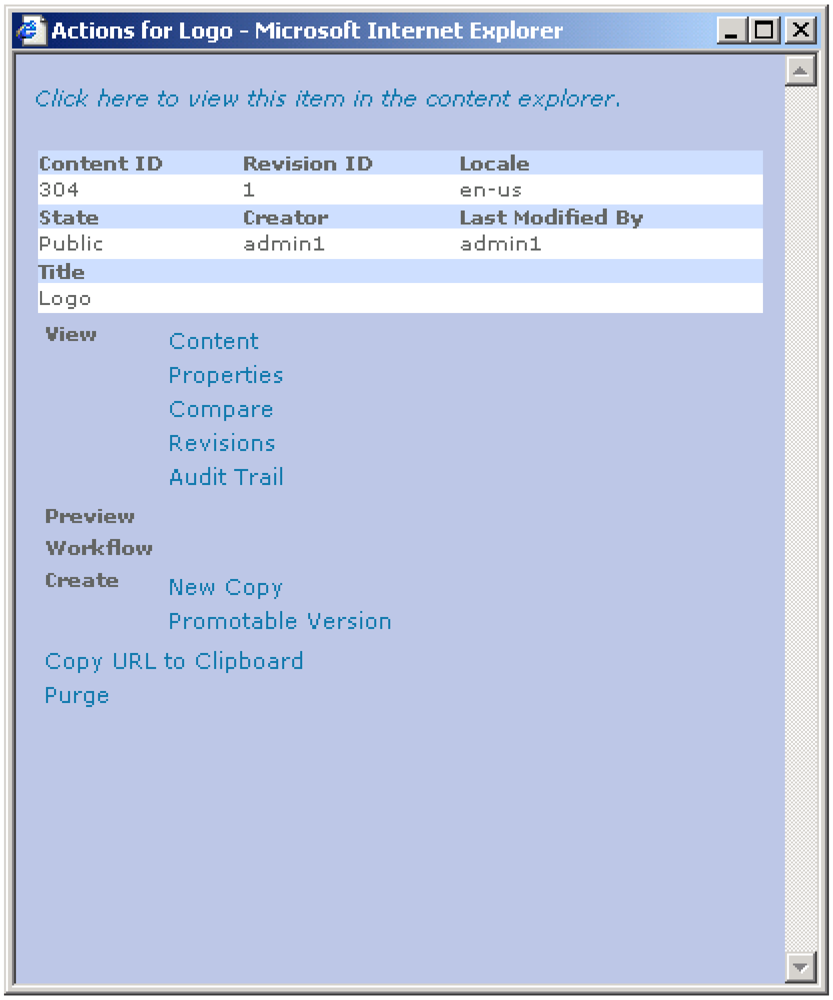
Search ☑ Show Actions

Title	State	Content Type
Retirement - Mutual Fund	Draft	Product

Portlet	Function	View of Portlet
Content Explorer	Displays the Content Explorer applet. Note that menu choices are slightly different than the choices available when Content Explorer is accessed directly from the Rhythmyx Server. See Choosing How to Use the Portal Connector for details.	 Rhythmyx Content Explorer Content View Site Explorer Help | Content Path://Views/My Content/Inbox Sites Folders Views My Content Checked Out By Me Inbox Outbox Recent Session Community Content All All Content All Other Content Searches New Search Name | Checked Out | State | Content Type Low Risk Bond Fund | | Draft | Article Mutual Fund | | Draft | Product Mutual Funds Graphic | | Draft | Image Retirement - Mutual Fund | | Draft | Product Woman facing laptop | | Draft | Image
Community and Locale	Displays a form with a drop list of available Communities and a drop list of available Locales. In the drop lists, the Login Community and Locale are currently selected. The form includes a [**Switch**] button that changes the user's logged-in Community and/or Locale to those selected in the drop lists. Refresh the page after you change the Community to reflect the change in other Portlets on the page.	 Rhythmyx Community Switch Current User: rxuser Community: Internet Language: US English Switch

Portlet	Function	View of Portlet
Article (This Portlet is used with the delivery functionality, not the contribution functionality)	Displays a list of article titles that you have published to your Portal. When you hover your cursor over the title, a pop-up displays the article abstract. The article title is also a link; when you click it, it opens the item in the Published HTML Page Variant format, which includes the article body.	 RxArticle What is net worth??? Pick the right executor to handle your estate 12 easy steps to preparing your estate plan Before planning your estate plan to see a lawyerÂ Get your estate in order while you're capable Tough medicine for MedicareÂ Choosing your health planÂ 11 ways to save even after retirement Beating the early-retirement bear-market blues How to plan for your children if you suddenly diedÂ

When you click on the arrow next to a Content Item in the Inbox and Search View Portlets, an Action panel with content metadata and Action Menu options opens. When the user moves the cursor over the Action Menu options, if they are clickable, they appear bold and underlined. The user clicks on an Action Menu option to perform the action on the Content Item.


Actions for Logo - Microsoft Internet Explorer

Click here to view this item in the content explorer.

Content ID	Revision ID	Locale
304	1	en-us
State	**Creator**	**Last Modified By**
Public	admin1	admin1
Title		
Logo		

Figure 10: Action Panel

Note: When accessed through the Portal, the Copy URL to Clipboard action adds the parameter sys_stickycommunity="true" to the URL. For example:

```
http://10.10.10.10:9992/Rhythmyx/sys_ActionPage/Panel.html?sys_contenti
d=308&sys_stickycommunity="true"
```

When set to "true," sys_stickycommunity tells Rhythmyx not to modify the user's Community when the user accesses the Content Item.

For more information about the Copy URL to Clipboard action, see the topic "Sending a Content Item's URL to Another User" in the *Rhythmyx Content Explorer* online help.

Extending the Content Contributor UI

Ways to extend the Content Contributor UI

You can extend the BEA Portal Connector by modifying the Rhythmyx Portlets provided or using them as templates for creating new Portlets. By using the classes and interfaces installed with the Portal Connector, you can make common Web Services requests to Rhythmyx. You can also use these classes as models for creating requests using any of the Rhythmyx Web Services.

The BEA Portal Connector uses the IPSWsHelper java interface to perform Web Services calls between WebLogic and Rhythmyx. The functions in this class are available to you to help you extend your Content Contributor UI. The following table lists a few of the actions that you may want to add to existing or new Portlets and the IPSWsHelper functions that you can use to implement these actions . This table is intended to give you an idea of the type of functions available to you in IPSWsHelper; many additional functions exist and may meet your specific needs. See the IPSWsHelper interface in the Rhythmyx online Javadoc in `<Rhythmyx root>\docs\Rhythmyx\javadocs\index.html` for references to all of the available functions.

To:	Use the IPSWsHelper function:	For more help, see:
Call a Rhythmyx application from a Portlet and get results	ExecuteCallDirect (uses the Call Direct Web Service)	Inbox Portlet code
Log in to a Portlet	login (uses the Login Web Service)	Community and Locale Portlet code Login Web Service in Web Services Developer's Kit document
Retrieve Content Types available to a user	GetContentTypeList (uses the ContentTypeList Web Service)	New Content Portlet code Content Type List Web Service in Web Services Developer's Kit document
Extend the search capability	Search (uses the PSSearch java class and the Search Web Service)	Inbox Portlet code Search Portlet code

NOTE: Java function names are subject to change. Check the Javadoc for PSWsHelper before using the functions listed in the above table.

You can also add functionality by using the sample Rhythmyx Portlets as templates. For example, the only sample View Portlet provided is the Inbox Portlet. If you wanted to add an additional View, you could, use the Inbox Portlet as a template for creating your new View Portlet.

Tag Library

The Portal Connector includes a sample tag library located in `<Rhythmyx root>/Samples/Portals/taglibsrc.jar`. You can use this tag library or modify it to make changes to the Rhythmyx Portlets or to use with your own Portlets. For general help creating and working with tag libraries, see `http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html`.

This section includes a short description of each tag in the tag library, including the details of all attributes of each tag. Note that in the example values listed, single and double quotation marks are equivalent when denoting the value of the attribute.

import

NOTE: This tag is currently not used.

Imports a referenced file from the file system into the output stream.

Attributes:

Attribute	**Description**	**Required?**
path	Specifies the path to the file to import.	Yes

This tag is useful for developers who want to pull static content that has been published to the file system.

inbox

Generates the HTML content of the authenticated user's inbox in the same fashion as Content Explorer.

Attributes:

Attribute	**Description**	**Required?**
imagepath	Specifies the directory where the graphics used to format the output are stored. The default version of the portal connector uses rxactionmenu.gif. You can customize the appearance of the portal by specifying a different graphic.	Yes
columns	Comma-separated list of the Display Format columns to render in the portlet. The count of these columns must match the count of the parameters specified in the `parameters` attribute. If you specify a value for this attribute, you must also specify a value for the `parameters` attribute. Example: `columns="Title,State,ContentType"`	No

Attribute	Description	Required?
parameters	Comma-separated list of CMS system fields to display. The count of the parameters specified in this attribute must match the count of the columns specified in the `columns` attribute. f you specify a value for this attribute, you must also specify a value for the `columns` attribute. Example: `parameters="sys_title,sys_statename,sys_contenttypename"`	No
titlecss	If specified, defines the CSS class to use when rendering the title of the results.	No
rowcss	If specified, defines the CSS class to use when rendering the output rows. If the evenrowcss attribute is also specified, the specified class is used only for rendering odd-numbered rows.	No
evenrowcss	If specified, defines the CSS class to use when rendering even-numbered output rows.	No.

search

Presents a simple user interface for entering a search request and displays the results of the search.

Attributes:

This tag uses the same attributes as the `inbox` tag.

newcontent

Renders one output line for each Content Type the current user is allowed to access based on the current logged Community and Role.

Attributes:

Attribute	Description	Required?
imagepath	Specifies the path to the directory where the graphics used to format the output are stored. The default version of the portal connector uses arrow_sm.gif. You can customize the appearance of the portal by specifying a different version of this graphic (different content but the same name).	Yes
titlecss	If specified, defines the CSS class to use when rendering the title of the results.	No

addresourceonce

This tag provides a workaround for a shortcoming in some portals. It adds a CSS, JavaScript, and imagepath to a page once when rendering a page rather than issuing repeated requests for it.

Attributes:

Attribute	Description	Required?
imagepath	Specifies the path to the directory where the graphics used in rendering the output are stored. Example: `imagepath='<%=portletResponse.encodeUrl("framework/rx_resources/images")%>'`	Yes
csspath	Specifies the path to a stylesheet to include. Example: `csspath='<%=portletResponse.encodeUrl("framework/rx_resources/popmenu.css")%>'`	Yes
jspath	Specifies the path to a JavaScript file to include. Example: `jspath='<%=portletResponse.encodeUrl("framework/rx_resources/popmenu.js")%>'`	Yes

setwshelper

This tag specifies the Web Services helper class that the tag library should instantiate. For the WebLogic 8.1 Portal Connector, the class is `com.percussion.integration.bea.webservices.PSWsHelper`.

Attributes:

Attribute	Description	Required?
classname	The complete classname of the Web Services helper class. For the WebLogic 8.1 Portal Connector, the class is `com.percussion.integration.bea.webservices.PSWsHelper`.	Yes

clob

Specifies the name of a page context data object to extract into a clob and read into the output stream.

Attributes:

Attribute	Description	Required?
value	Specifies a value in the form `name.columnname`. The `name` specifies the pageContext item name to retrieve. This object must be of the type java.util.Map. The `columnname` is used on the map to extract the actual value, which must be of the type java.sql.clob. Example: value="body.data"	Yes

Code example:

```
<a href="<c:out value='${url}'/>">Back to article list</a>
      <sql:query var="bodies" maxRows="1">
         SELECT data FROM item
         WHERE id = <c:out value="${param.document_id}"/>
      </sql:query>
      <c:forEach var="body" items="${bodies.rows}">
         <rx:clob value="body.data" />
      </c:forEach>
<a href="<c:out value='${url}'/>">Back to article list</a>
```

In this example, the `data` in the `value` attribute of the `<clob>` tag is derived from the SELECT statement in the `<sql:query>` tag. The `body` in the `value` attribute of the `<clob>` tag is derived from the `<c:forEach>` tag. Consult a JSTL reference for information on these tags.

cx

Displays the Content Explorer.

Attributes: None

Extending Other Parts of the Content Contributor Interface

The Web Services used in the Portal Connector are:

- Login
- Content Type List
- Search Request
- Call Direct

To extend the Portal Connector's capabilities, you may use any of the other Web Services provided by Rhythmyx. For help formatting calls to Web Services, refer to the IPSWsHelper interface in the online Javadoc at `<Rhythmyx root>\docs\Rhythmyx\javadocs\index.html.` For descriptions of the Rhythmyx Web Services, see the *Web Services Developer's Kit* document.

Migrating components from the Portal development environment to the Portal production environment

You should always create and modify Portlets in a development Portal and, after you test them, move them to your production Portal. Use the WebLogic Workshop as a development and test environment for your Portals and Portlets, and then deploy them to a production server. See the "Deploying Applications" section of the WebLogic Workshop online help for an explanation of the best methods of developing and deploying your WebLogic applications.

Best Practices for the Content Contributor UI

1 Remote contributors who have connections with narrow bandwidths may find downloading the Content Explorer applet through the BEA Portal time-consuming. If many of your remote contributors use narrow bandwidths, you may choose to disable the Content Explorer Portlet, or give users the option of opening it when they access the Portal. In this case, the Inbox Portlet would function as their interface with Rhythmyx.

CHAPTER 5

Setting up Delivery

Understanding Delivery

The delivery functionality of the BEA 8.1 Portal Connector lets you publish content and metadata associated with the content from the Rhythmyx repository to the BEA Portal repository. You can publish content to the Portal either as unformatted data or as formatted snippets and pages to use in Portlets or custom JSP. If you are planning to present the data as formatted when you retrieve it from the Portal, assemble it with a Variant in Rhythmyx and publish it as a formatted page or snippet. You can also publish data in other output types, such as PDF and XML.

To publish to the Portal, you must register and set up your Publishing components correctly and ***configure them to use the publisher plug-in*** (see "Performing and Extending Delivery" on page 45). See the document Implementing Publishing in Rhythmyx for complete information about publishing in Rhythmyx and registering publishing components.

NOTE: The BEA 8.1 publisher plug-in is designed to publish content and metadata to a database on the BEA Portal. You can also publish to a file system, however, using a Rhythmyx file system Publisher. The recommended practice is to publish dynamic content to the repository database and publish static content, such as generic site images and cascading stylesheets to the file system.

Performing and Extending Delivery

Publishing with the Publisher Plug-in

When the BEA publisher plug-in publishes content to the Portal repository, by default it publishes content and metadata into our sample WebLogic tables, ITEM and ITEM_METADATA. The schema for these tables is:

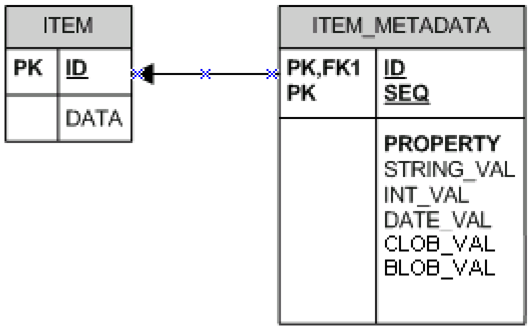


Figure 11: Item and Item_Metadata tables

NOTE: for information about adjusting these the repository tables for your specific database implementation, see ITEM and ITEM_METADATA Tables .

You publish body content to the DATA column in the ITEM table and metadata associated with the body content to the ITEM_METADATA table. The sample assembler applications use one of the default Rhythmyx assembler applications to format content and an additional resource for connecting to the Portal repository and retrieving the information to send to it. This section shows you how to configure and run publishing with the publisher plug-in using the sample applications, and other sample components provided by Rhythmyx or with the Portal Connector. You can modify this example to set up publishing in your system.

Static content, such as images used in navigation, is published to the file system. You must modify the publishing location of this content to match the installation directory of the portal. Static content is most commonly published to a separate Web server (such as Apache or IIS), but you can also publish static content directly to the portal by publishing it to the portal's .wlnotdelete/images subdirectory.

How the Example Works

In the Rhythmyx example, the Article Content Type (rx_ceArticle) is assembled using the Published HTML Page Variant. The Published HTML Page Variant displays the content of the article and includes Related Images, Related Articles, and Hot Articles in its Slots. The Published HTML Page Variant and metadata are published to the ITEM and ITEM_METADATA tables using the Article BEA8 Portal Variant. This Variant includes connection information for the database as well as mappings for the content and metadata.

The sample Article Portlet displays body content and metadata published to the BEA repository. The Article Portlet displays a list of the content (Articles) that you have published to your Portal. Each article title is a link that displays the item in the Published HTML Page Variant. A link in the Published HTML Page switches you back to the list of content.

The sample components installed with the BEA 8.1 Connector include the assembler application that assembles content and metadata for the Portal repository: rx_casPortalContent. This application includes two resources: article and bea_article. The BEA 8.1 Portal Connector only uses the bea_article resource, which assembles content and metadata for the BEA 8.1 repository. The sample components also include a Content List application, rx_pubPortalContentLists which publishes content and metadata to the Portal repository.

All necessary sample registrations for a Publisher, Variants, Sites, and other publishing components in Rhythmyx are also included, as well as a sample Edition and Content Lists for publishing.

These samples are intended to provide you with templates for creating your own BEA publishing applications and components. You may reuse them in your own system or copy them and modify them as needed.

Checklist for Publishing with the Plug-in

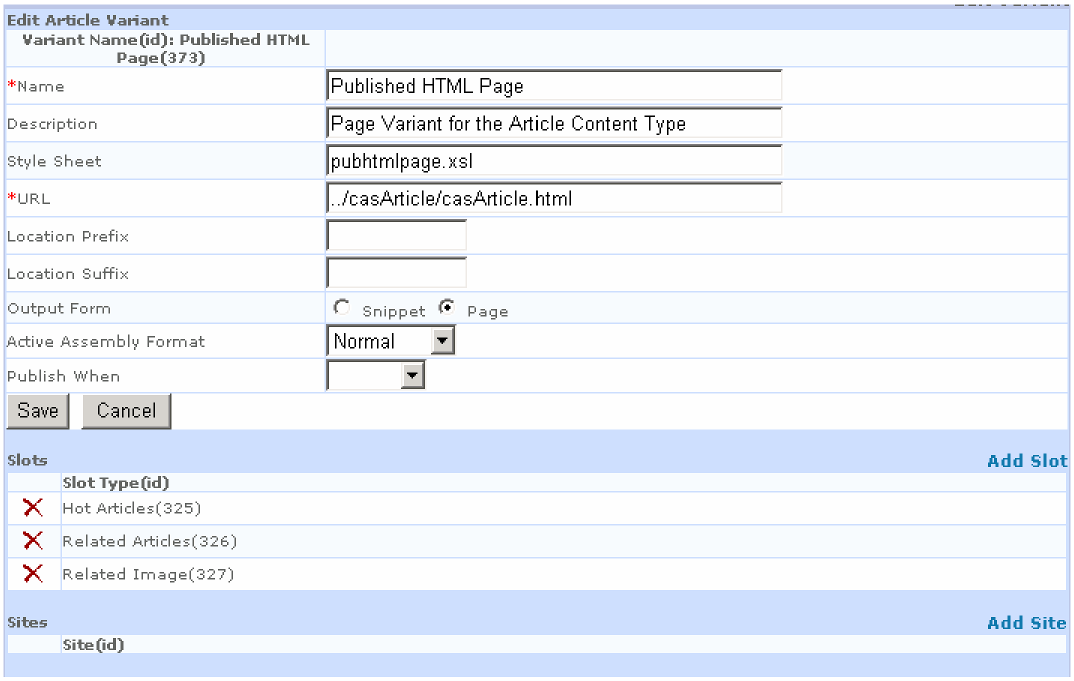
1. Develop resource files, Variants, and Slots.
2. Create assembler applications.
3. Build the Content List application.
4. Register Publishing components in Rhythmyx.
5. Set up WebLogic DataSources.
6. Enter `http://localhost:7001/Rhythmyx` and publish the Editions manually or automatically.
7. Review your results.

The following sections use the example components in the package `<rx root>/Samples/BEA8/portalExamples.pda` which you installed using ***Rhythmyx's Multi-Server Manager*** (see "Rhythmyx Setup Steps" on page 9), and the graphics display these sample components. However, each topic includes steps for creating your components that are customized to the needs of your system.

Developing Resource Files, Variants, and Slots

Rhythmyx includes sample resource files for the Variants assembled by its sample applications. You may use these, or create your own, by marking up HTML files. See the instructions in the Rhythmyx Workbench Online Help, "Managing Applications/Defining Page Content" section and the topic "Defining Variants" in the document Implementing Content Editors and Content Assembly in Rhythmyx for general help on creating assembler resources. After you create Variants and Slots in your resources, you must register them with Rhythmyx.

The following graphic shows the registration for the default Published HTML Variant. Use this as a guide for registering Variants that represent dynamic content. In this example, Page is selected as the output form because the Variant represents a full page. The three Slots included in the Variant (Related Image, Related Articles, and Hot Articles), are included under Slots.



Edit Article Variant

Variant Name(id): Published HTML Page(373)	
*Name	Published HTML Page
Description	Page Variant for the Article Content Type
Style Sheet	pubhtmlpage.xsl
*URL	../casArticle/casArticle.html
Location Prefix	
Location Suffix	
Output Form	Snippet [✓ Page]
Active Assembly Format	Normal
Publish When	

Slots

	Slot Type(id)
✗	Hot Articles(325)
✗	Related Articles(326)
✗	Related Image(327)

Sites

	Site(id)

Figure 12: Sample Variant Registration

To register your Variants:

1 Enter a **Name** and optionally a **Description** for the Variant.

2 If your Variant represents formatted content, in **Style Sheet**, enter the stylesheet file name. If your Variant represents an image or binary file, leave **Style Sheet** blank.

3 In **URL** enter the address of the Variant in your Rhythmyx root.

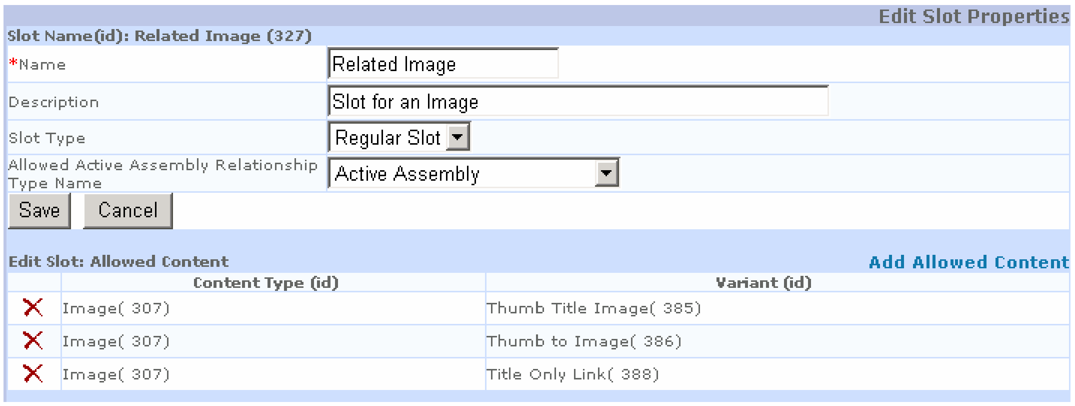
4 If you are using a single default Location Scheme, enter a value in **Location Prefix**. See the section "Site Folder Publishing" in the document *Implementing Publishing* for more information about using single default Location Schemes.

5 In **Output Form**, indicate if the Variant represents a Snippet or a Page. Note: a Snippet is assembled content that represents part of a Web page; a Page is assembled content that represents a full Web page.

6 If your Variant represents formatted content, in **Active Assembly Format**, choose Normal. If your Variant represents an image or binary file, in **Active Assembly Format**, choose Non-HTML.

7 If your Variant represents formatted content, add the Slots that appear in the Variant. If your Variant represents an image or binary file, do not add any Slots.

The following graphic shows the sample registration for the Related Image Slot. Use this as a guide for registering your Slots. The allowed content for the Slot is the Image Content Type formatted as either the Thumb to Image, Thumb Title Image, or the Title Only Link Variant..


Edit Slot Properties

Slot Name(id): Related Image (327)

*Name	Related Image
Description	Slot for an Image
Slot Type	Regular Slot
Allowed Active Assembly Relationship Type Name	Active Assembly

Edit Slot: Allowed Content

	Content Type (id)	Variant (id)
✗	Image(307)	Thumb Title Image(385)
✗	Image(307)	Thumb to Image(386)
✗	Image(307)	Title Only Link(388)

Figure 13: Sample Slot Registration

To add your Slots:

1 Enter a **Name** and optionally a **Description** for the Slot.

2 In **Slot Type**, choose Regular Slot if the Slot appears on a regular Variant or Inline Slot if the slot holds an inline link or inline image.

3 Add the Variants that may appear in the Slot.

To see the other sample Variant and Slot registrations, go to the Variant and Slot pages in your Content Explorer's System tab.

Creating the Assembler Application

You must create two types of assembler applications when you publish to the WebLogic database. One of these assembler applications assembles content for the ITEM and ITEM_METADATA tables in the database repository. In our sample this is the rx_casPortalContent application. The other assembler application assembles content as the Variant that is published to the DATA column in the ITEM table. In our sample, this is the default Rhythmyx casArticle application.

Variant Assembler Application

The casArticle assembler application is a Rhythmyx default application that creates Variants of the Rhythmyx default Article Content Type. By default, the bea_article resource in the rx_casPortalContent application obtains a Variant produced by the casArticle assembler application and inserts the Variant into the DATA field of the ITEM database table.

The graphic below shows casArticle as it appears in the Rhythmyx Workbench. pubhtmlpage.xsl produces the Published HTML Page Variant, which our sample Content List publishes to the DATA column in the database repository. The following graphic shows the resources that casArticle uses.

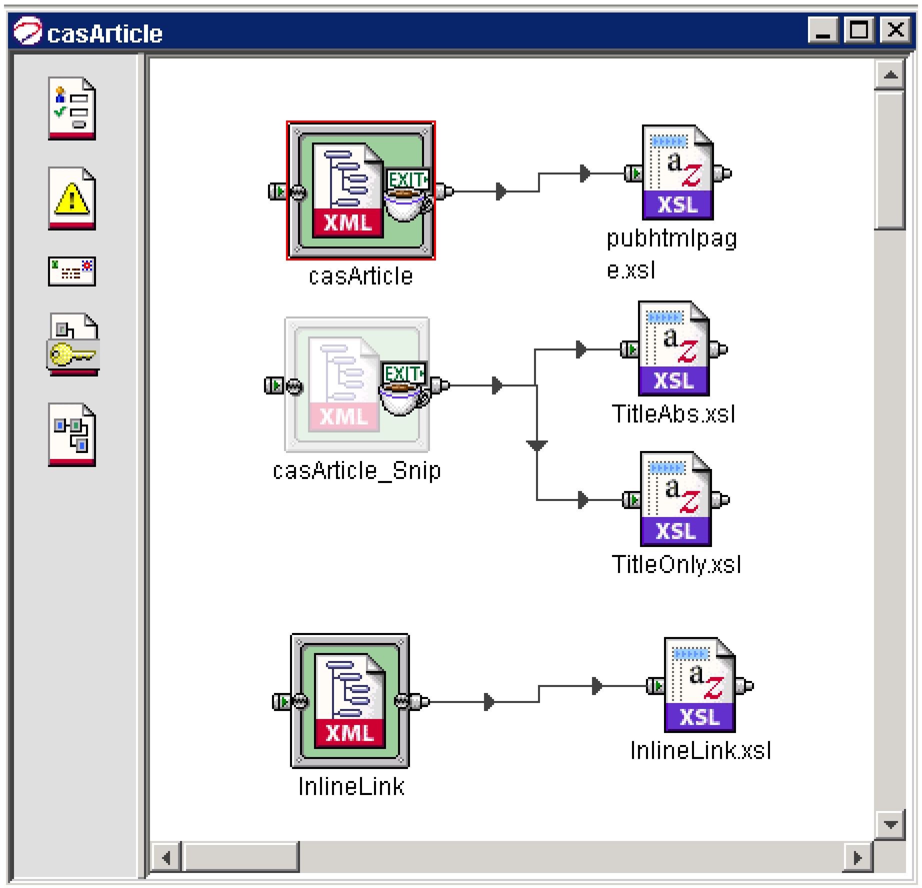
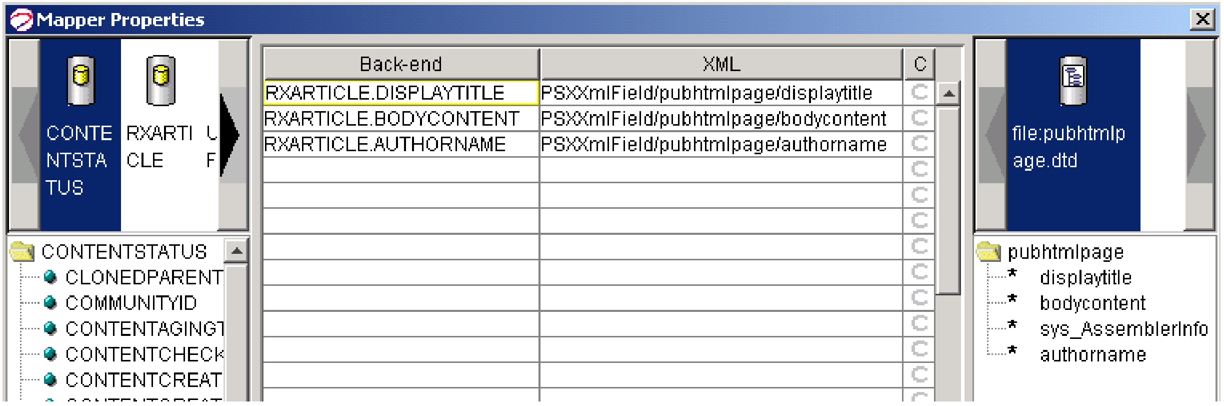


Figure 14: casArticle application in Workbench

The casArticle resource maps information to fields in the XML Variants.

The following graphic shows the mapping for the casArticle resource. Back-end data is mapped to displaytitle, bodycontent, and authorname fields in the published HTML page:



Back-end	XML	C
RXARTICLE.DISPLAYTITLE	PSXXmlField/pubhtmlpage/displaytitle	
RXARTICLE.BODYCONTENT	PSXXmlField/pubhtmlpage/bodycontent	
RXARTICLE.AUTHORNAME	PSXXmlField/pubhtmlpage/authorname	

Figure 15: Mapper for casArticle resource

Note that you must also use resources that assemble the Variants allowed in each Slot of the Variant that you are publishing to the DATA column as well. These resources may be included in the same assembler application that assembles the DATA Variant or in other Assembler Applications.

To create your own Variant assembler applications for publishing body content to the IBM Portal:

- Drop your resource HTML for your Variants onto a new application in the Workbench.
  - Attach the sys_casAddAssemblerInfo post-exit to any text resource.
  - Attach the sys_xdTextToTree post-exit to any text resource that that uses the eWebEditPro control.
  - Create static resources for image files and binary files and map them. See “Mapping Images and Other Mime Types” in the Rhythmyx Workbench online help.
  - For text resources, configure the back end tables, mappers, and selectors as specified in the “Creating Assembly Applications” section in the Rhythmyx document Implementing Content Editors and Content Assembly.
  - For resources that create inline images or inline links, requests are not made to backend tables, so add the RXDUAL “dummy” table to your datatank and do not define any requestor properties. Map elements as explained in the topic “Creating Inline Image and Inline Link Assemblers” in the document *Implementing Content Editors and Content Assembly*.

For more information about creating assembler applications, see the “Creating Assembly Applications” section in *Implementing Content Editors and Content Assembly*.

For more information about creating applications in the Rhythmyx Workbench, see the section “Managing Applications” in the Rhythmyx Workbench online help.

Database Assembler Application

The rx_casPortalcontent application is a standard Rhythmyx application that obtains content from the casArticle assembler and processes it for insertion into the ITEM and ITEM_METADATA tables of the BEA WebLogic repository.

The graphic below shows the rx_casPortalContent application opened in the Rhythmyx Workbench.

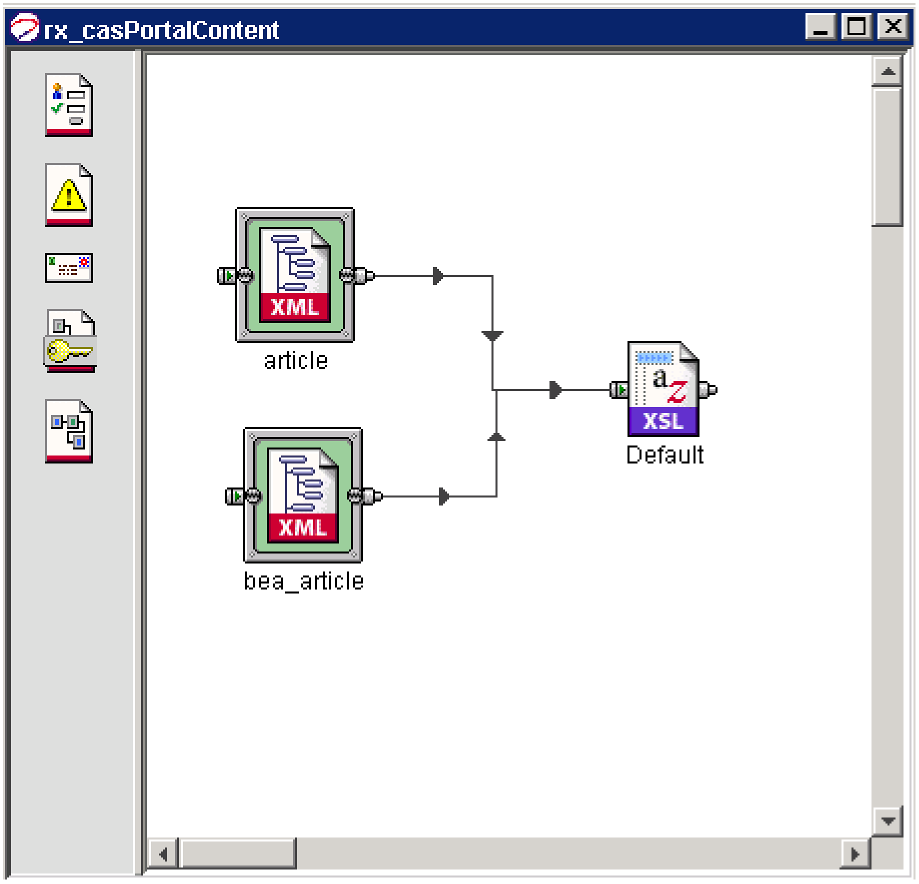
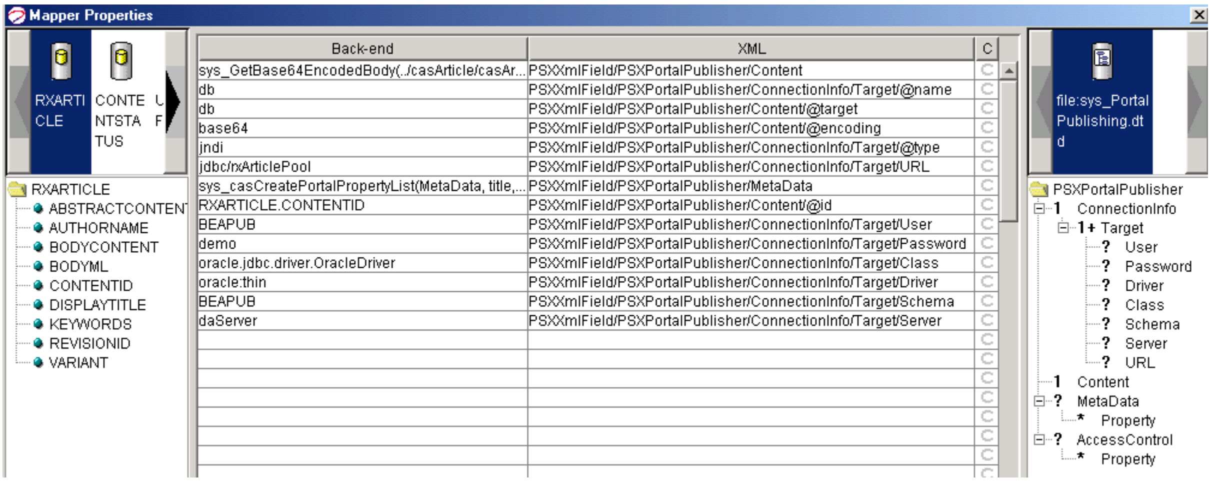


Figure 16: rx_casPortalContent application opened in Workbench

For BEA 8.1, we use the bea_article resource which connects to the BEA Portal database and assembles data for the database tables.

The mappings for the bea_article resource are as follows:



Mapper Properties

RXARTICLE CONTENTSTATUS

RXARTICLE
- ABSTRACTCONTENT
- AUTHORNAME
- BODYCONTENT
- BODYML
- CONTENTID
- DISPLAYTITLE
- KEYWORDS
- REVISIONID
- VARIANT

Back-end	XML	C
sys_GetBase64EncodedBody(../casArticle/casAr...	PSXXmlField/PSXPortalPublisher/Content	C
db	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/@name	C
db	PSXXmlField/PSXPortalPublisher/Content/@target	C
base64	PSXXmlField/PSXPortalPublisher/Content/@encoding	C
jndi	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/@type	C
jdbc/rxArticlePool	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/URL	C
sys_casCreatePortalPropertyList(MetaData, title,...	PSXXmlField/PSXPortalPublisher/MetaData	C
RXARTICLE.CONTENTID	PSXXmlField/PSXPortalPublisher/Content/@id	C
BEAPUB	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/User	C
demo	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/Password	C
oracle.jdbc.driver.OracleDriver	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/Class	C
oracle:thin	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/Driver	C
BEAPUB	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/Schema	C
daServer	PSXXmlField/PSXPortalPublisher/ConnectionInfo/Target/Server	C

file:sys_PortalPublishing.dtd

PSXPortalPublisher
- 1 ConnectionInfo
  - 1+ Target
    - ? User
    - ? Password
    - ? Driver
    - ? Class
    - ? Schema
    - ? Server
    - ? URL
- 1 Content
- ? MetaData
  - * Property
- ? AccessControl
  - * Property

Figure 17: Mapper for bea_article resource

XML field	**Back end Data**	**Description**
PSXPortalPublisher/Content	Maps to the sys_GetBase64EncodedBody exit.	The sys_GetBase64EncodedBody exit retrieves the HTML document specified by the first (resource) parameter, extracts the information between the `<BODY>` tags, base64 encodes it, and returns it as a String.In our example, we set the resource parameter to: `../casArticle/casArticle.html`, the location of the HTML document that contains the BODY content.
PSXPortalPublisher/ConnectionInfo/ Target/@name	db	The publishing target name. By default this is *db* (database).
PSXPortalPublisher/Content/@target	db	The literal *db* specifies that the information in the following XML `<db>` tag should be published to the BEA address specified for the `<db>` target under the `<ConnectionInfo>` element.
PSXPortalPublisher/Content/@encoding	base64	The type of encoding for the HTML body variant. See "Encoding Data" in the document *Implementing Database Publishing* for more information.
PSXPortalPublisher/ConnectionInfo/ Target/@type	jndi	The type of Target may be either *jndi* or *jdbc*. In our example we set it to *jndi*, which indicates that the application should use the database connection information specified in the Database Pool and Data Source registered for the Rhythmyx database on the WebLogic Server. If you specify *jdbc*, you must also specify values for User, Password, URL, Driver and Schema.It is recommended that you set the type to *jndi* and specify the database information in a Database Pool and Data Source. [cross ref]
PSXPortalPublisher/ConnectionInfo/ Target/URL	jdbc/rxArticlePool	The JDBC URL. In our example, we enter *jdbc/rxArticlePool*, the value of our JDBC Data Source JNDI Name. Check your database documentation for the syntax that you should use.

XML field	Back end Data	Description

PSXPortalPublisher/Metadata	Map to sys_casCreatePortalPropertyList exit	The sys_casCreatePortalPropertyList exit creates a list of the metadata properties stored in the ITEM_METADATA table. In our example, we add three Portal properties. The parameters for the exit are set as follows:
		ElementName = Metadata
		name0 = title
		type0 = string
		pattern0 =
		value0 = RXARTICLE.DISPLAYTITLE
		name1 = abstract
		type1 = string
		pattern1 =
		value1 = RXARTICLE.ABSTRACTCONTENT
		name2 = category
		type2 = string
		pattern2 = ,
		value2 = RXARTICLE.KEYWORDS

PSXPortalPublisher/Content/@id	RXARTICLE.CONTENTID	The ID of the Content Item to be published. In our example, the ID is taken from the backend RXARTICLE table.
PSXPortalPublisher/ConnectionInfo/Target/User	BEAPUB	Username that accesses the database schema.
PSXPortalPublisher/ConnectionInfo/Target/Password	demo	Password that accesses the database schema.
PSXPortalPublisher/ConnectionInfo/Target/Class	oracle.jdbc.driver.OracleDriver	Driver class.
PSXPortalPublisher/ConnectionInfo/Target/Driver	oracle:thin	Driver name.

XML field	Back end Data	Description
PSXPortalPublisher/ConnectionInfo/Target/Schema	BEAPUB	Schema. Required for Oracle.
PSXPortalPublisher/ConnectionInfo/Target/Server	daServer	Server IP or name.

For instructions on creating non-text resources, see the topic “Mapping Images and Other Mime Types” in the Rhythmyx Workbench online help

To create your own assembler applications for publishing to the BEA Portal, add or copy the bea_article publish resource from the rx_casPortalContent application into your application.

- The backend tables in the example are CONTENTSTATUS and RXARTICLE. Change RXARTICLE to the backend table for the Content Type that you are publishing.
- The DTD is `sys_PortalPublishing.dtd`. Do not modify this DTD; it includes the connection information and target table data expected by the BEA 8.1 Portal Publisher plugin.
- The selector selects on *contentid* and *revisionid*. You may leave the default or select content on any criteria.
- Make the appropriate changes in the mapper:
    - Map PSXPortalPublisher/Content to the sys_GetBase64Encoded exit as in the example, but set the resource parameter in the exit equal to the resource representing the Variant that you want to publish.
    - Leave PSXPortalPublisher/ConnectionInfo/Target/@name and PSXPortalPublisher/Content/@target mapped to db. The Portal Connector plugin only publishes to database, so this value may remain at the default.
    - Leave PSXPortalPublisher/Content/@encoding as base64.
    - In most cases, leave PSXPortalPublisher/ConnectionInfo/Target/@type as *jndi*. This allows you to use the connection information set in the database connection pool and data source. (If you use jdbc, you must specify the additional information listed below.)
    - If @type is jndi, map PSXPortalPublisher/ConnectionInfo/Target/URL to the resource name. If @type is jdbc, this parameter is not used.
    - Map PSXPortalPublisher/MetaData/Metadata to the sys_casCreatePortalPropertyList exit as in the example, but change the exit’s parameters to reflect the parameters used in your system.
    - Map PSXPortalPublisher/Content/@id to a Content ID column in the backend table representing the Content Type being published.
    - Map PSXPortalPublisher/ConnectionInfo/Target/Driver to the correct driver name for your database.
    - If you specify PSXPortalPublisher/ConnectionInfo/Target/@type as *jdbc* enter the following values. They are not used for *jndi*.

- enter the correct username for connecting to your database in PSXPortalPublisher/ConnectionInfo/Target/User
- enter the corresponding password in PSXPortalPublisher/ConnectionInfo/Target/Password
- enter the java classname of the JDBC driver in PSXPortalPublisher/ConnectionInfo/Target/Class
- enter the server ip or name in PSXPortalPublisher/ConnectionInfo/Target/Server

For more information about creating assembler applications, see the "Creating Assembly Applications" section in the Rhythmyx document *Implementing Content Editors and Content Assembly*.

For more information about creating applications in the Rhythmyx Workbench, see the section "Managing Applications" in the Rhythmyx Workbench online help.

Building the Content List Application

The Portal Connector includes the rx_pubPortalContentLists application which includes the information required for publishing the correct information to the BEA Portal. Use this application to publish your Content Lists or modify it to create your own Content List application.

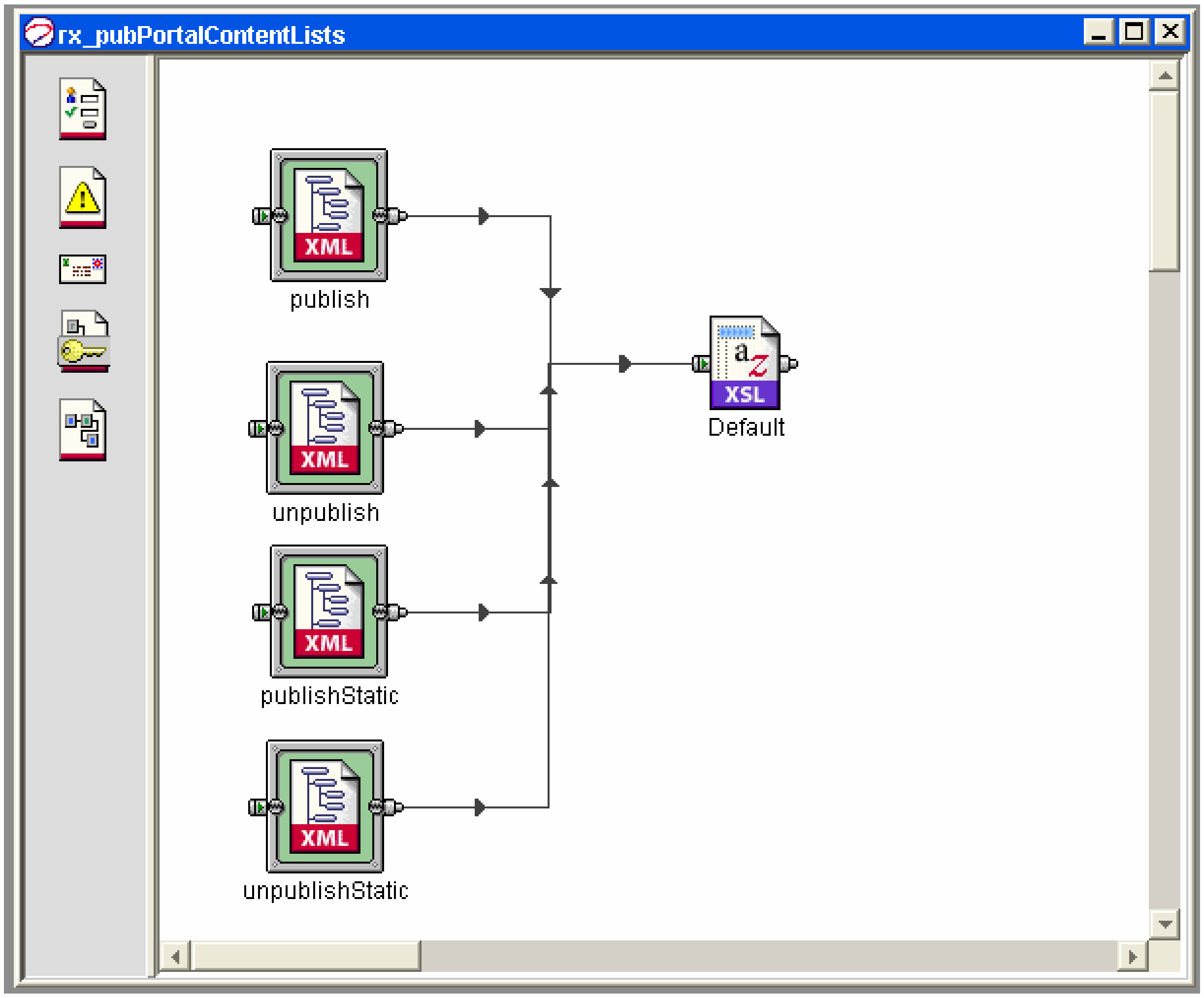


Figure 18: rx_pubPortalContentLists application in Workbench

The publish resource in rx_pubPortalContentLists is similar to the contentlist_article and contentlist_generic resources in the sample rx_pubContentLists application in Rhythmyx, but it :specifies the deliverytype portal, which indicates that Rhythmyx should use the BEA 8.1 Portal publisher plugin.

The following table shows the mappings in the sample publish.xml resource in the rx_pubPortalContentLists application. Refer to the descriptions to understand the necessary modifications that you must make to your own Content List resource:

DTD Element	**Map to**	**Description**
contentlist/@context	PSXSingleHtmlParameter/sys_context	Identifies the reference to the publishing location. Parameter value is obtained from the Edition Content List registration.
contentlist/@deliverytype	sys_Literal(portal)	Identifies the delivery format of the content. The value portal tells the Publisher to use the BEA 8.1 Publisher Plug-in. If you enter any other value, the Publisher will not use the plug-in.
contentlist/contentitem/title	CONTENTSTATUS.TITLE table column	Recovers the title of the content from the CONTENTSTATUS table.
contentlist/contentitem/@contentid	CONTENTSTATUS.CONTENTID table column	Recovers the identifier of the content to be published from the CONTENTSTATUS table.
contentlist/contentitem/@revision	CONTENTSTATUS.CURRENTREVISION table column	Recovers the revision identifier of the content to be published from the CONTENTSTATUS table.
contentlist/contentitem/@variantid	CONTENTVARIANTS.VARIANTID	Recovers the identifier of the Variant to be published. from the CONTENTVARIANTS table.

DTD Element	Map to	Description
contentlist/contentitem/contenturl	This element is mapped to the sys_MakeAbsLink UDF.	This is a link to the assembler application that will create the content that will be extracted. Map the sys_MakeAbsLink parameters as follows: Resource - CONTENTVARIANTS.ASSEMBLYURL ParamName1 - sys_contentid ParamValue1- CONTENTSTATUS.CONTENTID ParamName2 - sys_Revision ParamValue2 - CONTENTSTATUS.CONTENTREVISION ParamName3 - sys_variantid ParamValue4 - PSXSingleHTMLParameter/sys_variantid ParamName4 - sys_context ParamValue5 - PSXSingleHTMLParameter/sys_context ParamName5 - sys_authtype ParamValue6 - PSXSingleHTMLParameter/sys_authtype ParamName6 - sys_siteid ParamValue6 - PSXSingleHTMLParam/sys_siteid

DTD Element	Map to	Description
contentlist/contentitem/delivery/location	sys_casGeneratePubLocation UDF	This function generates the location where the content is published. Map the sys_casGeneratePubLocation parameters as follows: **Parameter** / **Value** VariantID: PSXSingleHTMLParameter/sys_variantid(Variant ID of the page variant to be published.) ContentID: CONTENTSTATUS.CONTENTID Revision: CONTENTSTATUS.CURRENTREVISION
contentlist/contentitem/modifydate	CONTENTSTATUS.CONTENTMODIFIEDDATE table column	This parameter recovers the date the content was modified from the CONTENTSTATUS table.
contentlist/contentitem/modifyuser	CONTENTSTATUS.CONTENTLASTMODIFIER table column	This parameter recovers the name of the last user to modify the content from the CONTENTSTATUS table.
contentlist/contentitem/expiredate	CONTENTSTATUS.CONTENTEXPIRYDATE table column	This parameter recovers the expiration date of the content from the CONTENTSTATUS table.
contentlist/contentitem/contenttype	CONTENTSTATUS.CONTENTTYPEID table column	This parameter recovers the identifier of the content type of the content from the CONTENTSTATUS table.

The unpublish application has the same resources, which you must modify in the same manner as the resources on the Publish application.

The publishStatic and unpublishStatic applications are essentially identical to the publish and unpublish applications, except that they publish to the file system.

Note that you can repeat the parameter sys_variantid as many times as necessary in each application. For example, in the publish application, it is repeated twice, once to publish the full image and once to publish the thumbnail.

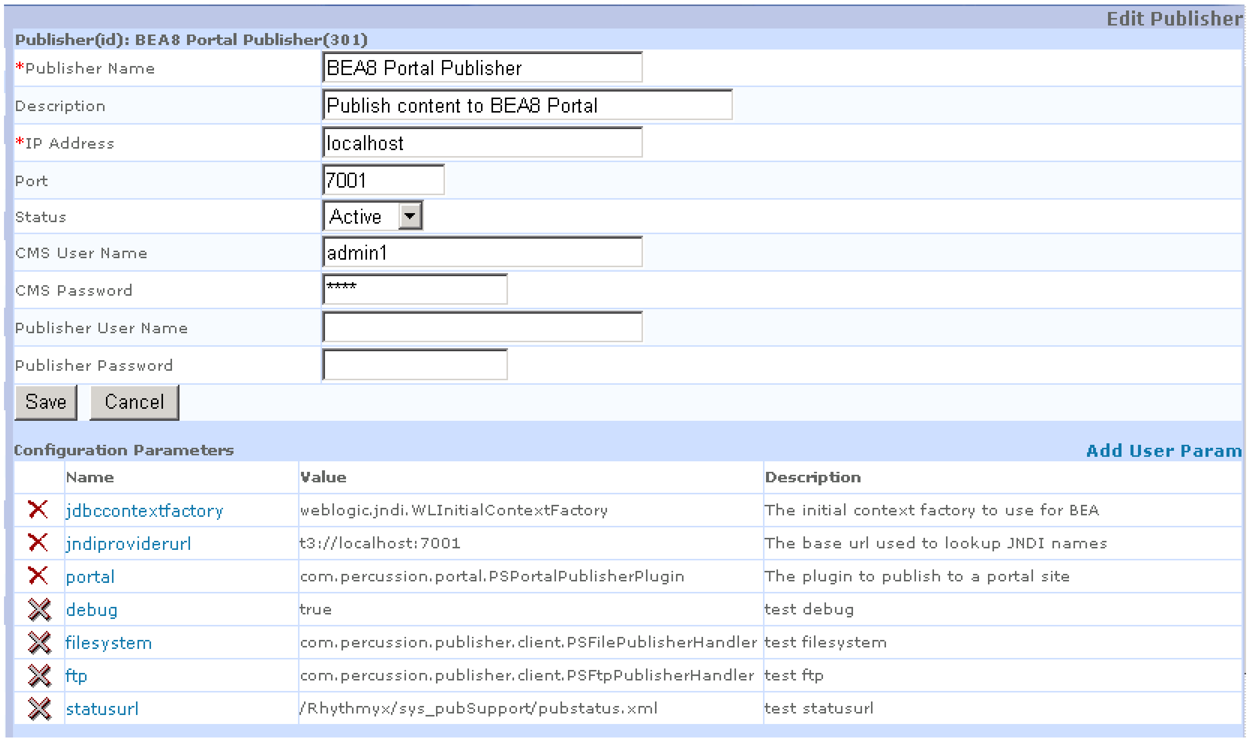
Registering Publishing Components

You must register the various publishing components for publishing to your Portal in Rhythmyx. Access Content Explorer directly from the Rhythmyx Server to register these components; they are not available through the WebLogic Rhythmyx Portal.

The sample publisher, site, edition, and content lists that you installed onto Rhythmyx using the Multi-Server Manager are examples of these registrations which you can use as models for your own.

Sample BEA Publisher

The following graphic shows the registration for the sample BEA Publisher:



Edit Publisher

Publisher(id): BEA8 Portal Publisher(301)

*Publisher Name	BEA8 Portal Publisher
Description	Publish content to BEA8 Portal
*IP Address	localhost
Port	7001
Status	Active
CMS User Name	admin1
CMS Password	****
Publisher User Name	
Publisher Password	

Configuration Parameters

Name	Value	Description
jdbccontextfactory	weblogic.jndi.WLInitialContextFactory	The initial context factory to use for BEA
jndiproviderurl	t3://localhost:7001	The base url used to lookup JNDI names
portal	com.percussion.portal.PSPortalPublisherPlugin	The plugin to publish to a portal site
debug	true	test debug
filesystem	com.percussion.publisher.client.PSFilePublisherHandler	test filesystem
ftp	com.percussion.publisher.client.PSFtpPublisherHandler	test ftp
statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	test statusurl

Figure 19: BEA8 Portal Publisher

To register your BEA Publisher, make the following changes:

1. Optionally change the Publisher Name and Description.
2. Change the IP Address to the address of your BEA Portal.
3. If your BEA port is not 7001, change Port to your current BEA port.

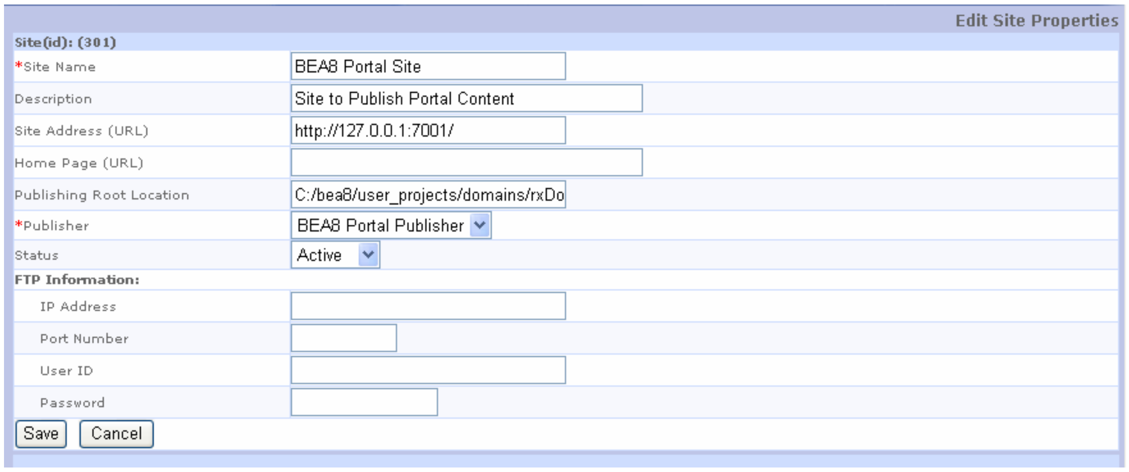
4 Change CMS User Name and CMS Password to your username and password. The sample password is demo.

5 Leave the user parameters added at the bottom of the page, but change values to correspond to your system:

Parameter	Function	Change Value?
portal	Tells the Publisher to use the Portal Publisher Plug-in.	Do not change value.
jdbccontextfactory	Specifies a JDBC context factory for the Publisher to use instead of the Tomcat default.	Do not change value.
jndiproviderurl	Specifies a JNDI provider URL for the Publisher to use instead of the Tomcat default.	If your BEA port is not 7001, change **Port** to your current BEA port.

Sample BEA Site

The following graphic shows the registration for the sample BEA 8.1 Portal Site. Note that this Site registration only contains information required for publishing to Portal repository since by default, our sample does not publish to filesystem.


Edit Site Properties

Site(id): (301)

*Site Name	BEA8 Portal Site
Description	Site to Publish Portal Content
Site Address (URL)	http://127.0.0.1:7001/
Home Page (URL)	
Publishing Root Location	C:/bea8/user_projects/domains/rxDo
*Publisher	BEA8 Portal Publisher
Status	Active
FTP Information:	
IP Address	
Port Number	
User ID	
Password	

Figure 20: BEA8 Portal Site

To register your BEA Portal repository Site, make the following changes:

1 Optionally, change the **Site Name** and **Description**.

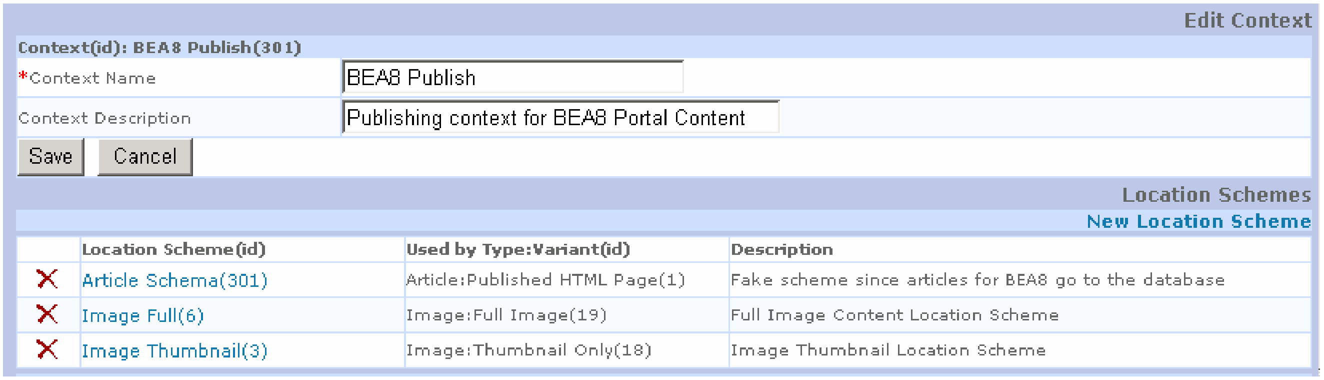
2 You must change the **Publishing Root Location** to specify the location from which static content will be served.

3 In the **Publisher** drop list, choose the publisher that you registered for BEA 8.1 Portal repository publishing.

Leave the value of **Status** as *Active*.

Sample BEA Location Schemes

The following graphic illustrates the publishing Context for BEA 8.1.



Edit Context

Context(id): BEA8 Publish(301)

*Context Name	BEA8 Publish
Context Description	Publishing context for BEA8 Portal Content

Save Cancel

Location Schemes

New Location Scheme

	Location Scheme(id)	Used by Type:Variant(id)	Description
✗	Article Schema(301)	Article:Published HTML Page(1)	Fake scheme since articles for BEA8 go to the database
✗	Image Full(6)	Image:Full Image(19)	Full Image Content Location Scheme
✗	Image Thumbnail(3)	Image:Thumbnail Only(18)	Image Thumbnail Location Scheme

Figure 21: BEA8 Publish Context

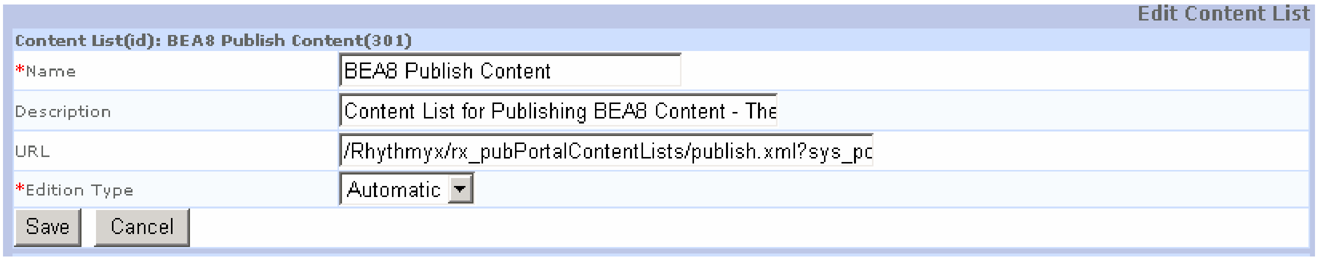
The Location Schemes installed with the BEA 8.1 Portal Connector support the sample Image Content Type. If you wish to publish other static content, you must either modify the existing Location Schemes or devise new schemes to publish the static content.

Sample BEA Content Lists

The following graphic shows the registration for the sample BEA Content List named BEA8 Publish Content. By default, it publishes the Article BEA8 Portal Variant to the Portal repository. The Content List's URL is:

```
/Rhythmyx/rx_pubPortalContentLists/publish.xml?sys_portal_variantid=36&s
ys_variantid=1&sys_contenttypeid=1
```

Notice that parameters specify both the Variant for publishing to the Portal repository (sys_portal_variantid=36) and the Variant of the content published to the DATA column of the ITEM table (sys_variantid=1).



Edit Content List

Content List(id): BEA8 Publish Content(301)

*Name	BEA8 Publish Content
Description	Content List for Publishing BEA8 Content - The
URL	/Rhythmyx/rx_pubPortalContentLists/publish.xml?sys_pc
*Edition Type	Automatic

Save Cancel

Figure 22: BEA8 Publish Content Content List

To register your Content Lists:

1. Create a separate Content List for each Variant that you are publishing.
2. Give each Content List an appropriate **Name** and **Description.**

   Enter the URL for publishing as:

   ```
    /<Rhythmyx
   root>/rx_pubPortalContentLists/[resourcename].xml?sys_portal_va
   riantid=[id#]&sys_variantid=[id#]
   &sys_contenttypeid=[id#]
   ```

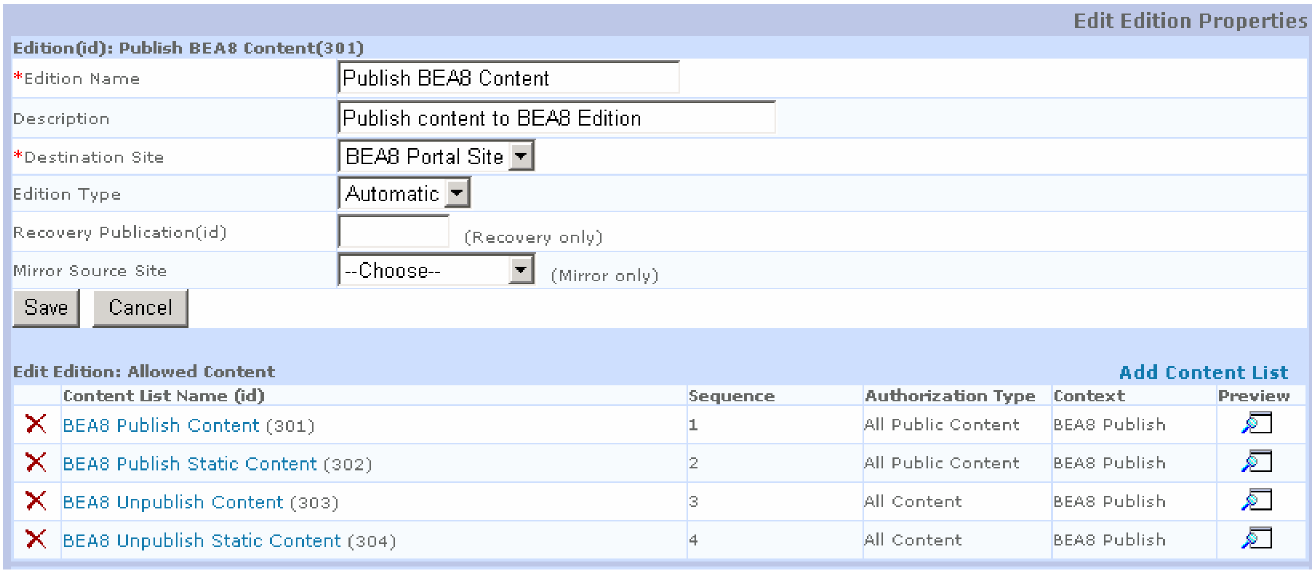
Enter the URL for unpublishing Portals:

```
/<Rhythmyx
root>/bea_pubContentLists/[resourcename].xml?sys_contenttypeid=
[id#]&sys_variantid=[id#]&sys_unpublish=yes
```

3 Leave the **Edition Type** as Automatic.

Sample BEA Editions

The following graphic shows the registration for the sample BEA Edition that publishes to the repository, the Content Lists associated with the Edition.


Edit Edition Properties

Edition(id): Publish BEA8 Content(301)

*Edition Name	Publish BEA8 Content
Description	Publish content to BEA8 Edition
*Destination Site	BEA8 Portal Site
Edition Type	Automatic
Recovery Publication(id)	(Recovery only)
Mirror Source Site	--Choose-- (Mirror only)

Edit Edition: Allowed Content

	Content List Name (id)	Sequence	Authorization Type	Context	Preview
✗	BEA8 Publish Content (301)	1	All Public Content	BEA8 Publish	🔍
✗	BEA8 Publish Static Content (302)	2	All Public Content	BEA8 Publish	🔍
✗	BEA8 Unpublish Content (303)	3	All Content	BEA8 Publish	🔍
✗	BEA8 Unpublish Static Content (304)	4	All Content	BEA8 Publish	🔍

Figure 23: Publish BEA8 Content Edition

To register your own Editions:

1 Enter an **Edition Name** and optionally a **Description.**

2 In the **Destination Site** drop list, choose the Site to which you want to publish the content.

3 In the **Edition Type** drop list, choose *Automatic* or *Manual*.

4 Add the Content Lists to publish with the edition.

Setting Up WebLogic DataSources

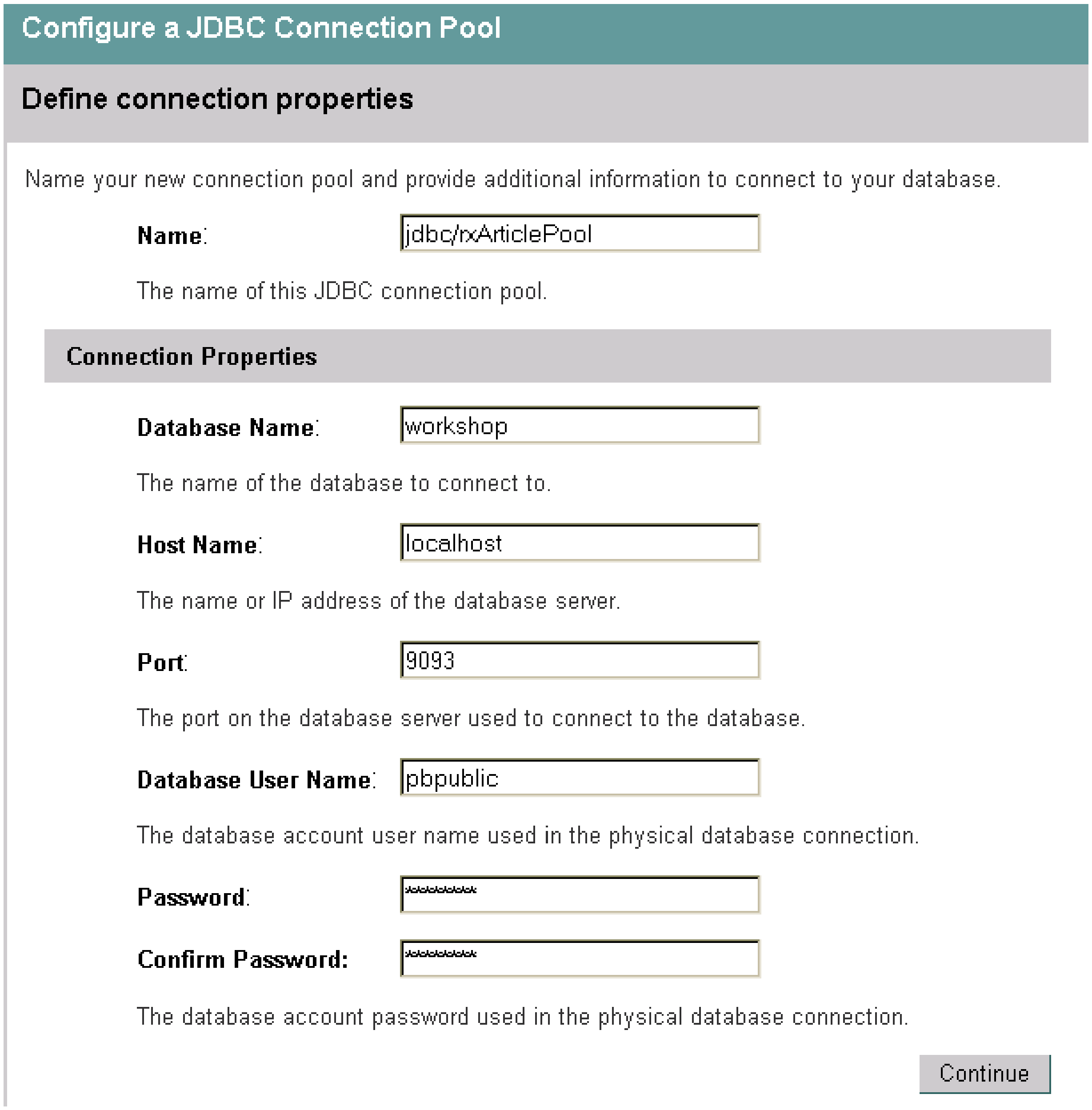
To publish to the BEA Portal repository from Rhythmyx and to use the Article Portlet, you must set up a data pool and a database source for your Rhythmyx database server on your WebLogic server. The data source name is defined in the dialogs shown in the following procedure.

To set up a database pool:

1. Open the WebLogic Server Console by starting a browser and entering: `http://localhost:7001/console`, or, in Windows, by clicking the [Start] button and selecting *Programs > BEA WebLogic Platform 8.1 > Examples > WebLogic Portal > Server Admin Console*.
2. On the Home Page, under portal/Services/JDBC, click Connection Pools.
3. Click [?] and follow the steps under JDBC Connection Pools/Configuring JDBC Connection Pools/Using the JDBC Connection Pool Assistant to create the Rhythmyx connection pool. Use the following information:
   - in the Choose Database screen, leave the Database Type as PointBase and choose the Database Driver Type 4X (unless you are using a different database);

NOTE: Rhythmyx supports only the Oracle Thin drivers. If you use Oracle as the Database Type, select Oracle Thin as the Database Driver.

- In the Define Connection Properties screen, if you are using the PointBase database, enter the values displayed in the following graphic for Name, Database Name, Port, and Database User Name. Enter pbpublic as the Password. Enter your local host in Host Name. If you are using a different database or have changed any of the default values, enter the correct values for your configuration.


Configure a JDBC Connection Pool

Define connection properties

Name your new connection pool and provide additional information to connect to your database.

Name: jdbc/rxArticlePool

The name of this JDBC connection pool.

Connection Properties

Database Name: workshop

The name of the database to connect to.

Host Name: localhost

The name or IP address of the database server.

Port: 9093

The port on the database server used to connect to the database.

Database User Name: pbpublic

The database account user name used in the physical database connection.

Password: ********

Confirm Password: ********

The database account password used in the physical database connection.

Figure 24: Define Connection Properties screen of the WebLogic Server Console

4 Click the [Continue] button and add the following properties:

If you use a SQL Server database:

- user
- PortNumber

- ServerName
- DatabaseName

If you use an Oracle database:

- TableSchema
- Origin
- Schema
- user
- DatabaseName

If you use a DB2 database:

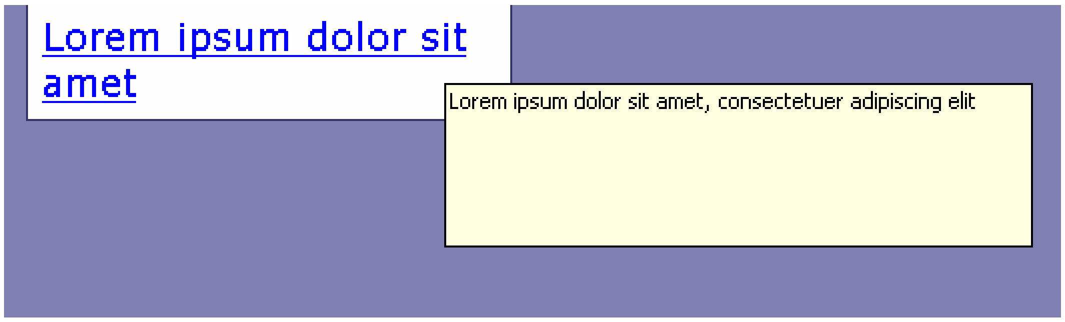
- TableSchema
- user
- DatabaseName

5. Return to the WebLogic Server Console Home Page, and under Services Configurations/JDBC, click Data Sources.
6. Click [?] and follow the steps under JDBC /Data Sources/Configuring JDBC Data Sources/Creating and Configuring a JDBC Data Source to create a JDBC Data Source. Use the following information:
   - In the Configure the Data Source Screen, enter any name for the data source in Name. The Article Portlet expects the JNDI Name jdbc/rxArticlePool. Enter this value in JNDI Name unless you are not using the Article Portlet or are modifying the JNDI Name in the Article Portlet. Uncheck Honor Global Transactions.
   - In the Connect to The Connection Pool screen, choose the Pool Name that you created in the beginning of this procedure.
   - In the Target the Data Source Screen, leave the default server (portalServer) checked.

Viewing Published Content

Log in to `http://localhost:7001/Rhythmyx` and publish your BEA Editions as you publish other Rhythmyx Editions. If you publish content to the BEA repository, you can view it through a Portlet in the BEA Portal.

The Portlet Connector includes the Article Portlet, which displays a list of Article titles that you have published to your Portal repository. When you hover your mouse over a title, a pop-up displays the Article abstract.

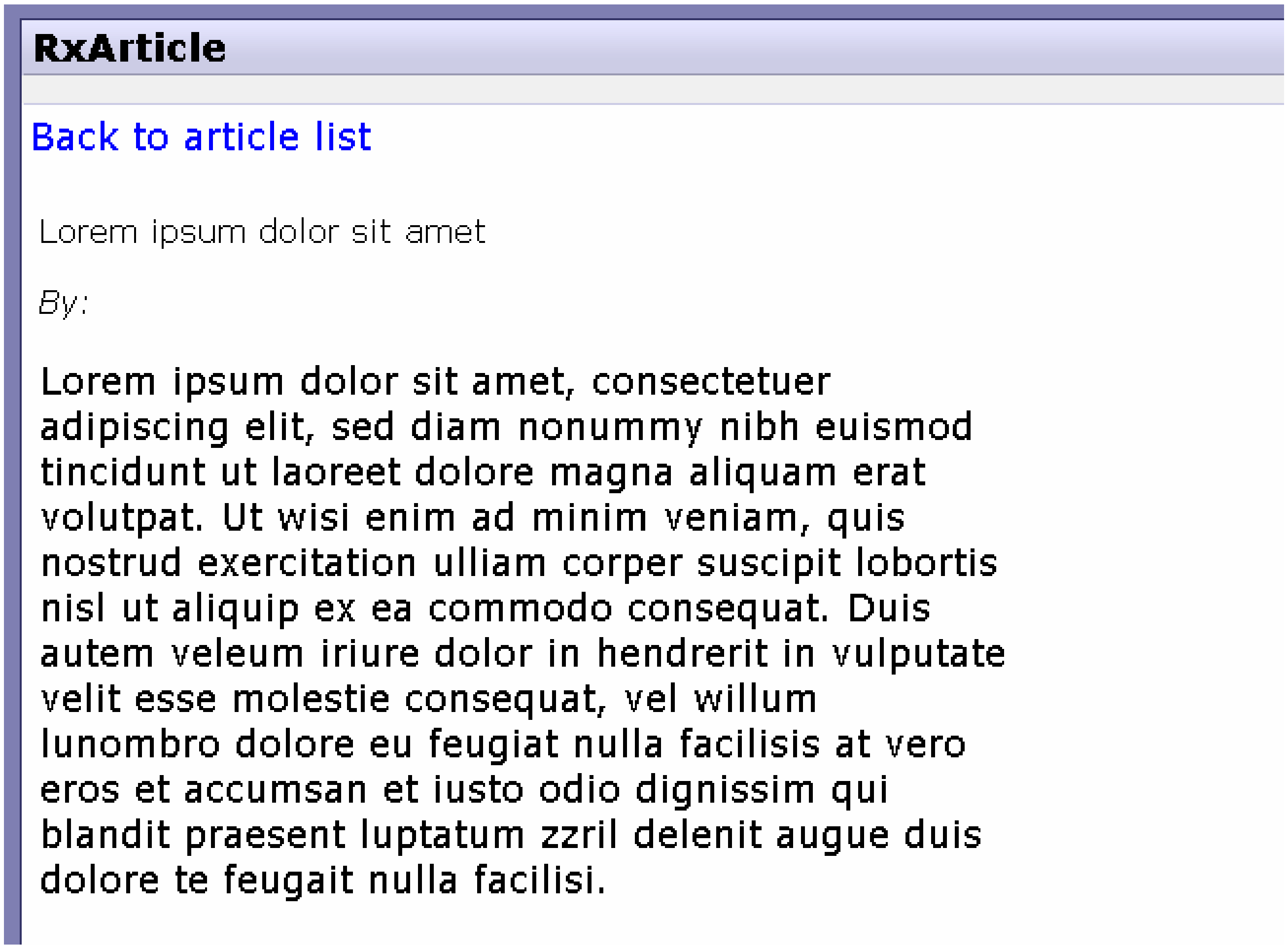


Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

Figure 25: Title listed in Article Portlet

The article title is a link that displays the item in the Published HTML Variant, which includes the article body.



Lorem ipsum dolor sit amet

By:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exercitation ulliam corper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem veleum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel willum lunombro dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Figure 26: Article Portlet displaying Article's body

The Article Portlet is an example of how you can view contents published to your Portal repository. You can modify this Portlet to view custom Content Types that you have published to the Portal repository.

Best Practices for Delivery

If you are planning to present data as formatted when you retrieve it from the Portal repository, assemble it in Rhythmyx and publish it in one or more Variant formats. The most efficient model for Web content maintenance stores dynamic data in the repository and static content in a file system. To implement this model, use the BEA 8.1 Portal Connector to publish your Web Site's dynamic content to the BEA Portal repository, and use a Rhythmyx file system Publisher to publish your Web Site's static content to a file system.

CHAPTER 6

Reference

sys_PortalPublishing.DTD

Drag and drop this DTD onto the Content List application in the Rhythmyx Workbench to create the content lists used with the BEA 8.1 publisher plug-in.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XMLSPY v5 rel. 4 U (http://www.xmlspy.com)-->
<!--This is the element that will be published to the portal publisher
when it requests the content varient from the content list-->
<!ELEMENT PSXPortalPublisher (ConnectionInfo, Content, MetaData?,
AccessControl?)>
<!--Each property specifies a single associated value. The type
specifies the type of the value of this element.-->
<!ELEMENT Property EMPTY>
<!ATTLIST Property
   type (string | integer | date) "string"
   name CDATA #REQUIRED
   pattern CDATA #REQUIRED
>
<!--Connection info holds information about making the database
connection-->
<!ELEMENT ConnectionInfo (Target+)>
<!--Holds the actual content being published. This will normally be the
result of the assembly of a variant. -->
<!ELEMENT Content EMPTY>
<!--target: Publish the content to this target. The target must be in
the list of specified targets under ConnectionInfo.-->
<!--encoding: The encoding of the contained value. This is most likely
base64 to avoid issues with embedded HTML not complying with XHTML and
therefore invalidating the containing XML document.-->
<!--id: Generally this id will be the Rhythmyx content id, but it may be
any unique id that will remain constant across publishing sessions.-->
<!ATTLIST Content
   target CDATA #REQUIRED
   encoding (none | base64 | escaped) "none"
   id CDATA #REQUIRED
>
<!--Metadata represents the extra information about this content item.
It includes both viewable information such as titles and abstracts, as
well as non-viewable information such as regions, categories and mime
types.-->
<!ELEMENT MetaData (Property*)>
<!--Information about restricting access to this content item. Not
currently used.-->
<!ELEMENT AccessControl (Property*)>
<!--Each target specifies a single specification of a database
connection. Each connection can be to either a jndi or jdbc datasource.
The child elements are required for the jdbc datasource.-->
<!ELEMENT Target (User?, Password?, Driver?, Class?, Schema?, Server?,
URL?)>
<!--Type should be either jdbc or jndi-->
<!ATTLIST Target
   name CDATA #REQUIRED
   type CDATA #REQUIRED
>
<!--The user name for the specified database. Must correspond to the
password given. This is ignored for a JNDI resource.-->
<!ELEMENT User ANY>
<!--Corresponding password for given user. This is ignored for a JNDI
resource.-->
<!ELEMENT Password ANY>
<!--The JDBC Driver name, this is generally the second component from
the url for your jdbc driver, e.g. pointbase. Sometimes this contains
another component, e.g. oracle:thin.-->
<!ELEMENT Driver ANY>
<!--The exact java classname of the JDBC driver. See your database
supplier's documentation for the class information (or your JDBC
driver's documentation if it is a third party driver). Required for
JDBC. This is ignored for a JNDI resource.-->
<!ELEMENT Class ANY>
<!--Optional information to specify what database schema is in use.
Required for Oracle.-->
<!ELEMENT Schema ANY>
<!--The database specifies the name of  the server to connect to for a
JDBC driver. This is ignored for a JNDI resource.-->
<!ELEMENT Server ANY>
<!--This specifies the name of the JNDI resource being used. Ignored for
JDBC.-->
<!ELEMENT URL ANY>
```

sys_casCreatePortalPropertyList

Context:
global/percussion/assemblers/

Description:
This exit returns optional metadata properties that the BEA 8.1 Portal Connector publishes to the ITEM_METADATA table in the BEA repository.

Class name:
com.percussion.cas.PSAddPortalProperties

Interface:
com.percussion.extension.IPSUdfProcessor

Parameters:

Name	**Data Type**	**Description**
elementname	java.lang.String	The XML element that holds the database properties returned by this exit. (The element that sys_casCreatePortalPropertyList is mapped to in the assembler application.)
name[0]	java.lang.String	The property name. May not be NULL or empty when defining a property. Parsing of properties stops at the first NULL or empty property name.
type[0]	java.lang.String	The property type. Required if a valid property name is specified. Valid types are string, numeric, date, clob, and binary.
pattern[0]	java.lang.String	Optional format pattern if the type is date. Ignored if the type is not date. Default is yyyy-MM-dd. Valid patterns are specified in java.text.SimpleDateFormat.
value[0]	java.lang.String	The property value. May be a string, NULL, empty, or a comma-separated list.
name[1]	java.lang.String	The property name. May not be NULL or empty when defining a property. Parsing of properties stops at the first NULL or empty property name.
type[1]	java.lang.String	The property type. Required if a valid property name is specified. Valid types are string, numeric, date, clob, and binary.
pattern[1]	java.lang.String	Optional format pattern if the type is date. Ignored if the type is not date. Default is yyyy-MM-dd. Valid patterns are specified in java.text.SimpleDateFormat.
value[1]	java.lang.String	The property value. May be a string, NULL, empty, or a comma-separated list.
. . .		
name[9]	java.lang.String	The property name. May not be NULL or empty when defining a property. Parsing of properties stops at the first NULL or empty property name.

Name	Data Type	Description
type[9]	java.lang.String	The property type. Required if a valid property name is specified. Valid types are string, numeric, date, clob, and binary.
pattern[9]	java.lang.String	Optional format pattern if the type is date. Ignored if the type is not date. Default is yyyy-MM-dd. Valid patterns are specified in java.text.SimpleDateFormat.
value[9]	java.lang.String	The property value. May be a string, NULL, empty, or a comma-separated list.

ITEM and ITEM_METADATA Tables

The Portal Connector publishes to these tables.

ITEM Table

Name	Type	Parameters	Allows null values	Description
ID	INT	4	no	Unique ID. Primary Key.
DATA	TEXT or CLOB	See your database documentation	no	Body content.

ITEM_METADATA Table

Name	Type	Parameters	Allows null values	Description
ID	INT	4	no	Unique ID. Forms Primary Key with SEQ.
SEQ	INT	4	no	Unique sequence number. Forms Primary Key with ID.
PROPERTY	NVARCHAR	32	no	Property name.
STRING_VAL	NVARCHAR	See your database documentation .	yes	String value for property if property value is expressed as character or text.
INT_VAL	INT	4	yes	Integer value for property if property value is expressed as a number.
DATE_VAL	DATE	---	yes	Date value for property if property value is expressed as a date.
CLOB_VAL	CLOB	---	yes	Large string value.
BLOB_VAL	BLOB	---	yes	Large binary value.

Indexes:

To decrease response time for personalization queries in the Article Portlet, create indexes such as the following:

- RX_ITEM_PROP ON ITEM_METADATA(PROPERTY)
- RX_ITEM_STR_VAL ON ITEM_METADATA(STRING_VAL)
- RX_ITEM_INT_VAL ON ITEM_METADATA(INT_VAL)
- RX_ITEM_DATE_VAL ON ITEM_METADATA(DATE_VAL)

Indexes increase database size and decrease the speed of record insertion; if you do not require some of the fields for searching, do not create the associated indexes.

CHAPTER 7

Troubleshooting

Publishing Logs: For help resolving publishing errors, check BEA's publishing logs in: `<bea root>\weblogic81\samples\domains\portal\webapps\publogs`.

Problem Resolution:

Problem: Browser hangs after invoking an action that launches a new browser window.

Resolution: This problem is caused by the WebLogic's server's memory usage. The only known solution is to reboot the server.

Problem: Page fails to load when refreshed after making a Workflow Transition or user receives a "not authorized" error message after making a Workflow Transition.

Resolution: These errors indicate that the current browser session has timed out. No message is displayed indicating that the session has timed out. Terminate the existing browser session and start a new session.

APPENDIX I

Appendix: For Users Migrating from BEA 7.0 to BEA 8.1

If you have migrated from BEA WebLogic 7.0 to 8.1 and have been using Rhythmyx's BEA WebLogic Portal Connector to maintain a Rhythmyx Portal in WebLogic 7.0, you must install Rhythmyx's BEA Connector for WebLogic 8.1 and reconfigure your Rhythmyx Portal on WebLogic 8.1. The components of the Rhythmyx Connector for 7.0 will not function properly on WebLogic 8.1.

The theory behind the 7.0 and 8.1 Connector is the same; however, most of the installation and setup instructions have changed, and the components provided by the WebLogic 8.1 Connector are different than those provided by the 7.0 Connector. This appendix highlights what has changed.

Major changes between 7.0 and 8.1 Portal Connectors:

- The BEA 7.0 Connector did not include a sample Rhythmyx Portal, and we instructed users to use the WebLogic Sample Portal. The BEA 8.1 Connector includes a sample Rhythmyx Portal, rxportal.
- The BEA 7.0 Connector used port 7501 by default; the BEA 8.1 Connector uses port 7001 by default.
- The BEA 7.0 Connector provided sample BEA Content Types with their own Content Editor and Content Assembler applications, Variants, and Slots. The BEA 8.1 Connector uses default Rhythmyx Content Types and the corresponding default Content Editors, Content Assemblers, Variants, and Slots that Rhythmyx provides.

  The only sample Variant that the BEA 8.1 Connector provides is Article BEA8 Portal Variant, which formats the data for the BEA Portal Repository tables. The only sample assembler that the BEA 8.1 Connector provides is rx_casPortalContent, which assembles data using this Variant.

  See "Rhythmyx Setup Steps" for a list of the sample components that the BEA 8.1 Portal Connector installs.

- The BEA 7.0 Connector included the Content List Application bea_pubContentLists. The BEA 8.1 Connector includes the Content List Application rx_pubPortalContentLists. bea_pubContentList used the delivery type “bea” which indicated that the Rhythmyx Publisher should use the BEA 7.0 Portal Plugin. rx_pubPortalContentLists uses the delivery type “portal” which indicates that the Rhythmyx Publisher should use the BEA 8.1 Portal Plugin. rx_pubPortalContentList includes fewer mappings because the Portal repository tables in 8.1 include fewer fields, and the @unpublish attribute is not included (you can specify unpublishing in the Selector of a separate unpublishing resource).
- The process for installing Web applications is different in WebLogic 8.1 and WebLogic 7.0. The topic “Deploying the Rhythmyx Publisher and Web Services” reflects these changes.
- In WebLogic 8.1 you can modify Deployment Descriptors in the WebLogic Builder instead of the Administration Console. The topic “Modifying the Rhythmyx Web Application Deployment Descriptor” reflects this change.
- In WebLogic 8.1, the Rhythmyx Portlets and Portals are include in the file rxPortal.ear and are deployed to WebLogic using the WebLogic Server Administration Console Deploy a New Application procedure. In WebLogic 7.0, the Rhythmyx Portlets were unzipped from the file portlets.zip into the Sample Portal folder. The topic “Deploying the Rhythmyx Portal and Portlets to WebLogic” reflects this change.
- The Article Portlet functions slightly differently in BEA WebLogic 8.1 and BEA WebLogic 7.0. See the topic “Determining Which Portlets to Use in Your System” for a description of the way the Article Portlet functions in 8.1.
- In BEA 7.0, the Publisher plugin published content to file system and metadata to the Portal repository using the default BEA tables DOCUMENT and DOCUMENT_METADATA. The BEA 8.1 Publisher plugin publishes both content and metadata to the Portal repository and uses the sample tables provided by the Portal Connector, ITEM and ITEM_METADATA. See the topic “ITEM and ITEM_METADATA tables” for the ITEM and ITEM_METADATA table schemas.
- The sample Publishing component registrations provided by the BEA 8.1 Connector differ than those provided by the BEA 7.0 Connector and reflect the changes in the Content Types and applications used in the example. See “Registering Publishing Components” for the new registrations.
- The BEA 7.0 Connector Variant Assembler application included resources for assembling the Variant written to the Portal Repository and a resource for assembling the body content written to file system. The BEA 8.1 Connector Variant Assembler application only includes resources for assembling the Variant that is written to the Portal Repository. The resource for assembling the body content is included in the default Rhythmyx assembler, casArticle. See the section “Creating the Assembler Applications” for information about the BEA 8.1 Connector assemblers.
- BEA 8.1 provides slightly different screens for entering WebLogic Data Sources, and the 8.1 Portal Connector requires some different default entries. See “Setting Up WebLogic Data Sources” for the method of entering Data Source values in 8.1 and the values to enter.

- The 8.1 Portal Connector uses sys_PortalPublishing.dtd to create the Variant published to the Portal repository; the 7.0 Portal Connector used sys_PortalPublisher.dtd. Although the DTDs include the same main elements, they use them differently. sys_portalPublisher.dtd did not use the AccessControl element, but sys_PortalPublishing.dtd uses it to specify connection through a jndi or jdbc resource. sys_portalPublisher.dtd used the target element to specify different delivery locations, but sys_PortalPublishing.dtd always uses it to specify a database. See sys_PortalPublishing.dtd to view all of the differences between sys_portalPublisher.dtd and sys_PortalPublishing.dtd.
- The BEA 7.0 Connector uses the extension sys_casAddPortalProperties, which returned metadata properties that were published to DOCUMENT_METADATA. The BEA 8.1 Connector uses the extension sys_casCreatePortalPropertyList which returns the metadata properties that are published to ITEM_METADATA.

Index