
Rhythmyx

Implementing Publishing

Version 5.7

Copyright and Licensing Statement

All intellectual property rights in the SOFTWARE and associated user documentation, implementation documentation, and reference documentation are owned by Percussion Software or its suppliers and are protected by United States and Canadian copyright laws, other applicable copyright laws, and international treaty provisions. Percussion Software retains all rights, title, and interest not expressly granted. You may either (a) make one (1) copy of the SOFTWARE solely for backup or archival purposes or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You must reproduce and include the copyright notice on any copy made. You may not copy the user documentation accompanying the SOFTWARE.

The information in Rhythmyx documentation is subject to change without notice and does not represent a commitment on the part of Percussion Software, Inc. This document describes proprietary trade secrets of Percussion Software, Inc. Licensees of this document must acknowledge the proprietary claims of Percussion Software, Inc., in advance of receiving this document or any software to which it refers, and must agree to hold the trade secrets in confidence for the sole use of Percussion Software, Inc.

The software contains proprietary information of Percussion Software; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between Percussion Software and the client and remains the exclusive property of Percussion Software. If you find any problems in the documentation, please report them to us in writing. Percussion Software does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Percussion Software.

Copyright © 1999-2005 Percussion Software.
All rights reserved

Licenses and Source Code

Rhythmyx uses Mozilla's JavaScript C API. See <http://www.mozilla.org/source.html> (<http://www.mozilla.org/source.html>) for the source code. In addition, see the *Mozilla Public License* (<http://www.mozilla.org/source.html>).

Netscape Public License

Apache Software License

IBM Public License

Lesser GNU Public License

Other Copyrights

The Rhythmyx installation application was developed using InstallShield, which is a licensed and copyrighted by InstallShield Software Corporation.

The Sprinta JDBC driver is licensed and copyrighted by I-NET Software Corporation.

The Sentry Spellingchecker Engine Software Development Kit is licensed and copyrighted by Wintertree Software.

The Java™ 2 Runtime Environment is licensed and copyrighted by Sun Microsystems, Inc.

The Oracle JDBC driver is licensed and copyrighted by Oracle Corporation.

The Sybase JDBC driver is licensed and copyrighted by Sybase, Inc.

The AS/400 driver is licensed and copyrighted by International Business Machines Corporation.

The Ephox EditLive! for Java DHTML editor is licensed and copyrighted by Ephox, Inc.

This product includes software developed by CDS Networks, Inc.

The software contains proprietary information of Percussion Software; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between Percussion Software and the client and remains the exclusive property of Percussion Software. If you find any problems in the documentation, please report them to us in writing. Percussion Software does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Percussion Software.

AuthorIT™ is a trademark of Optical Systems Corporation Ltd.

Microsoft Word, Microsoft Office, Windows®, Window 95™, Window 98™, Windows NT® and MS-DOS™ are trademarks of the Microsoft Corporation.

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

Schema documentation was created using XMLSpy™.

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

Publishing in Rhythmyx 5

How the Publishing Process Works	6
Setting Up Rhythmyx for Publishing	7
Configuring Publishing Components in the User Interface	8
Rhythmyx Publishing Administrator	9
Rhythmyx Publishing Manager	10
Rhythmyx Publishers	11
Installing and Configuring Publishing System Components	13
Installing a Remote Rhythmyx Publisher	13
Registering a Publisher	19
Configuring a Publisher	21
Setting Up Scheduled Publishing	28
Publishing with SSL	29

Sites 39

Site Maintenance Dialogs	40
Edit Site Properties Page	40
Copy Site Page	42
Site Maintenance Procedures	43
Defining a New Site	43
Copying a Site Registration	44
Editing a Site	44
Deleting a Site	44
Viewing a Virtual Edition Map	45

Content Lists 47

Creating a Content List Application	48
Mapping a Content List Resource	49
Publishing from Different States	54
Mirror Editions	54
Registering Content Lists	58
Edit Content List Page	58
Registering a New Content List	59
Deleting a Content List Registration	59
Editing a Content List Registration	59
Incremental Publishing	60
Incremental Content Lists	61
sys_IncrementalContentFilter	62

Editions 65

Edition Types.....66
 Publishing an Edition.....67
 Edit Edition Properties Page.....68
 Defining a New Edition.....69
 Edition Content Lists.....70
 Adding Content Lists to an Edition.....70
 Editing a Content List Assignment to an Edition.....71
 Previewing Content List Output.....71
 Deleting a Content List From an Edition.....71
 Editing an Edition.....72
 Deleting an Edition.....73

Implementing a Custom AuthType 75

Adding new Keywords.....76
 Creating a Content Assembly Support Application.....77
 Maintaining the Authtypes.properties file.....80

Developing and Managing Contexts and Link Generation Schemes 81

Context.....82
 Context Editor.....83
 Adding Contexts and Location Schemes.....83
 Editing Contexts and Location Schemes.....84
 Deleting Contexts and Location Schemes.....84
 Developing and Managing Link Generation Schemes.....85
 Default Location Scheme Generators.....85
 Defining Scheme Generators for Contexts.....86
 Editing Scheme Generators in Contexts.....87
 Deleting Scheme Generators in Contexts.....87
 Location Scheme Parameters.....88
 Adding Scheme Generator Properties.....89
 Editing Scheme Generator Properties.....90
 Deleting Scheme Generator Properties.....90

Defining Complex Location Schemes	91
Reviewing Publications	93
Publication Details: Publication Maps	95
Viewing Detailed Logs	97
Purging the Publication Log	99
Monitoring Publication of Localized Content	100
Republishing Failed Content	101
Implementing Plugins	103
Implementing a User-Created Plugin.....	104
Registering a Publisher Plugin.....	105
Including the Java Class in the Tomcat Server File.....	106
Mapping to the Plugin in a Content List Resource	107
Appendix: Troubleshooting	109
Troubleshooting Publishing.....	110
Errors	111
Incorrect links to published files on Web server.....	112
Access is Denied Error	113
Publication Error Log is Empty	114
Files Do Not Render Properly in Browser.....	115
Error Opening Socket	116
RXSITEITEMS Is Updating Incorrectly	118
Content is Publishing without Related Content	119
Error Logs in the Publisher	120
Troubleshooting Content Lists.....	121
Publishing FAQs	123
Index	125

CHAPTER 1

Publishing in Rhythmyx

Publishing is the final phase of the content management process. Publishing extracts the content item data from the database, merges it with formatting to produce a final output page, and saves the final output page in its delivery location.

How the Publishing Process Works

After you have installed the Publishing components, created content list applications, and set up a Publisher, Site, Content Lists and an Edition in the user interface, you can publish the Edition.

Manually initiate a publishing request by clicking the [**Publish**] button next to an Edition on the Editions page or schedule automatic publishing. Rhythmyx passes the publishing request to the Publishing Manager, which acts as a centralized controller for all of the system's Publishers. The Publishing Manager selects the Publisher to run the job, and passes a request to that Publisher. The Publisher activates Content List applications to extract content items from the database then activates Content Assembler applications to merge the content items with formatting to produce the final content pages. The Publisher saves the final pages to the file system or transfers them via FTP to their final destination and sends status information back to the Publishing Manager. The Publishing Manager updates the Rhythmyx tables.

Setting Up Rhythmyx for Publishing

Before you can publish content in Rhythmyx, you must install the Publishing Manager and a Publisher application. You can install Publishers in other locations (remote Publishers) and set up scheduled publishing.

Configuring Publishing Components in the User Interface

After you set up Publishing in your Rhythmyx System, configure Publishers, Sites, Content Lists, and Editions in the user interface. To create an edition, you must first create Sites and Content Lists.

Rhythmyx Publishing Administrator

Use the browser-based Publishing Administrator to design and configure all publishing components, such as Publishers, Editions, and Content Lists, and to initiate publishing manually. The Publishing Administrator includes views that display all publication runs and virtual Publication maps.

Content List Administration		New Content List	
Content List(id)	URL	Edition Type	
✗ Manual Content List(316)	/Rhythmyx/rx_pubPreviewEdition/contentlist_manual.xml?sys_editionid=313&pubOp=pub		
✗ Mirror Full(301)	/Rhythmyx/rx_Support_pub/mirror_dist.xml?delivery=filesystem		
✗ Mirror Incremental(302)	/Rhythmyx/rx_Support_pub/mirror_dist.xml?inc=y,i&delivery=filesystem		
✗ Mirror Unpublish(303)	/Rhythmyx/rx_Support_pub/mirror_unpub_dist.xml?delivery=filesystem		
✗ Site Root Full(310)	/Rhythmyx/rx_Support_pub/folder_dist.xml?valid=y,i&delivery=filesystem&pubOp=pub		
✗ Site Root Incremental(311)	/Rhythmyx/rx_Support_pub/folder_dist.xml?valid=y,i&inc=y&delivery=filesystem&pubOp=pub		
✗ Unpublish(315)	/Rhythmyx/rx_Support_pub/unpub_dist.xml?delivery=filesystem		

Figure 1: Rhythmyx Publisher Administrator

Rhythmyx Publishing Manager

The Rhythmyx Publishing Manager is the central controller of all publishing activities in the Rhythmyx Publishing System. Check the *Rhythmyx Publisher* option to include it as part of the Rhythmyx server during the initial installation process.

Once publishing is initiated, Rhythmyx sends the request to the Publishing Manager. The Publishing Manager compiles lists of content items to publish, and determines the site where the content will be published and the Publisher that will create the Publication. After it compiles each content list, the Publishing Manager issues an HTTP request via Simple Object Access Protocol (*SOAP*¹) to the location where the Publisher resides. The request includes the compiled content list and directs the Publisher to publish it.

¹ Simple Object Access Protocol (SOAP) lets programs in the same or different operating systems communicate using Hypertext Transfer Protocol (HTTP) and XML. Because firewalls do not usually block HTTP requests, systems use SOAP to ensure transmission of requests.

Rhythmyx Publishers

A Rhythmyx Publishing system includes one or more Publishers. Publishers are applications that reside on the Rhythmyx server or one or more remote locations, and run off a Tomcat Web server.



The screenshot shows a web interface titled "Publishers" with a "New Publisher" link. Below is a table with the following data:

Publisher(id)	Description	IP Address	Status
 MainPublisher(304)	Publisher for company Web site.	www.WebSite.com	Active

Figure 2: Publisher registered in Publisher Administrator

The Publisher runs after it receives a request from the Publishing Manager. When the Publishing Manager receives an initial publishing request, it sends each content list in the Edition to the Publisher via an SOAP.

After receiving a request, the Publisher

- retrieves the content items specified on the content list from the Rhythmyx server;
- calls the content assembler to generate the final page(s);
- sends the page(s) to the publishing destination (using HTTP or FTP depending on the delivery type mapped to the content list resource); and finally
- sends a status document specifying the content items processed and any errors encountered back to the Publishing Manager via SOAP.

The following graphic illustrates this process:

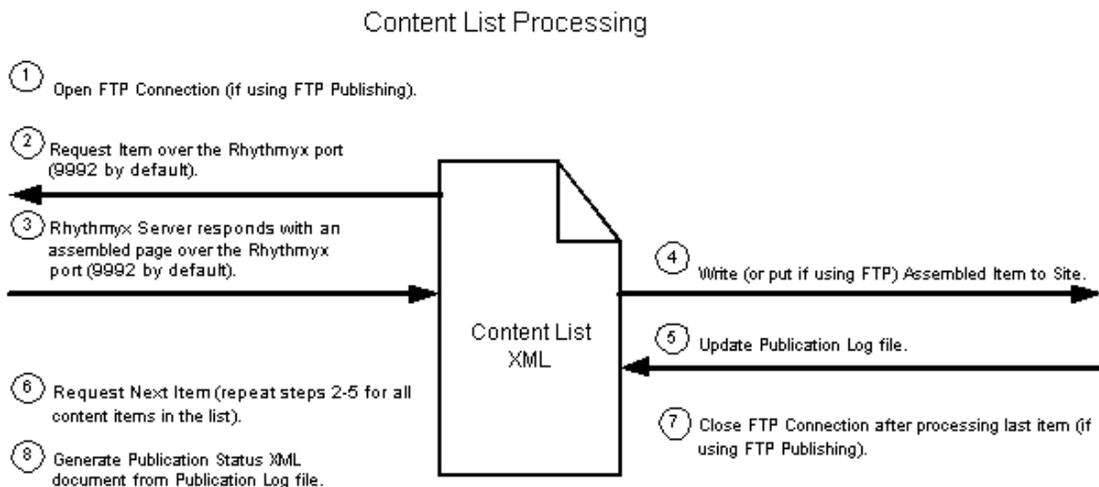
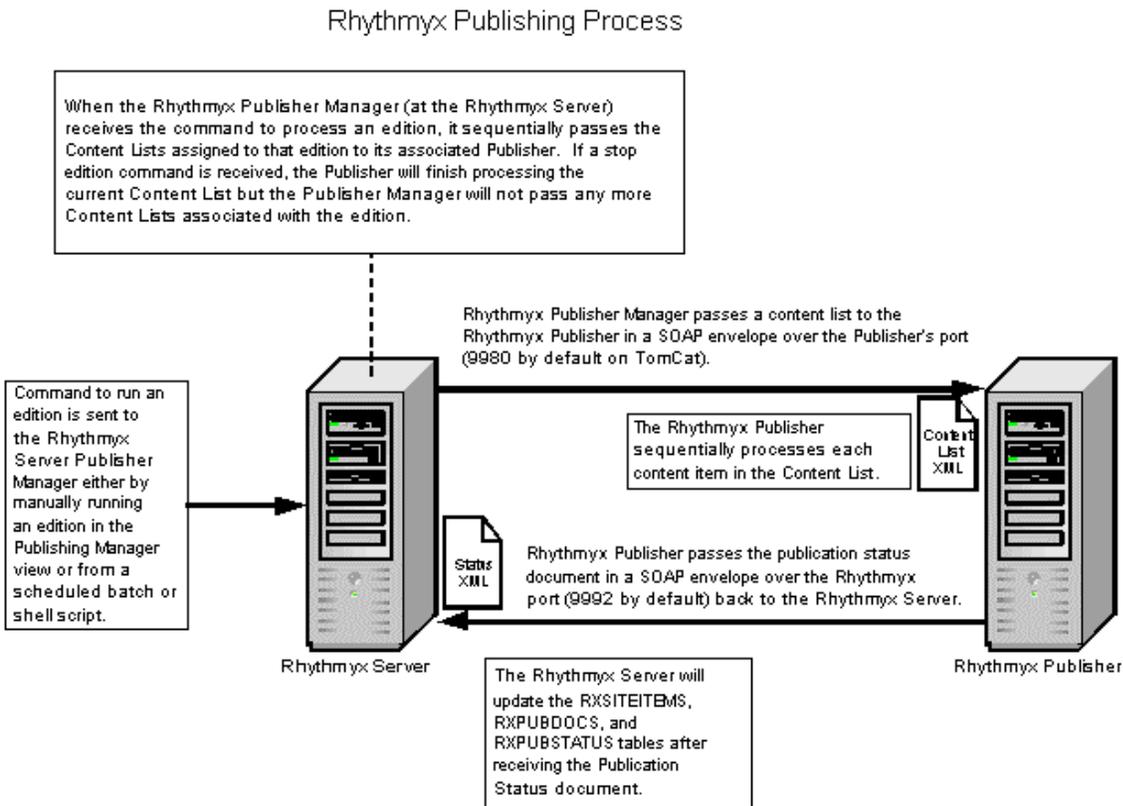


Figure 3: Rhythmyx Publishing Process

Installing and Configuring Publishing System Components

Installing a Remote Rhythmyx Publisher

The following tasks all take place on the server housing the Rhythmyx Publisher.

- 1 Locate the installer for the Rhythmyx Publisher on the Installation CD. For Windows systems, the executable is named PublisherSetup.exe and is located in the windows directory.

Note: Shell scripts for Linux and Solaris are contained in their corresponding directories

- 2 Execute the Publisher Setup.
- 3 At the Welcome screen press [Next].



Figure 4: Welcome

- 4 Read and accept the License Agreement. Press [Next].

- 5 If this is a new install, select the new install radio button. Otherwise select upgrade to upgrade an existing Publisher. Press [Next].

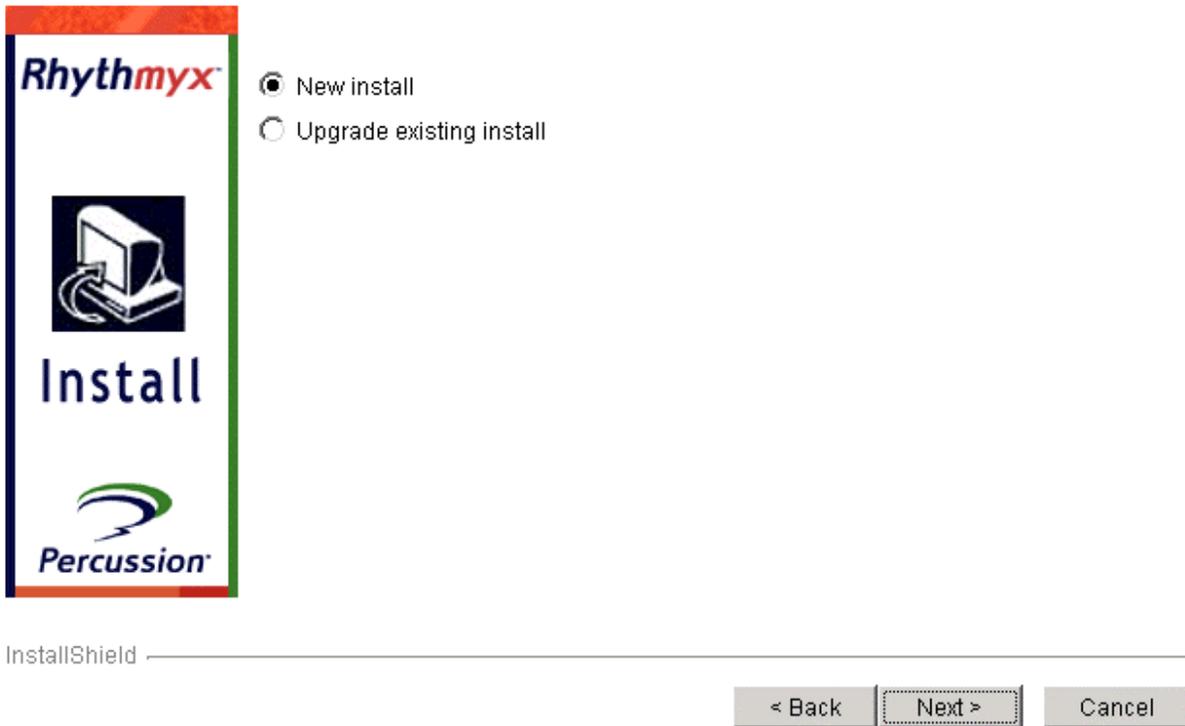


Figure 5: Install Type

- 6 Select a directory to install the publisher. By default the Installer chooses C:\Rhythmyx. Press [Next].
- 7 Note: If the target directory does not exist, the Installer will prompt to confirm creation of this new directory. Select [OK] to create the directory.

- 8 The next screen allows for the installation of additional features. As this procedure is focused primarily on filesystem and FTP publishing, we will not select any additional features. Press [Next].

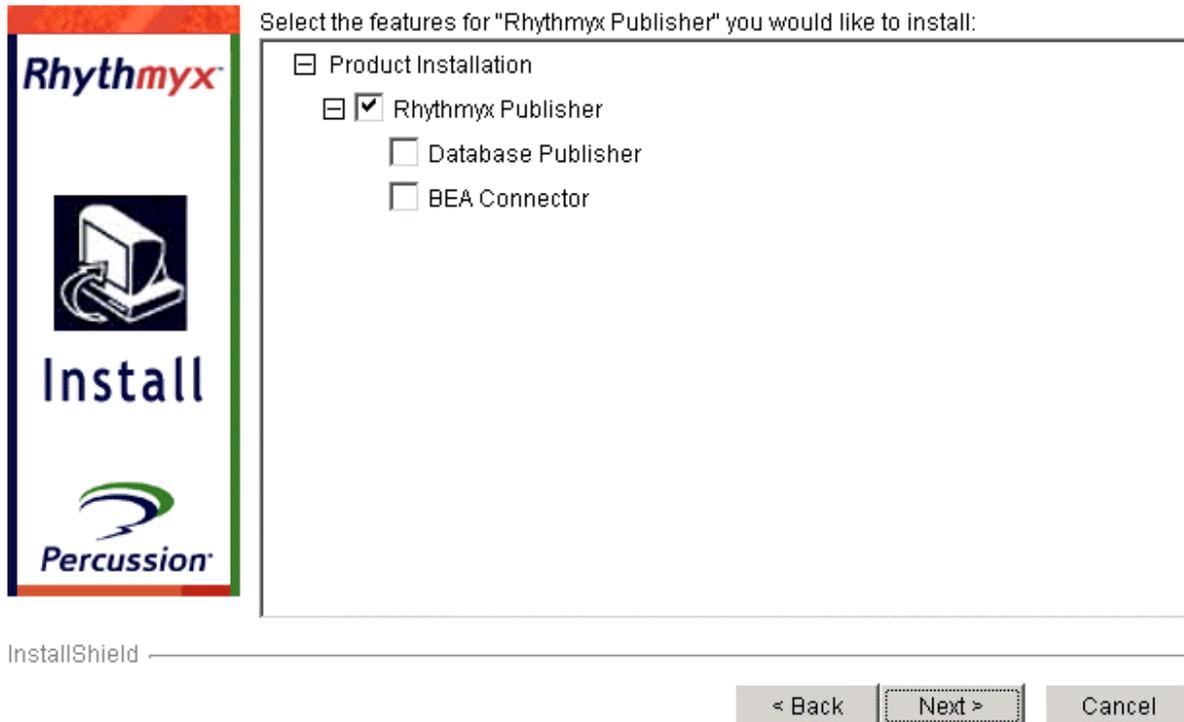


Figure 6: Rhythmyx Features

- 9 Select the Rhythmyx Publisher as the Web application being deployed into the Tomcat Server. Press [Next].

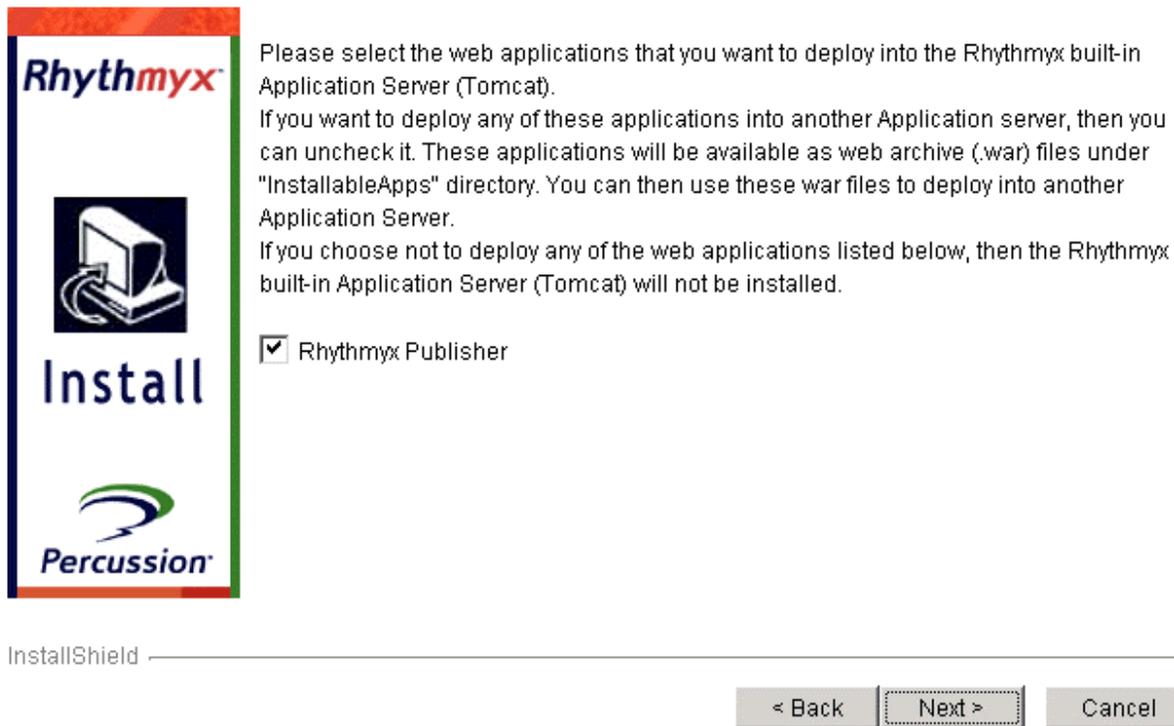
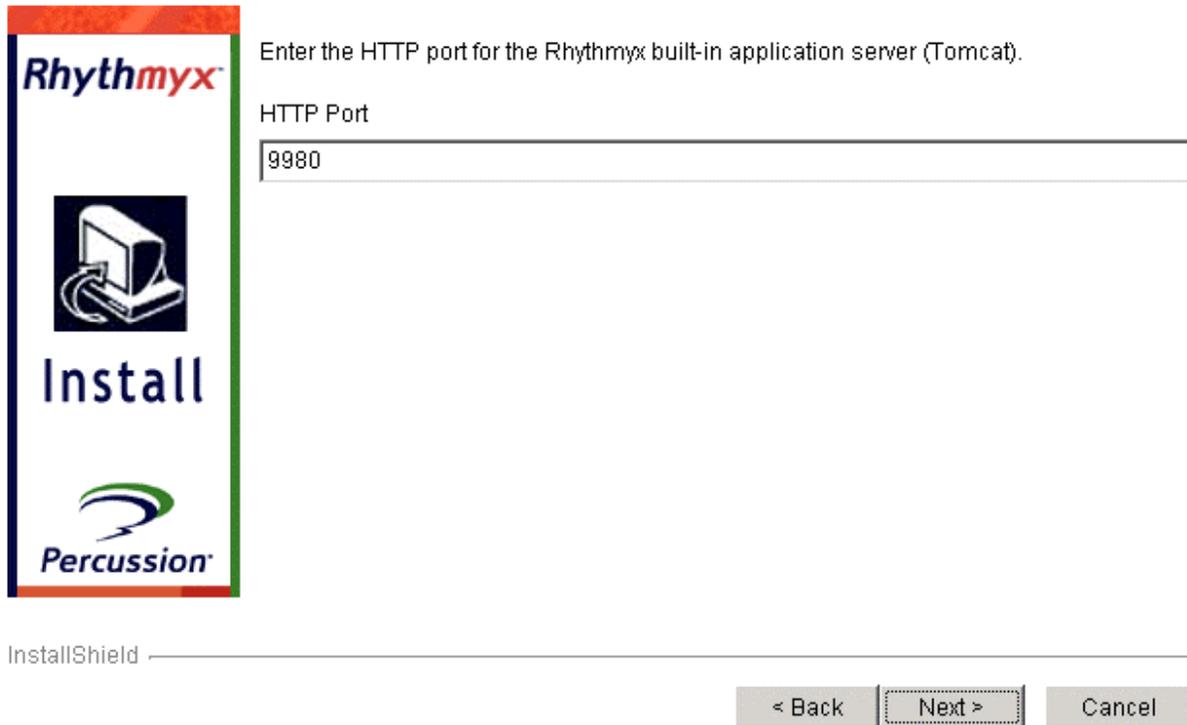


Figure 7: Rhythmyx Publisher

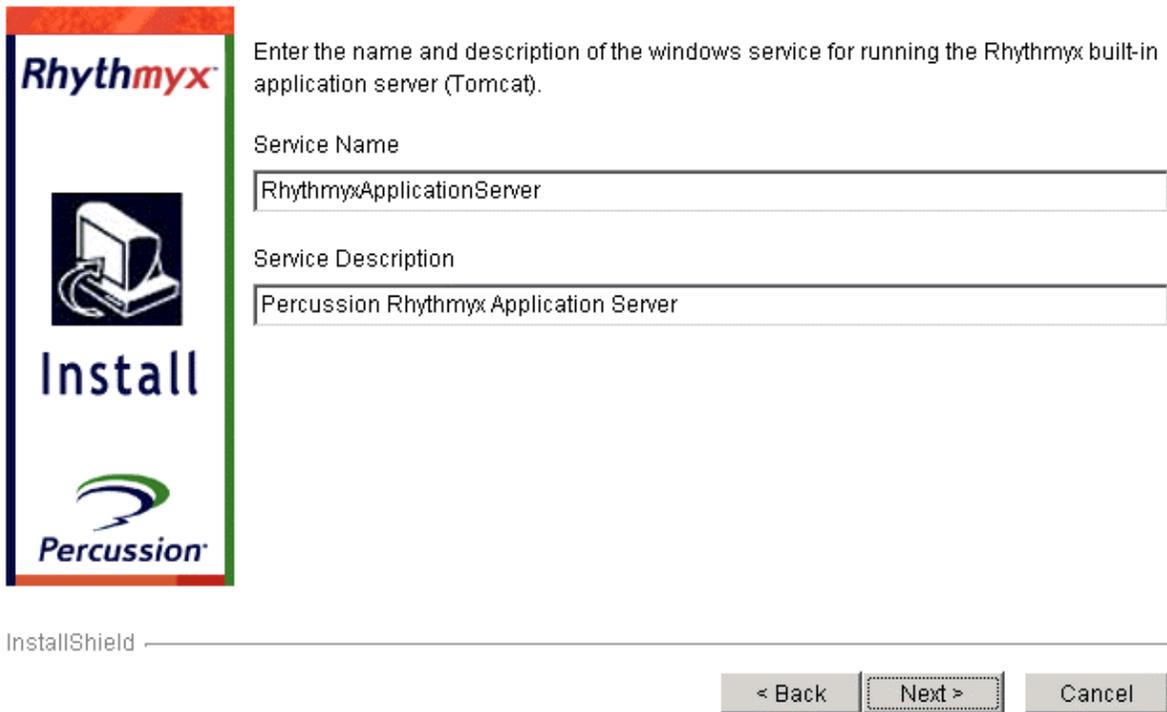
- 10 Define the port Tomcat will listen on for HTTP requests from the Rhythmyx Server during publishing. The Default is 9980. Press [Next].



The screenshot shows the Rhythmyx installation wizard interface. On the left is a vertical sidebar with the Rhythmyx logo at the top, a computer icon in the middle, the word "Install" below it, and the Percussion logo at the bottom. The main area contains the text "Enter the HTTP port for the Rhythmyx built-in application server (Tomcat)." followed by a label "HTTP Port" and a text input field containing the number "9980". At the bottom of the window, there are three buttons: "< Back", "Next >" (which is highlighted with a dotted border), and "Cancel". The text "InstallShield" is visible in the bottom left corner of the window frame.

Figure 8: Tomcat Listening Port

11 Name the Windows Service running Tomcat. Press [Next].



Enter the name and description of the windows service for running the Rhythmyx built-in application server (Tomcat).

Service Name
RhythmyxApplicationServer

Service Description
Percussion Rhythmyx Application Server

InstallShield

< Back Next > Cancel

Figure 9: Service Name

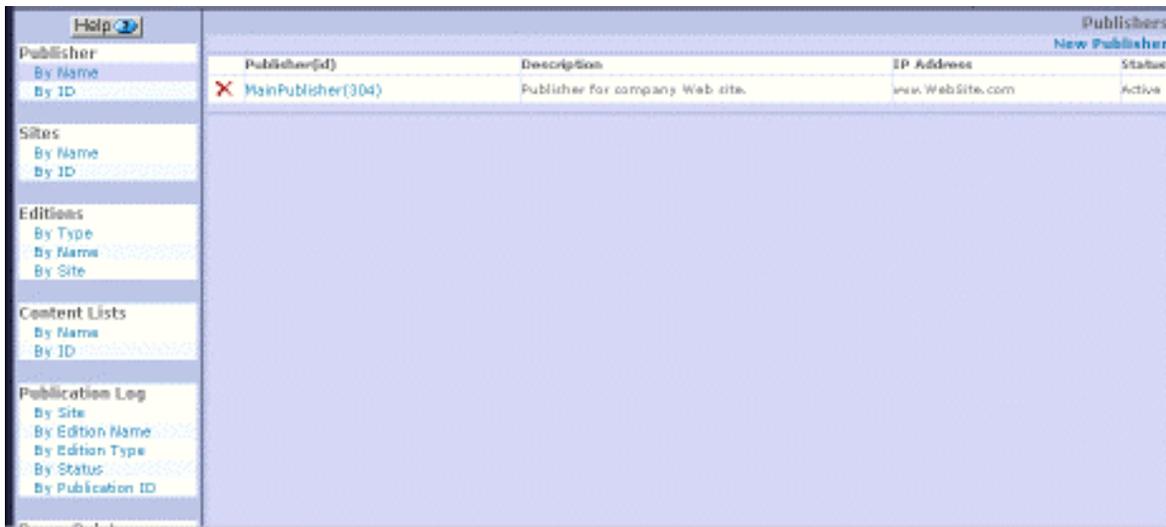
12 Continue the installation by pressing [next] at the pre-installation summary. Once the installation completes, close the installation dialog.

Note: The Service is automatically started after installation. Additionally, this Service is set to automatically start on server startup. This publisher is now ready to receive Publishing requests from a Rhythmyx Server. The registration of this "Publisher" is finalized in the Publisher submenu of the Publishing Web Browser tab.

Registering a Publisher

New/Edit Publisher Page

To view a Publisher's registration, go to the Publisher page. Under Publisher click the [By Name] or [By ID] sort links to display the currently registered Publishers on the system.

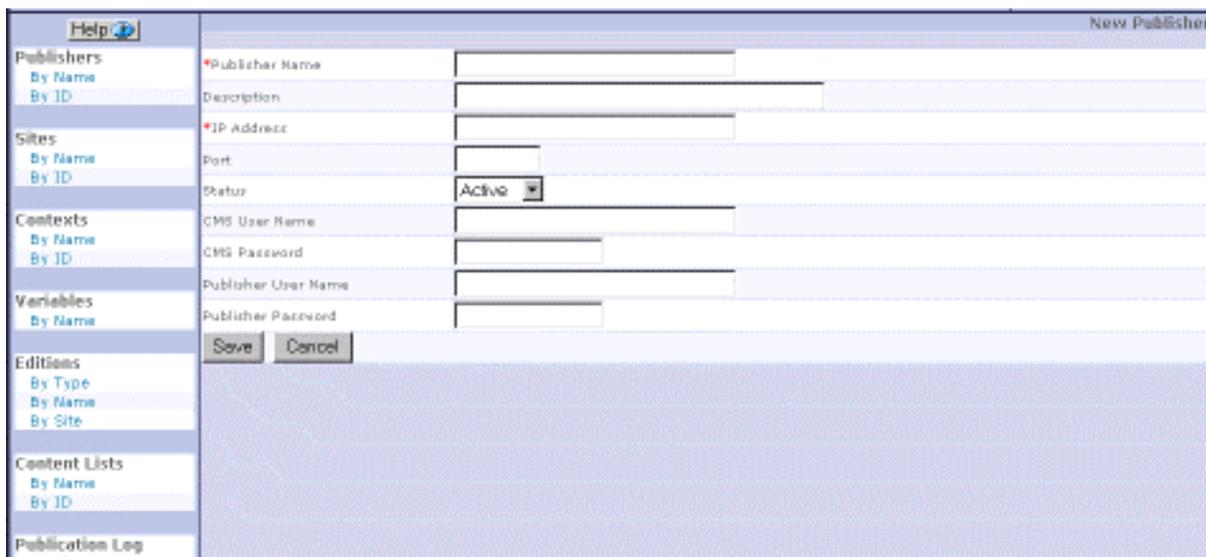


Publisher		Publisher(id)	Description	IP Address	Status
By Name					
By ID		X MainPublisher(304)	Publisher for company Web site.	www.WebSite.com	Active

The screenshot shows a web application interface with a left-hand navigation menu and a main content area. The navigation menu includes sections for Publisher, Sites, Editions, Content Lists, and Publication Log, each with sub-links for sorting by Name or ID. The main content area displays a table of registered publishers. The table has columns for Publisher ID, Description, IP Address, and Status. One publisher is listed: 'MainPublisher(304)' with description 'Publisher for company Web site.', IP 'www.WebSite.com', and status 'Active'. A 'New Publisher' link is visible in the top right corner of the table area.

Figure 10: Publisher Editor

Click [New Publisher] or the name of an existing Publisher to view the New/Edit Publisher page.



The screenshot shows the 'New Publisher' form. It includes a left-hand navigation menu and a main form area. The form fields are:

- Publisher Name (Mandatory, indicated by a red asterisk)
- Description
- IP Address (Mandatory, indicated by a red asterisk)
- Port
- Status (Dropdown menu, currently set to 'Active')
- CMS User Name
- CMS Password
- Publisher User Name
- Publisher Password

At the bottom of the form are 'Save' and 'Cancel' buttons.

Figure 11: Edit Publisher Page

Field Descriptions

Publisher Name - Unique name for Publisher. Mandatory.

Description - Description of Publisher. Optional.

IP Address - IP address of Publisher's location. If you use dynamic IP addresses, you can enter the machine name as long as it is resolvable (for example, using a DNS server or a host file). Mandatory.

Port - Port number the Publisher listens on for requests. This is defined during the installation of the actual Publisher itself.

Status - Status of the Publisher. Options are Active (default) and Inactive. A publisher can be made unavailable to Editions if necessary.

CMS User Name - Content Management System user name. The Publisher must authenticate when connecting to the Rhythmyx Server during publishing. This user must have the appropriate rights to access the applications responsible for the assembly of the item being published.

CMS Password - Corresponding password for the CMS User Name.

Publisher User Name - Optional. If entered, the Publisher Manager requires the user to log on when transmitting SOAP requests between the Publisher Manager and the Publisher. This user name and subsequent password will need to be provided in any script used to initiate an Edition.

Publisher Password - Optional. The login password for the user entered in Publisher User Name.

Registering a New Publisher

To register a Publisher:

- 1 On the Publishing Administrator, click [New Publisher].
- 2 Rhythmyx displays the New/Edit Publisher page.
- 3 Enter the Publisher Name, Publisher Description, IP Address, Port, Status, CMS User Name, CMS Password, Publisher User Name and Publisher Password.
- 4 Click [Save] to save the new record or [Cancel] to cancel the Publisher Registration.
- 5 *Configure the Publisher* (see "[Descriptions of Publisher Parameters](#)" on page 22).

Editing a Publisher's Registration

To edit a Publisher's Registration:

- 1 In the Publishing Administrator, click the name of the Publisher you want to modify.
- 2 Rhythmyx displays the Edit Publisher Page.
- 3 Modify the fields you want to change. The Publisher Name and IP Address are mandatory. Description, Port, Status, Publisher User Name, and Publisher Password are optional.
- 4 *Edit the Publishing Parameters* (see "[Editing a Configuration Parameter](#)" on page 24) that you want to change.
- 5 Click the [Save] button to save your changes or the [Cancel] button to cancel your changes.

Deleting a Publisher's Registration

To Delete a Publisher's Registration:

- 1 Go to the Publishing Administrator and display Publishers [By Name] or [By ID].

2 Click the delete button  in the row of the Publisher you want to delete. Rhythmyx deletes the Publisher.

Configuring a Publisher

By default, the lower portion of the Edit Publisher page displays four default Publisher configuration parameters: debug, filesystem, ftp, statusurl. You can add any of the optional parameters: sslport, soapurl, loglocation, jndiproviderurl, and jdbccontextfactory. You can also add custom configuration parameters.

Configuration Parameters			Add User Param
Name	Value	Description	
 database	com.percussion.publisher.client.PSDatabasePublisherHandler	database publisher handler	
 enablepassivemode	true	use FTP passive mode	
 ftprcvtimeout	60	seconds ftp publisher waits to time out	
 jdbccontextfactory	weblogic.jndi.WLInitialContextFactory	the JDBC context factory	
 jndiproviderurl	t3://devserver:7001	the JNDI provider URL	
 loglocation	webapps/publogs	location for storing log files	
 portal	com.percussion.portal.PSPortalPublisherPlugin	The plugin to publish to a portal site	
 serverrequesttimeout	120	seconds that ftp publisher waits for Content List before timeout	
 debug	true	test debug	
 filesystem	com.percussion.publisher.client.PSFilePublisherHandler	component used for filesystem publishing	
 ftp	com.percussion.publisher.client.PSFtpPublisherHandler	component used for FTP publishing	
 statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	Receives XML log from publisher and updates database	

Figure 12: Edit Publisher Page showing Publisher Parameters

Click [Add User Parameter] or the name of a parameter to go to the Edit Config Parameter page.

Edit Config Parameter	
*Name	Sample
Value	Sample
Description	Sample Parameter
Save	Cancel

Figure 13: Edit Config Parameter page

Edit Config Parameter Field Descriptions

- Name - Mandatory. The name of the Publisher configuration parameter.
- Value - Mandatory. The value of the configuration parameter. *Descriptions of Publisher Parameters* Rhythmyx will use this value in processing.
- Description - Optional. The description of the configuration parameter.

A list of the possible configuration values is available in the topic, *Descriptions of Publisher Parameters*.

Descriptions of Publisher Parameters

Publisher parameters are stored in the table RXPUBLISHERCONFIG.

Parameter	Description
debug	Defines whether the Publisher traces the operation of the publishing plug-ins and writes additional logging data for them. Default value is <code>true</code> . To turn off debugging, set this parameter to <code>false</code> .
filesystem	Defines the class name of the plugin used when publishing to a file system. The default value is <code>com.percussion.publisher.client.PSFilePublisherHandler</code> . Do not change the value of this field unless you have a new plugin to handle publishing to a file system. This field cannot be deleted.
ftp	Defines the class name of the plugin used when publishing through file transfer protocol. The default value is <code>com.percussion.client.PSFTPHandler</code> . Do not change the value of this field unless you have a new plugin to handle publishing through ftp. This field cannot be deleted.
statusurl	Defines the URL to which the Publisher will post updates in the Publish status of content items. The default value is <code>/Rhythmyx/sys_pubSupport/pubstatus.xml</code> .
sslport	Optional, user-added parameter. Indicates that Rhythmyx should send http requests over channels that support SSL using https, and defines the port that receives http requests. The value of the parameter should be the Rhythmyx server's SSL port (9443 by default). If this parameter is not present, Rhythmyx uses the original protocol and port supplied by the content list. To enable SSL support, see <i>Enabling SSL in the Publisher Server</i> (on page 29). <i>Note: You must enter the exact spelling "sslport" for Rhythmyx to recognize the parameter and use it correctly.</i>
soapurl	Optional, user-added parameter. Defines an alternative SOAP URL. If this parameter is not supplied, Rhythmyx uses the default SOAP URL <code>/soap/servlet/rpcrouter</code> . <i>Note: You must enter the exact spelling "soapurl" for Rhythmyx to recognize the parameter and use it correctly.</i>

Parameter	Description
loglocation	<p>Optional, user-added parameter. Defines the address of the publishing log. Recognizes forward and backward slashes, and accepts absolute or relative paths. If the path starts with a slash or driver letter (/webapps/publogs or c:\webapps\publogs), loglocation interprets it as an absolute path. If the path starts with ../ (../webapps/publogs), loglocation interprets it as relative from the publisher installation root. If this parameter is not supplied, Rhythmyx uses the default log location (../webapps/publogs).</p> <p>In order to access the log files through the Content Explorer, you must specify the appropriate context (virtual directory) for your Web server. For example, if you set loglocation to cmslogs, to make the log files available when you are using Tomcat as your Publisher server, you would change server.xml (the server configuration file) by adding a new context:</p> <pre><Context path="/publogs" docBase="cmslogs" debug="0" privileged="true"/></pre> <p>The <Context> element must be a child element of the <Host> element.</p> <p><i>NOTE: the path must always be /publogs, while docBase is the directory relative to the Web server (in this case the publisher) directory.</i></p> <p><i>NOTE: You must enter the exact spelling "loglocation" for Rhythmyx to recognize the parameter and use it correctly.</i></p>
jndiproviderurl	<p>Optional, user-added parameter. Specifies the JNDI provider URL. If this parameter is not supplied, Rhythmyx uses the default for Tomcat.</p> <p><i>Note: You must enter the exact spelling "jndiproviderurl" for Rhythmyx to recognize the parameter and use it correctly.</i></p>
jdbccontextfactory	<p>Optional, user-added parameter. Specifies the JDBC context factory. If this parameter is not supplied, Rhythmyx uses the default for Tomcat.</p> <p><i>Note: You must enter the exact spelling "jdbccontextfactory" for Rhythmyx to recognize the parameter and use it correctly.</i></p>
enablepassivemode	<p>Optional, user-added parameter. Specifies whether to use FTP active or passive mode. The mode may depend on the settings of the firewall that is local to the Publisher. Values are true or false:</p> <ul style="list-style-type: none"> true – Use FTP passive mode. false – Use FTP active mode. <p>If this parameter is not supplied, Rhythmyx uses FTP active mode.</p> <p>Configuring Firewalls to Facilitate Publishing (on page 27)</p> <p><i>Note: You must enter the exact spelling "enablepassivemode" for Rhythmyx to recognize the parameter and use it correctly.</i></p>

Parameter	Description
serverrequesttimeout	Optional, user-added parameter. Specifies how many seconds the Rhythmyx Publisher waits for a response to a request for a content list from the Rhythmyx server before timing out. If the user chooses to try again, Rhythmyx uses the same timeout value. Timeout occurs even if the content list is transferring, but has not arrived. If this parameter is not supplied, the Publisher waits 120 seconds. <i>Note: You must enter the exact spelling "serverrequesttimeout" for Rhythmyx to recognize the parameter and use it correctly.</i>
ftprcvtimeout	Optional, user-added parameter. Specifies number of seconds that the FTP Publisher waits when expecting data from the FTP server. If this parameter is not supplied, the Publisher waits 60 seconds for data to arrive. <i>Note: You must enter the exact spelling "ftprcvtimeout" for Rhythmyx to recognize the parameter and use it correctly.</i>
Custom Properties	Lets you define custom properties to use with custom Publisher plugins that require parameters in addition to those included.

Adding a Publisher Parameter

To add a new Config Parameter to a Publisher:

- 1 Go to the Edit Config Parameter page.

Figure 14: Edit Config Parameter page

- 2 Enter the Name of the new parameter, its Value, and its optional Description.
- 3 Click [Save] to save the new configuration parameter record or [Cancel] to cancel your edits.

Note: The topic, *Descriptions of Publisher Parameters* provides a list of available values.

Editing a Configuration Parameter

To edit a Publisher parameter:

- 1 On the Edit Publisher page, click the name of the parameter you want to edit.
Rhythmyx displays the Edit Configuration Parameter page.
- 2 You can change Value and Description.
- 3 Click [Save] to save your changes.

Note: The topic, Descriptions of Publisher Parameters provides a list of available values.

Deleting a Publisher Parameter

NOTE: When you delete a parameter, the record is deleted. Any extensions that use that parameter will generate errors.

To delete a Publisher parameter:

- 1 On the lower portion of the Edit Publisher page, click the delete button  of the parameter you want to delete.
- 2 Rhythmyx displays a warning message. Click [OK] to confirm the delete action. Click [Cancel] to abort the delete action.

Publisher Logging

If you use a server other than the Rhythmyx Tomcat server, you must define the `loglocation` configuration parameter with your Publisher to specify the location of the Publisher log.

Specify a relative or absolute location. Relative path information must be relative from the `<tomcat>` root directory.

Using a Custom SOAP URL

If you do not want to use the default SOAP URL, use the configuration parameter `soapurl` to customize it. If you do not specify it, the Publisher uses the default: `/soap/servlet/rpcrouter`.

Adding Publisher Authentication

Even if you have enabled SSL support, it is possible for a knowledgeable user to publish content to your Web site using the Publisher. To prevent this from happening, add authentication to transmissions between the Publisher Manager and the Publisher.

To add authentication between the Publisher Manager and the Publisher:

- 1 Enter a Publisher User Name and a Publisher Password in the New/Edit Publisher page.
- 2 Add Publisher authentication to your Publisher server.

Adding Publisher Authentication to the Publisher Server

The Publisher server handles authentication of transmissions between the Publisher and the Publisher Manager. If you are using the Tomcat server, follow the steps below to add authentication. If you are using another server for publishing, use the procedure required by the server's platform to add authentication.

To add authentication to the Tomcat server:

- 1 Add a security constraint to the **SOAP** application, `<AppServer Root>/AppServer/webapps/RxServices/WEB-INF`, by adding the following code to `web.xml` root element:

```
<!-- Define a Security Constraint on this Application -->
<security-constraint>
```

```
<web-resource-collection>
  <web-resource-name>Entire Application</web-resource-name>
  <url-pattern>/*</url-pattern>
</web-resource-collection>
<auth-constraint>
  <role-name>rxpublisher</role-name>
</auth-constraint>
</security-constraint>
<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Rhythmyx Publisher Application</realm-name>
</login-config>
```

This code requires users of the SOAP application to log in as a specified role. In this example, the specified role is "rxpublisher."

- 2 Add the following `<Context>` element for the SOAP application within the `<Host>` element in `<AppServer Root>/AppServer/conf/server.xml`:

```
<Context path="/soap" docBase="soap" debug="0" privileged="true">
  <Realm className="org.apache.catalina.realm.MemoryRealm"
  digest="SHA"/>
</Context>
```

In this example, the `<Context>` element causes the SOAP application to use `MemoryRealm` and digest passwords with the SHA algorithm, but you can use any `Realm` implementation and digesting algorithm. See the Tomcat documentation for more information <http://host:port/tomcat-docs/realm-howto.html>.

- 3 Use the default file, `<AppServer Root>/AppServer/conf/tomcat-users.xml`, to get the user names and passwords. Encrypt all publisher users passwords with the same algorithm that you specified in the previous step:

```
<user name="rhythmyx"
password="b497a0aad7d4c7179b4fa30ccb0b930e674048dd"
roles="rxpublisher" />
```

Tomcat provides a tool to encrypt your passwords. See <http://host:port/tomcat-docs/realm-howto.html#Digested%20Passwords>.

Using SSH Tunneling with FTP Publishing

To use SSH tunneling with passive FTP Publishing:

- 1 Configure an SSH tunnel. For instructions, see SSH Tunneling at www.ssh.com (<http://www.ssh.com>).
- 2 Configure your Publisher to use passive mode by setting the *enablepassivemode* parameter to true.
- 3 Confirm that your FTP server accepts passive FTP connections.
- 4 **Confirm that your firewall is configured to allow passive FTP between the Rhythmyx Publisher and the FTP server** (see "[Configuring Firewalls to Facilitate Publishing](#)" on page 27).

Configuring Firewalls to Facilitate Publishing

To configure the firewalls in your network to allow Rhythmyx components to communicate during Publishing, open communication between ports on the FTP client (in this case the Rhythmyx Publishing hub) and the FTP server.

Firewall configuration differs for active FTP and passive FTP. Note that you may need to use Passive FTP if:

- Your local firewall does not allow connections to be made from remote servers into local ports 1024 - 65535 (this is common)
- The remote firewall does not allow outgoing connections to be made from port 20 to remote ports 1024 - 65535 (this is less common)

Note that many firewalls have specific configuration settings to allow FTP connections. (For example, some firewalls will detect which local ports are being established for Active FTP sessions.) The description below may be used if finer control over ports is required.

Firewall Configuration for Active FTP

In the FTP Server's Firewall

FTP Control: Allow incoming connections to local FTP control port (usually 21) from ports in the range 1024 - 65535 on the Publishing Hub(s). Incoming connections may be disallowed on other ports.

FTP Data: Allow outgoing connections from local port 20 (FTP Data Port) to ports in the range 1024 - 65535 on the Publishing Hub(s). Optionally disallow incoming connections

In the Publishing Hub's Firewall

FTP Control: Allow outgoing connections from local ports 1024 - 65535 to the remote FTP control port (usually 21) on the FTP server(s). Optionally disallow incoming connections.

FTP Data: Allow incoming connections from remote port 20 (FTP Data Port) on the FTP server(s) to local ports in the range 1024 - 65535. Incoming connections may be disallowed on other ports.

Firewall Configuration for Passive FTP

In the FTP Server's Firewall

FTP Control: Allow incoming connections to local FTP control port (usually 21) from ports in the range 1024 - 65535 on the Publishing Hub(s). Incoming connections may be disallowed on other ports.

FTP Data: Allow incoming connections to the local range 1024 - 65535 from remote ports in the range 1024 - 65535 on the Publishing Hub(s). Incoming connections may be disallowed on other ports.

In the Publishing Hub's Firewall

FTP Control: Allow outgoing connections from local ports (1024 - 65535) to the remote FTP control port (usually 21) on the FTP server(s). Optionally disallow incoming connections.

FTP Data: Allow outgoing connections from local ports in the range 1024 - 65535 to the remote ports in the range 1024 - 65535 on the FTP server(s). Optionally disallow incoming connections.

Setting Up Scheduled Publishing

The Rhythmyx Publishing Launcher is a mechanism for initiating automatic Publishing runs from a scheduler. A Java program that requires an associated batch file or shell script, the Publishing Launcher acts as an alternative "run button" that can be invoked by a scheduling application, such as NT Scheduler or Cron in Unix.

When you install Rhythmyx, a batch file (ScheduledPublication.bat) or shell script (ScheduledPublication.sh) is created in the <Rhythmyxroot>/AppServer/bin directory. This script calls the Publisher Launcher Java program and passes the parameters for the job:

Parameter Order	Parameter Name	Description	Default Value
1	server	Name or IP address of Rhythmyx server. (required) Included in sample batch file by default.	localhost
2	port	Rhythmyx server listening port. (required) Included in sample batch file by default.	Port specified for the Web application server when installing Rhythmyx
3	editionid	ID of the Edition to Publish. (Required) Included in sample batch file by default.	301
4	cmsuserid	CMS user ID (optional). Not included in sample batch file by default	<null>
5	password	Password for user specified in the cmsuserid parameter. (optional). Not included in sample batch file by default.	<null>
6	useSSL	Specifies whether to use SSL when communicating with the Rhythmyx server during Publishing. (optional) Options are "yes" and "no". If no value is provided for this parameter, the default value of "no" is assumed. Not included in the sample batch file by default.	<null>

The parameters are separated by spaces. For example, the batch file in Windows resembles the following (without the line breaks):

```
cmd /K ..\JRE\bin\java -cp ../webapps/RxServices/WEB-INF/lib/
rxpublisher.jar;../webapps/RxServices/WEB-INF/lib/rxmisctools.jar;
com.percussion.publisher.runner.PSRemotePublisher localhost 9880 301
```

To set up scheduled publishing of an Edition, create a copy of the file with a new name. Update the parameters to match . For example, to publish an Edition with the ID 472, on a Rhythmyx server residing on a machine with the IP address 255.255.255.56 and listening on port 9550, you would modify the code above as follows:

```
cmd /K ..\JRE\bin\java -cp ../webapps/RxServices/WEB-INF/lib/  
rxpublisher.jar;../webapps/RxServices/WEB-INF/lib/rxmisctools.jar;  
com.percussion.publisher.runner.PSRemotePublisher 255.255.255.56 9550  
472
```

Once you have modified the script to match your requirements, use Windows Scheduler or cron (in Unix) to set up a schedule to run the script.

Publishing with SSL

SSL (Secure Sockets Layer) is a protocol that ensures the authenticity of information exchanged by servers. It uses a digital certificate, which identifies the sender, and public and private keys for signing messages and encrypting/decrypting data. SSL-enabling your Rhythmyx Server ensures secure communications with the Rhythmyx Publisher and other servers with which it communicates. The Rhythmyx Server and Rhythmyx Publisher must both be SSL-enabled or -disabled to be able to communicate. You SSL-enable Rhythmyx Publishing by creating a private/public key pair and a digital certificate for the Publisher server as well as configuring the Publisher to use SSL. You must also configure the Rhythmyx Server to use SSL if you have not already SSL-enabled it.

Enabling SSL in the Publisher Server

To enable SSL in the Publisher Server:

- 1 **Create an SSL Certificate** (see "[Creating SSL Certificates](#)" on page 31) for the Rhythmyx Publisher. If your Rhythmyx Publisher and Rhythmyx Server reside on the same machine, you can use the same certificate for both of them.
- 2 In the Publisher registration
 - a) Add the optional *sslport parameter*. Set it equal to the default value of 9943 or the value you assign to the SSL port in the Publisher server.xml file (see the following step). Note that the Rhythmyx Publisher SSL Port is 9943 by default, and the Rhythmyx Server SSL Port is 9443 by default.

- b) Set the IP Address to the DNS name for which you created the SSL certificate for the Publisher. SSL cannot resolve simple host names.

Publisher(id): Sample Publisher(1)		Edit Publisher
*Publisher Name	<input type="text" value="Sample Publisher"/>	
Description	<input type="text" value="Sample Publisher"/>	
*IP Address	<input type="text" value="127.0.0.1"/>	
Port	<input type="text" value="9980"/>	
Status	<input type="text" value="Active"/>	
CMS User Name	<input type="text" value="admin1"/>	
CMS Password	<input type="password" value="****"/>	
Publisher User Name	<input type="text"/>	
Publisher Password	<input type="password"/>	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>		
Configuration Parameters		Add User Param
Name	Value	Description
<input checked="" type="checkbox"/> sslport	9943	Port for SSL Publishing
<input checked="" type="checkbox"/> debug	true	test debug
<input checked="" type="checkbox"/> filesystem	com.percussion.publisher.client.PSFilePublisherHandler	test filesystem
<input checked="" type="checkbox"/> ftp	com.percussion.publisher.client.PSFtpPublisherHandler	test ftp
<input checked="" type="checkbox"/> statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	test statusurl

- 3 In the file `<Rhythmyx root>/AppServer/conf/server.xml`,

uncomment the code:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 9943 -->
<!--
  <Connector
    className="org.apache.coyote.tomcat4.CoyoteConnector"
    port="9943" minProcessors="5" maxProcessors="75"
    enableLookups="true" acceptCount="100" debug="0"
    scheme="https"
    secure="true" useURIVValidationHack="false"
    disableUploadTimeout="true">
    <Factory
      className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
      clientAuth="false" protocol="TLS" keystorePass="cle231"
    />
  </Connector>
```

and include in the `<Factory>` element the parameter:

- `keystoreFile` - Specifies the keystore file. Must be a fully qualified path.

For example:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 9943 -->
<Connector
  className="org.apache.coyote.tomcat4.CoyoteConnector"
  port="9943" minProcessors="5" maxProcessors="75"
  enableLookups="true" acceptCount="100" debug="0"
  scheme="https" secure="true" useURIVValidationHack="false"
  disableUploadTimeout="true">
```

```

    <Factory
    className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
    clientAuth="false" protocol="TLS"
    keystoreFile="d:\rx\installs\keystores\sslTest.jks"
    keystorePass="changeit"/>

```

(If you are using another J2EE server as your Publisher server, follow the instructions for that server on how to enable SSL support.)

- 4 In your Content List application the Content Item URL should be mapped to *sys_MakeAbsLinkSecureEx* (on page 36), an extension that creates the correct URL. In this extension, set the value of the parameter `host` to the DNS name for which you created the SSL certificate for the Publisher; if you do not set `host`, the extension uses the value of the `host` from the original request, which SSL cannot resolve because it is not in DNS format.
- 5 If you have not already SSL-enabled the Rhythmyx Server, *configure it to use SSL* (see "Setting Up SSL in the Rhythmyx Server" on page 34).

Creating SSL Certificates

Implementing SSL requires a certificate. You can either obtain a certificate from a recognized Certificate Authority (CA) or create a self-signed certificate. Regardless of who signs the certificate, SSL will work and will provide your users with the same level of data-transmission security. If you use a certificate signed by a well-known or trusted CA, your users' web browser software can easily verify that it is communicating with the specific server for which the certificate was issued. The browser will also indicate that the certificate was signed by a trusted CA. The drawback to using a certificate signed by a well-known CA is that the CAs tend to charge a high price to sign your certificate.

NOTE: If your Rhythmyx Server and Publisher run on the same host, you can use the same SSL Certificate for both of them. If they run on separate hosts, each host has a different DNS name, and therefore, each must have its own SSL certificate.

Creating Self-Signed SSL Certificates

To set up self-certification, use the `keytool` included with the Rhythmyx JRE to create an SSL certificate, and sign it yourself (hence, self-certification).

To set up self-certification:

- 1 Create a certificate file (.csr file) to store your certificate. To create the .csr file, start a simple text editor, such as Notepad or vi, and save an empty file. Use the name of your Rhythmyx system alias. For example, if you name your Rhythmyx server "Rhythmyx", you would save your file with the name `Rhythmyx.csr`. Save this file to the directory `<Rhythmyxroot>/JRE/bin`.
- 2 Start a Command Window or console. Change directories to `<Rhythmyxroot>/JRE/bin`.
- 3 Use the `keytool` to create a public/private key pair:

```

keytool -genkey -alias <keystoreentryalias> -keyalg RSA -dname
"CN=<rhythmyxhostname>,OU=<organizationalunit>,O=<organization>,L=
<location>,S=<state>,C=<country>" -keypass <keypassword> -
storepass <storepassword> -keystore <keystorefilename>

```

Where:

- `<keystoreentryalias>` is the keystore entry alias;

- `<rhythmyxhostname>` is the name of the machine where the Rhythmyx server resides;
- `<organizationalunit>` is the name of the organization unit for which you are issuing the certificate (typically the department responsible for operating the Rhythmyx server);
- `<organization>` is the organization for which you are issuing the certificate (typically your company name);
- `<location>` is the name of your location, typically the town or city
- `<state>` is the name of the state or province;
- `<country>` is the name of your country;
- `<keypassword>` is the password for the key pair;
- `<storepassword>` is the password for the keystore file;
- `<keystorefilename>` is the name of the keystore file; you can give this file any name.

For example:

```
keytool -genkey -alias Rhythmyx -keyalg RSA -dname
"CN=rhythmyxhostname,OU=Development,O=Percussion,L=Stoneham,S=Mass
achusetts,C=US" -keypass mypass -storepass mypass -keystore
Rhythmyx.keystore
```

4 Self-certify the key pair:

```
keytool -selfcert -alias <keystoreentryalias> -keypass
<keypassword> -storepass <storepassword> -keystore
<keystorefilename>
```

Where:

- `<keystoreentryalias>` is the keystore entry alias;
- `<keypassword>` is the password for the key pair;
- `<storepassword>` is the password for the keystore file;
- `<keystorefilename>` is the name of the keystore file.

For example:

```
keytool -selfcert -alias Rhythmyx -keypass mypass -storepass
mypass -keystore Rhythmyx.keystore
```

5 Export the certificate file from the newly created key pair to the certification file (.csr):

```
keytool -export -alias <keystoreentryalias> -keypass <keypassword>
-storepass <storepassword> -keystore <keystorefilename> -file
<certificatefile>
```

Where:

- `<keystoreentryalias>` is the keystore entry alias;
- `<keypassword>` is the password for the key pair;
- `<storepassword>` is the password for the keystore file;
- `<keystorefilename>` is the name of the keystore file.

- <certificatefile> is the name of the certificate file; you can give this file any name, but to make it easy to recognize, it is strongly recommended that you use “.csr” as the extension.

For example:

```
keytool -export -alias Rhythmyx -keypass mypass -storepass
mypass -keystore Rhythmyx.keystore -file Rhythmyx.csr
```

6 Import the certificate file into the cacerts files of the JRE of both the Rhythmyx server and Application Server:

```
keytool -import -noprompt -trustcacerts -alias
<keystoreentryalias> -storepass changeit -file <certificatefile> -
keystore <cacertpath>
```

Where

- <keystoreentryalias> is the keystore entry alias;
- <certificatefile> is the name of the certificate file. The file name must end with the extension “.csr”;
- <cacertpath> is the path to the cacert file into which you want to import the certificate; use the following paths:

To import the certificate to the...	use the path:
server	<Rhythmyxroot>\JRE\lib\security\cacerts
Application Server (Rhythmyx 5)	<Rhythmyxroot>\AppServer\JRE\lib\security\cacerts

Note that the value of the storepass parameter must be *changeit*, which is the default password for the certificate files.

For example, to update the Publisher certification, you would use the command:

```
keytool -import -noprompt -trustcacerts -alias Rhythmyx -storepass
changeit -file Rhythmyx.csr -keystore
C:\Rhythmyx\Publisher\JRE\lib\security\cacerts
```

- 7** Move the keystore (Rhythmyx.keystore) file to the Rhythmyx root directory.
- 8** On the Rhythmyx Server Properties dialog, configure the Rhythmyx server to use the keystore file. For details about starting the Rhythmyx Server Properties dialog, see “Maintaining Server Properties” in the Rhythmyx Server Administrator Help. For details about server SSL parameters, see “SSL Server Parameters” in the Rhythmyx Server Administrator Help.
- 9** Restart the Rhythmyx server.
- 10** Repeat the process, creating the certificate on the Application Server and importing it into the Rhythmyx server.

NOTE: On Unix, the order of switches and parameters may be important. Best practice when implementing SSL on Unix is to enter the switches in the order specified.

NOTE: In order for Content Types based on Microsoft Word to work correctly with a self-certified certificate, you must set up your browser to trust the certificate. Follow the instructions for modifying browser security settings in the topic “Prerequisites for using the Rhythmyx Connector for Word” in the Rhythmyx Content Explorer Help.

Certificate Authority

A certificate authority is an organization that issues digital certificates to users or groups of users. The digital certificates confirm the identity of the user or group.

Setting Up SSL Certificates Issued by Certificate Authorities

For an example of how to set up an SSL Certificate issued by a Certificate Authority, see the topic “Setting Up SSL Certificates” in the Discussion Forum on the Rhythmyx Support Extranet (<http://www.percussion.com/support/rhythmyx-extranet/>).

Setting Up SSL in the Rhythmyx Server

If your Rhythmyx Server and Publisher run on the same host, you can use the same SSL Certificate for both of them. If they run on separate hosts, you must create separate keystores and certificates for each of them. To create an SSL certificate for the Rhythmyx Server, follow the instructions in *Creating Self-signed SSL Certificates* (on page 31) or *Setting Up SSL Certificates Issued by Certificate Authorities* (on page 34).

SSL support in the Rhythmyx Server requires configuration of several parameters in the `../rxconfig/Server/server.properties` file. When the `SSLKeyStoreFile` parameter has a value, SSL support is enabled; when the parameter does not have a value, SSL support is disabled. By default, the `SSLKeyStoreFile` parameter and the other SSL parameters are not defined and SSL support is disabled.

Change the SSL parameters through the Rhythmyx Server Properties dialog.

Parameter	Description	Default
SSLKeyStoreFile	Name and location of the <i>keystore</i> (on page 37) file that contains the server's public and private key and <i>digital certificates</i> (see "Digital Certificate" on page 37). The path indicated may be relative from the Rhythmyx installation directory.	None
SSLKeyStore Password	The encrypted password for validating the key store specified in <code>SSLKeyStoreFile</code> . <i>Required when <code>SSLKeyStoreFile</code> is defined.</i>	None

Parameter	Description	Default
SSLCipher	<p>A list of allowed cipher suites (encryption protocols) delimited with semi-colons. The Rhythmyx server ignores white space around the delimiters. Supported cipher suites are:</p> <p>SSL_DHE_DSS_WITH_DES_CBC_SHA</p> <p>SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA</p> <p>SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA</p> <p>SSL_RSA_WITH_RC4_128_MD5SSL_RSA_WITH_RC4_128_SHA</p> <p>SSL_RSA_WITH_DES_CBC_SHA</p> <p>SSL_RSA_WITH_3DES_EDE_CBC_SHA</p> <p>SSL_RSA_EXPORT_WITH_RC4_40_MD5</p> <p>For descriptions of these cipher suites, see http://www.netscape.com/eng/ssl3/draft302.txt (http://www.netscape.com/eng/ssl3/draft302.txt)</p> <p><i>Not required when SSLKeyStoreFile is defined.</i></p>	<p>SSL_RSA_WITH_RC4_128_MD5;</p> <p>SSL_RSA_WITH_RC4_128_SHA;</p> <p>SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA</p>
SSLPort	<p>Port that receives requests that require an SSL connection.</p> <p><i>Not required when SSLKeyStoreFile is defined.</i></p>	9443

To test whether you have implemented SSL correctly in the Rhythmyx Server, access Rhythmyx using HTTPS:

`https://<rhythmyxhostname>:9443/Rhythmyx`

Where <Rhythmyxhostname> is the name of the machine where your Rhythmyx server resides and 9443 is the SSL port you specified when you configured your server to use SSL. If your browser presents you with the login dialog, SSL is working correctly.

sys_MakeAbsLinkSecureEx

Context: Java/global/percussion/generic

Description:

This exit creates an absolute URL with up to 10 name/value pairs. It is identical to `sys_MakeAbsLinkSecure` except that it allows you to specify a host name and port. Typically, the host specified is the name of the secure DNS server.

This UDF is preferred to the `sys_MakeAbsLinkSecure` UDF when using SSL for publishing.

Class name: `com.percussion.extensions.general.PSMakeAbsLinkSecureEx`

Interface: `com.percussion.extension.IPSUdfProcessor`

Parameters

Name	Data Type	Description
useSecure	java.lang.String	Optional. Flag specifying whether or not to use a secure connection. Enter <code>yes</code> to specify use of https for a secure connection; enter <code>no</code> (or any value other than <code>yes</code>) to specify use of http for a non-secure connection. Defaults to <code>yes</code> .
host	java.lang.String	(Optional) The host name to be used to produce the output url. If not specified, the host name of the originating request will be used.
port	java.lang.String	(Optional) The port to be used to produce the output url. If not specified, the port of the originating request will be used, subject to the value of the <code>\\"useSecure\\"</code> .
resource	java.lang.String	Relative resource without the parameters.
paramName1	java.lang.String	(Optional) Name of the first HTML parameter.
paramValue1	java.lang.String	(Optional) Value of the first HTML parameter.
paramName2	java.lang.String	(Optional) Name of the second HTML parameter.
paramValue2	java.lang.String	(Optional) Value of the second HTML parameter.
paramName3	java.lang.String	(Optional) Name of the third HTML parameter.
paramValue3	java.lang.String	(Optional) Value of the third HTML parameter.
paramName4	java.lang.String	(Optional) Name of the fourth HTML parameter.
paramValue4	java.lang.String	(Optional) Value of the fourth HTML parameter.
paramName5	java.lang.String	(Optional) Name of the fifth HTML parameter.
paramValue5	java.lang.String	(Optional) Value of the fifth HTML parameter.
paramName6	java.lang.String	(Optional) Name of the sixth HTML parameter.
paramValue6	java.lang.String	(Optional) Value of the sixth HTML parameter.
paramName7	java.lang.String	(Optional) Name of the seventh HTML parameter.

Name	Data Type	Description
paramValue7	java.lang.String	(Optional) Value of the seventh HTML parameter.
paramName8	java.lang.String	(Optional) Name of the eighth HTML parameter.
paramValue8	java.lang.String	(Optional) Value of the eighth HTML parameter.
paramName9	java.lang.String	(Optional) Name of the ninth HTML parameter.
paramValue9	java.lang.String	(Optional) Value of the ninth HTML parameter.
paramName10	java.lang.String	(Optional) Name of the tenth HTML parameter.
paramValue10	java.lang.String	(Optional) Value of the tenth HTML parameter.

Debugging Your SSL Publishing Implementation

If your SSL publishing configuration is not working, determine whether the Publisher is retrieving content from the correct URL. The publisher log should specify a URL that uses “https” instead of “http”, a DNS name instead of a simple host name, and the correct SSL port for the Rhythmyx Server (9443 by default). For example:

```
Rhythmyx Publisher 5.0; Build:20030801 Tue Sep 16 13:11:53 EDT 2003 -
begin log
```

```
[Debug] Processing content item # 1(content ID: 1014) ...
```

```
[Debug] PSContentItem - Getting content item from the URL :
```

```
https://abc.xyz.gov.uk:9443/Rhythmyx/EI_casArticle/article.html?sys_sit
eid=1&sys_contentid=1014&sys_variantid=329&sys_context=1&sys_revision=6
&sys_authtype=0&currentpage=1
```

For additional information about debugging SSL, see the following URL:

<http://java.sun.com/products/jsse/install.html> (<http://java.sun.com/products/jsse/install.html>).

SSL Glossary

SSL

Secure Sockets Layer (SSL) is a protocol that lies beneath another protocol (such as HTTP) to provide secure transfer of information on the Internet.

Keystore

A keystore is a file that holds public and private keys and digital certificates.

Digital Certificate

A digital certificate is a file of credentials that a user can submit for validation and identification purposes when performing transactions on the Web. A digital certificate can be issued by a Certificate Authority, or it can be self-signed. It contains identifying information and a public key, which is used to encrypt messages.

CHAPTER 2

Sites

A Site defines a location where output will be published. A Site may be a directory location or it may point to an ftp location. Rhythmyx can maintain multiple Sites on the same machine.

Usually, different Sites refer to different servers. Multiple Editions may publish to a single Site. However, if you want to separately track publishing activity of Editions that are published on the same server, configure different Sites.

To access the Sites Administrator:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
- 2 On Publishing tab, click By Name or By ID under *Sites*.

Rhythmyx displays the Site Editor.

				Sites	
				New Site	Copy Site
Site(id)	Description	File Tree	Last Publication Date		
✘ Corporate Investments(303)	Represents the Corporate Investments web site		2005-03-31 21:12:51.0		
✘ Corporate Investments - Mirror (304)	Represents a mirror site for Corporate Investments web site				
✘ Enterprise Investments(301)	Represents the Enterprise Investments web site		2005-04-01 09:36:18.0		
✘ Enterprise Investments-Mirror (302)	Represents a mirror site for Enterprise Investments web site				

You can sort the Sites listed in the Site Editor by the name of the Site (default) or by the ID of the Site. To change the sort order, click on the link for the order you want.

Site Maintenance Dialogs

Rhythmyx provides two pages you can use to maintain Site data.

- Edit Slot Properties Page
- *Copy Site Page* (on page 42)

Edit Site Properties Page

Use the Edit Site page to *create* (see "Deleting a Site" on page 44) and *maintain* (see "Editing a Site" on page 44) Site records.

To access the Edit Site page, access the Sites page and:

- click [New Site](#) to create a new Site; or
- click the name of the Site whose record you want to edit.

Edit Site Properties	
Site(id): (301)	
*Site Name	Internet
Description	Represents the Internet web site
Site Address (URL)	http://127.0.0.1
Home Page (URL)	
Publishing Root Location	c:\inetpub\wwwroot
*Publisher	Localhost Publisher Default Port
Status	Active
Folder Root	//Sites/Internet
Global Template	
Nav Theme	
FTP Information:	
IP Address	127.0.0.1
Port Number	21
User ID	
Password	
Save	Cancel

Figure 15: Edit Site Properties Page

Edit Site Properties Field Descriptions

Site Name - Mandatory. The name of the Site.

Description - Optional. Description of the Site.

Site Address (URL) - Optional. The URL entered to browse the page from a web server to an http server. This can be a fully qualified path if the output target is *filesystem* or a virtual directory if the output target is *ftp*

Home Page (URL) - Optional. The Site Explorer home page for the site. The URL must include contentid and variantid parameters for the sys_casGeneratePubLocation exit to use when it generates URLs for pages in the site. Only sites with Home Page URLs appear in Sites View in the Content Explorer and are available in Site Explorer.

Example Home Page URL:

```
http://10.10.10.10:9992/Rhythmyx/xrd_casHomePage/HomePage.html?sys_contentid=468&sys_variantid=100&sys_revision=20&sys_authtype=0&sys_context=0&sys_command=editrc
```

NOTE: Currently Site Explorer only supports virtual sites (non-published Sites in your Rhythmyx directory); the Home Page (URL) must be the address of a virtual site.

Publishing Root Location - Optional. The directory location where the files will be saved.

Publisher - Drop list. The Publisher that publishes to this Site.

Status - Drop list. Indicates whether the Site is *Active* or *Inactive*.

Folder Root - Optional. Used with Site Folder Publishing. Content Explorer location of the Root Folder in the format //Sites/<root folder name>.

Global Template Optional. Specifies the Global Template to use when rendering this Site. Note that you can override this template for a specific Folder or a specific Variant. Options include all Global Templates set up in the system.

Nav Theme - Optional. Used with Managed Navigation. Theme indicator for stylesheet coding.

FTP Information:

IP Address - Optional. The IP address where files are located for an FTP output target.

Port Number - Optional. The FTP port number.

UserID - Optional. The user ID used to access the database.

Password - Optional. The password used to access the database.

Copy Site Page

Use the Copy Site Page to *create a duplicate of an existing Site registration* (see "[Copying a Site Registration](#)" on page 44). The new Site registration contains all of the data of the original registration, but has a new name, which you define when creating the copy.

To access the Copy Site Page, on the Sites Administrator, click the [Copy Site](#) link.



The screenshot shows a web form titled "Copy Site" in the top right corner. The form is divided into two main sections: "Source Site" and "New Site".

- Source Site:** This section contains a label "Source Site" and a text prompt "Select the Site you want to copy". Below this is a dropdown menu with "Corporate Investments" selected.
- New Site:** This section contains a label "New Site" and a text prompt "*Name". Below this is an empty text input field.

At the bottom of the form, there are two buttons: "Create" and "Cancel".

Figure 16: Copy Site Page

Field Descriptions

Source Site Drop List. Name of the Site you want to copy.

New Site Required. Free-form name for the new Site.

Site Maintenance Procedures

You can perform the following maintenance operations on Site registrations:

- define a new Site registration;
- *Copying a Site Registration* (on page 44);
- *edit a Site registration* (see "Editing a Site" on page 44);
- *delete a Site registration* (see "Deleting a Site" on page 44).

If you have published a Site, you can *view a virtual map* (see "Viewing a Virtual Edition Map" on page 45) of the content that currently exists on the Site.

Defining a New Site

To define a new Site:

- 1 Go to the Site Editor.
- 2 Click New Site.
Rhythmyx displays the Edit Site Properties page.
- 3 Enter the mandatory **Site Name** and optional **Site Description**.
- 4 Enter the **Site Address (URL)** for the Site. The Site Address is the URL where the Site is located. This can be a fully qualified path if the output target is *filesystem* or a virtual directory if the output target is *ftp*.
- 5 Enter the **Publishing Root Location**. The Publishing Root Location is the directory location where the files will be saved.
- 6 Select the **Publisher** that publishes to the Site. Options are all Publishers defined in the system.
- 7 Select the **Status** of the Site. Options are *Active* and *Inactive*.
- 8 If you are using Site Folder Publishing, enter the **Folder Root** of the Site in Content Explorer.
- 9 Specify the **Global Template** you want to use for the Site. Note that you can override the Global Template in a Folder or in a specific Variant.
- 10 If you are using Managed Navigation, enter a **Nav Theme** if it is used in the stylesheet.
- 11 If this is a remote Site to which the Publisher will FTP information, fill in the **FTP Information** fields. Enter the Site's **IP Address** and **Port Number**. Optionally enter the **User Id** and **Password** required to access the database.
- 12 Click [**Save**] to save the new Site record.

Copying a Site Registration

If you have a Site registration that already works (such as the Enterprise Investments and Corporate Investments Sites in FastForward), you may find it easier to create a new Site by copying and modifying that Site registration rather than creating a new Site registration from scratch. You can then modify the registration data to match the data for the new Site.

Note that if you copy a Site in Content Explorer, the Copy Site wizard automatically creates a new registration with updated data. You do not need to create a new Site registration.

To copy a Site Registration:

- 1 On the Sites Administrator, click the [Copy Site](#) link.
Rhythmyx displays the *Copy Site page* (on page 42).
- 2 In the **Source Site** drop list, choose the Site you want to duplicate.
- 3 In the **New Site** field, enter a name for the new Site.
- 4 Click the **[Copy]** button.
Rhythmyx processes your request and returns you to the Sites Administrator.

Editing a Site

To edit a Site:

- 1 Go to the Site Editor.
- 2 Click name of the Site you want to edit.
Rhythmyx displays the Edit Site Properties page.
- 3 Modify the fields you want to change. The **Site Name** field is mandatory.
- 4 Click **[Save]** to save your edits.

Deleting a Site

To delete a Site:

- 1 Go to the Site Editor.
- 2 Click the delete button  in the row of the Site you want to delete.
- 3 Rhythmyx displays a confirmation message. Click **[OK]** to delete the Site, or **[Cancel]** to abort the delete action.

Viewing a Virtual Edition Map

To view a Virtual Edition Map, a plan of all the content currently published to a Site, in the Site Editor, click the File Tree icon  in the row of the Site whose map you want to view.

Site(id)	Description	File Tree	Last Publication Date
 TestSite(302)			

Figure 17: Site Entry in Site Editor

Rhythmyx displays a Site map of the current Site.

Filename	Operation	Status	CMS Link
C:/			
Rhythmyx/			
Publisher/			
webapps/			
Xroads40/			
images/			
	publish	success	System Menu
	publish	success	Image - Products
	publish	success	System Menu
	publish	success	Image - News
	publish	success	System Menu
			Image - Events
			System Menu
			Image -
			Downloads
			System Menu ...

Figure 18: Virtual Edition Map

Click on the **Filename** to view the page on the Web site; click on the **CMS Link** to view the page as assembled by the Rhythmyx Content Assembler.

Rhythmyx displays the same map for any Publication when you click the date/time stamp of a Publication in the Publishing log. See *Publishing Details* (see "[Publication Details: Publication Maps](#)" on page 95).

CHAPTER 3

Content Lists

A content list is a Rhythmyx resource that defines the rules that determine which content items to extract from the database for Publishing. Divide a Site into separate content lists to give site managers control over the way the Publication looks on the Site. Generally, a Content List is defined for each content type that generates a full page, and for any binary content type, which includes images. Content types that generate snippets do not generally require Content Lists.

In many cases, a different Content List may be specified for different Variants. For example, a Site might include one Content List to Publish full-size Image content items, and another to Publish thumbnails of the same content items.

Content lists also define whether Rhythmyx will re-Publish the whole Site (a Full Publish) or only those content items that have been added or updated since the last Publication run (an Incremental Publish). For each content type or Variant, a separate Content List is required for Full Publishing and for Incremental Publishing.

Content Lists also define the content that Rhythmyx will remove from the Site, or unpublish. Unpublishing can be part of a Full Publish or Incremental Publish Content List, but to simplify the definition of the content lists and provide greater control over both Publishing and unpublishing, it is more common to define a separate unpublish Content List for each content type or variant.

After you register a Content List, you can include it in an Edition in the Publishing Administrator.

Creating a Content List Application

Content List applications are based on the Content List Document Type Definition, which is provided by Percussion Software. The specific mappings of the elements and attributes of this DTD in each resource, and the data selectors defined for the resource, define the type of publishing supported in the resource.

Content List applications are generally organized by Site, Edition type, and content type. In most cases, a Content List application will consist of the following resources:

- Publish
- Unpublish

Each resource can perform both full and incremental publishing/unpublishing if you include the `sys_IncrementalContentFilter` exit. (see "[Incremental Content Lists](#)" on page 61)

Content Lists for Manual, Automatic, and Recovery/Mirror editions are generally each in separate applications.

To create a Content List application:

- 1 Create a new Rhythmyx application.
- 2 Use the procedure to create a resource starting with an existing DTD to create the Content List resources. Content List resources should be query resources (query resources extract data from the database).
- 3 Map the `contentlist.dtd` in each resource.
- 4 If you publish large Content Lists (of 4000 items or more), your Publisher may return “Out of Memory” exceptions or SOAP timeout exceptions, and errors in large Content Lists may not be included in the Publisher log. To avoid these problems, configure your Content List application to publish Content Lists in chunks. The Publisher will report the result of each chunk to the Publisher log. To configure your application to publish in chunks, add a Result Pager to the resource, and set Max rows per page to 500 or another number of Content Items that your Publisher can handle.
- 5 Add the `sys_FlushCache` pre-exit to the resource. This exit will flush all caches on a publishing hub server at the outset of the publishing run.
- 6 If caching is enabled, add the `sys_FlushAssemblerCache` pre-exit to the resource once for each variant that includes an auto-index, and set the `variantid` parameter in the pre-exit equal to the id of the variant. Set the other parameters in the exit to null or an empty string.

NOTE: You may include the `sys_FlushAssemblerCache` pre-exit on all content list application resources (once for each variant) if you’re not sure which ones include auto-indexes.
- 7 If you use a macro in the mapper for a Content List resource:
 - c) check **Return Empty XML**.
 - d) add the `sys_emptyDoc` post-exit and set the parameter `rootName` to `contentlist`.
- 8 Save the new application.

NOTE: Applications with more than eight resources tend to run slowly. If an application has more than six resources, consider dividing it into two applications.

Mapping a Content List Resource

Map your content list resource in the Rhythmyx Workbench to provide data for fields in the `contentlist.dtd`.

The following table shows sample mappings . Use them as general principles for mapping a Content List resource for a Full Publish.

Make necessary modifications for Incremental Publish Content Lists, Incremental Unpublish Content Lists, and content lists published from different Workflow States.

DTD Element	Map to	Description
contentlist/@deliverytype	sys_Literal UDF	<p>Identifies the delivery format of the content.</p> <p>Options:</p> <ul style="list-style-type: none"> ▪ <i>filesystem</i> (publishes the content as HTML files) ▪ <i>ftp</i> (publishes the content to an FTP site). ▪ Other values indicate a custom publisher plugin. <p>The value must match the configuration parameter specified for the plugin class; otherwise, the error message: <i>The plug-in for</i></p>
contentlist/@context	PSXParam/sys_context	Identifies the reference to the publishing location.
contentlist/contentitem/@variantid	<p>Depends on the content type.</p> <ul style="list-style-type: none"> ▪ If the content type permits the user to select the Variant, map to the table column that holds the Variant ID in the backend table that stores the Content Editor's local fields. ▪ If Publishing automatically selects several Variants for the content type (for example, image and thumbnail Variants of an image content type), map to PSXParam/sys_variantid. 	Recovers the identifier of the Variant to be published.
contentlist/contentitem/@contentid	CONTENTSTATUS.CONTENTID table column	Recovers the identifier of the content to be published.
contentlist/contentitem/@revision	CONTENTSTATUS.CURRENTREVISION table column	Recovers the revision identifier of the content to be published.

DTD Element	Map to	Description
contentlist/contentitem/@unpublish	sys_Literal UDF	<p>Unpublish is a Boolean flag that indicates whether a Content Item is eligible to be published.</p> <p>If Unpublish = Y, then the Publisher will remove the content item.</p> <p>If Unpublish = N, then the Publisher will publish the content.</p>
contentlist/delivery/location	sys_casGeneratePubLocation UDF	<p>This function generates the location where the content is published.</p> <p>Map the sys_casGeneratePubLocation parameters as follows:</p> <p>ContentID - Backend Column, CONTENTSTATUS.CONTENTID</p> <p>Revision - Backend Column, CONTENTSTATUS.CURRENTREVISION</p> <p>VariantID - Literal: Variant ID of the page variant to be published.</p> <p>context - Optional. Literal: Value used instead of sys_context for the context ID.</p>
contentlist/contentitem/title	CONTENTSTATUS.TITLE table column	Recovers the title of the content from the CONTENTSTATUS table.

DTD Element	Map to	Description
contentlist/contentitem/ contenturl	This element is mapped to the sys_MakeAbsLink UDF.	<p>This is a link to the assembler application that will create the content that will be extracted.</p> <p>Map the sys_MakeAbsLinkSecureEx parameters as follows:</p> <p>UseSecure – Set to no if you are not using SSL publishing; set to yes if you are using SSL publishing.</p> <p>Host – If unspecified, the extension takes the simple host name from the originating request. Since SSL can only read host names in DNS format, specify the host in DNS format if you are using SSL publishing.</p> <p>Port – If specified, overrides the port specified by the originating request. Resource - CONTENTVARIANTS.ASSEMBLYURL</p> <p>ParamName1 - sys_contentid</p> <p>ParamValue1 - CONTENTSTATUS.CONTENTID</p> <p>ParamName2 - sys_variantid</p> <p>ParamValue2 - Depends on the content type. If the content type permits the user to select the Variant, map sys_variantid to the backend column where the Variant selection is stored. If Publishing automatically selects several Variants for the content type (for example, image and thumbnail Variants of an image content type), map sys_variantid to PSXParam/sys_variantid</p> <p>ParamName3 - sys_context</p> <p>ParamValue3 - PSXParam/sys_context</p> <p>ParamName4 - sys_authtype</p> <p>ParamValue4 - PSXParam/sys_authtype</p> <p>ParamName5 - sys_Revision</p> <p>ParamValue5 - PSXMacro/\$contextDependentRevision</p>

DTD Element	Map to	Description
contentlist/contentitem/modifydate	CONTENTSTATUS.CONTENTMODIFIEDDATE table column	This parameter recovers the date the content was modified.
contentlist/contentitem/modifyuser	CONTENTSTATUS.CONTENTLASTMODIFIER table column	This parameter recovers the name of the last user to modify the content.
contentlist/contentitem/expiredate	CONTENTSTATUS.CONTENTEXPIRYDATE table column	This parameter recovers the expiration date of the content.
contentlist/contentitem/contenttype	CONTENTSTATUS.CONTENTTYPEID table column	This parameter recovers the identifier of the content type of the content.

Publishing from Different States

To define a Content List to Publish content from a specific workflow State:

- 1 Do not map the contentlist/contentitem@unpublish attribute.
- 2 Add the STATES table to the backend datatank.
- 3 Join the CONTENTSTATUS.STATEID to the STATES.STATEID and CONTENTSTATUS.WORKFLOWAPPID to the STATES.WORKFLOWAPPID columns.
- 4 In the Selector, select content items in the State you want: STATE.STATENAME = (State). The example below shows the Selector Properties to publish content in the QA State:

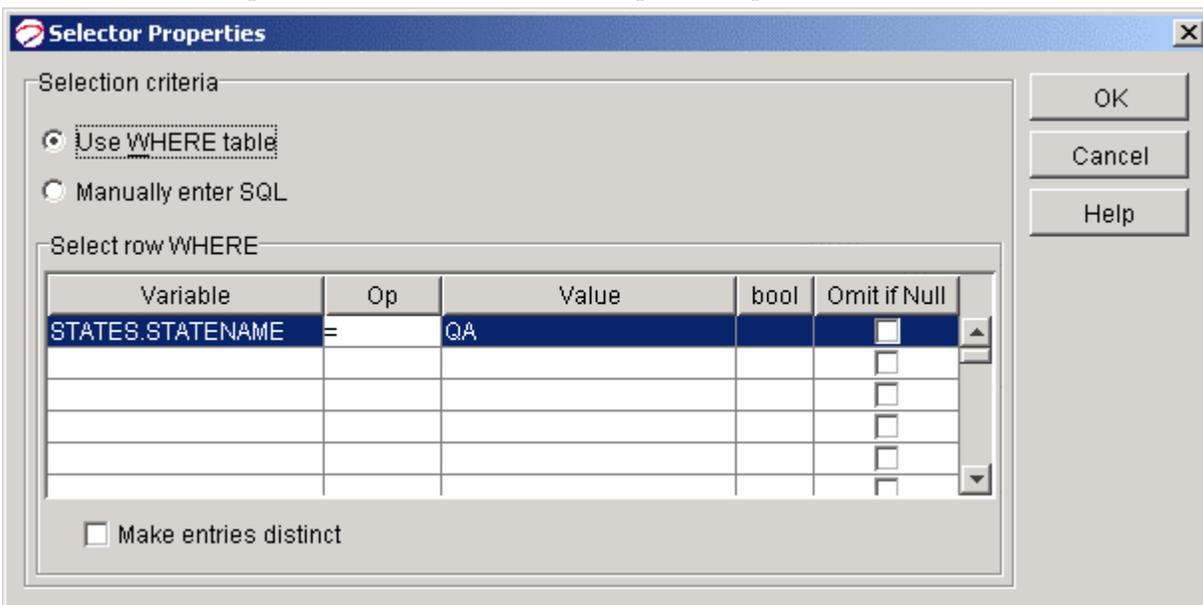


Figure 19: Selector Properties to Publish Content in the QA State

Mirror Editions

A Mirror Edition duplicates an Edition on another site or Web server. A Mirror Edition can serve as a backup, an additional location for users to access your Website, or a site mirror behind a redirector. To create a mirror edition, you must create a content list for that edition and map it to the table RXSITEITEMS, which stores the published content items you want to copy to the alternate site.

To create a mirror edition:

- 1 **Create a mirror edition content list resource** (see "[Creating a Mirror Edition Content List](#)" on page 55).
- 2 **Register the mirror edition content list** (see "[Registering a Mirror Site Content List](#)" on page 56).
- 3 Register the mirror site as you would register a standard site.
- 4 **Register the mirror edition** (see "[Defining a New Edition](#)" on page 69).

- In the **Destination Site** drop list, choose the name of the mirror site.
- In the **Edition Type** drop list, choose Mirror.
- In the **Mirror Source Site** drop list, choose the name of the original site.

Edit Edition Properties	
*Edition Name	Mirror Site
Description	Edition to Mirror Original Site
*Destination Site	Site B
Edition Type	Mirror
Recovery Publication(id)	(Recovery only)
Mirror Source Site	Site A (Mirror only)
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 20: Example Mirror Edition Registration

- 5 Run the original edition. Then run the mirror edition.

Creating a Mirror Edition Content List

Instead of selecting content items from the table that stores items for the specific content type, select content items from the RXSITEITEMS table. Select items where RXSITEITEMS.SITEID equals the site id of the site you want to mirror. In addition, you may set up other selection criteria to ensure that the selected items were published to the original site

- 1 Create the content list application.
- 2 When you add the backend datatank for your contentlist, add the RXSITEITEMS table and the other tables required for content assembly. Join the content type table to the RXSITEITEMS table on the CONTENTID column.
- 3 Create selection data for the mirror site. Set RXSITEITEMS.SITED equal to the system parameter sys_srcsiteid, which holds the original site ID:

```
RXSITEITEMS.SITEID=PSXSingleHtmlParameter/sys_srcsiteid
```

- 4 Include other selector properties that will ensure that you are selecting the exact content that was published to the original site. In the example below, the additional criteria select items with a specific content type, a successful publication status, a positive site location, and a publishing operation of "publish."

Selector Properties

Selection criteria

Use WHERE table

Manually enter SQL

Select row WHERE

Variable	Op	Value	O...
RXSITEITEMS.SITEID	=	PSXSingleHtmlParameter/sys_srcsiteid	<input type="checkbox"/>
CONTENTSTATUS.CONTENTTYPEID	=	321	<input type="checkbox"/>
RXSITEITEMS.PUBSTATUS	=	success	<input type="checkbox"/>
RXSITEITEMS.LOCATION	IS NOT NULL		<input type="checkbox"/>
RXSITEITEMS.PUBOPERATION	=	publish	<input type="checkbox"/>

Make entries distinct

Figure 21: Example Mirror Edition Selector

- 5 Map the fields in your content list resource using RXSITEITEMS as a backend table in place of the table that stores the content type's items (for example, RXARTICLE or RXBRIEF).

Registering a Mirror Site Content List

- 1 Go to the Edit Content List page to register the mirror edition content list.
- 2 Enter a **Name** and optional **Description**.
- 3 Enter the **URL**.
- 4 In the **Edition Type** drop list, choose *Mirror*.

5 Click [Save].

New Site			
Site(id)	Description	File Tree	Last Publication Date
✗ Site A(302)		⌄ ⌄ ⌄	
✗ Site B(303)		⌄ ⌄ ⌄	

Figure 22: New Site Page for an Example Mirror Edition. Use the id in the column Site(id) to fill the "original site" parameter of the content list URL (see the following graphic).

Edit Content List

Content List(id): Articles Mirror(1)

*Name:

Description:

URL:

*Edition Type: ▼

Save

Figure 23: Example Mirror Edition Content List Registration

Unpublishing a Mirror Edition

When you unpublish the source edition that a mirror edition duplicates, in most cases you will also unpublish the mirror edition. To set up a resource to unpublish a mirror edition, follow the procedure for publishing a mirror edition, except:

- Include a selection criterion that looks for items in the RXSITEITEMS table with a PUBOPERATION of "unpublish."

Selector Properties

Selection criteria

Use WHERE table

Manually enter SQL

Select row WHERE

Variable	Op	Value	b...	O...
RXSITEITEMS.SITEID	=	PSXSingleHtmlParameter/sys_srcsiteid	A...	<input type="checkbox"/>
CONTENTSTATUS.CONTENTTYPEID	=	321	A...	<input type="checkbox"/>
RXSITEITEMS.PUBSTATUS	=	success	A...	<input type="checkbox"/>
RXSITEITEMS.LOCATION	IS NOT NULL		A...	<input type="checkbox"/>
RXSITEITEMS.PUBOPERATION	=	unpublish		<input type="checkbox"/>

Make entries distinct

Figure 24: Selector Showing Values to Unpublish a Mirror Edition

- In the Mapper, set the contentlist/contentitem/@unpublish parameter to "yes":

Registering Content Lists

You must register a Content List to make it available to be included in an Edition. When you click Content list on the Publishing Administrator, Rhythmyx displays the Content List Administrator. You can sort the list of Content Lists by Content List name (default) or by Content List ID. Click on the link for the sort option you want to use.

Content List Administration			New Content List
Content List(id)	URL	Edition Type	
Mirror Full(301)	/Rhythmyx/rx_Support_pub/mirror_dist.xml?delivery=filesystem		
Mirror Incremental (302)	/Rhythmyx/rx_Support_pub/mirror_dist.xml?inc=y,i&delivery=filesystem		
Mirror Unpublish (303)	/Rhythmyx/rx_Support_pub/mirror_unpub_dist.xml?delivery=filesystem		
Site Root Full (310)	/Rhythmyx/rx_Support_pub/folder_dist.xml?valid=y,i&delivery=filesystem&pubOp=pub		
Site Root Incremental (311)	/Rhythmyx/rx_Support_pub/folder_dist.xml?valid=y,i&inc=y&delivery=filesystem&pubOp=pub		
Unpublish(315)	/Rhythmyx/rx_Support_pub/unpub_dist.xml?delivery=filesystem		

Figure 25: Content List Administration

Edit Content List Page

Use the Edit Content List page to create and maintain content list registrations.

Content List(id): Articles(1)		Edit Content List
*Name	<input type="text" value="Articles"/>	
Description	<input type="text" value="Content list for Articles"/>	
URL	<input type="text" value="/Rhythmyx/rx_pubContentLists/contentlist_article.xml"/>	
*Edition Type	<input type="text" value="Automatic"/>	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>		

Figure 26: Edit Content List Page

Field Descriptions

Name Mandatory. The name of the content list.

Description Optional. Description of the content list.

URL Optional. The URL of the content list application.

Edition Type Drop list. The type of edition for which the content list is define. Options are *Automatic*, *Manual*, *Mirror*, and *Recovery*.

Registering a New Content List

To register a new Content List:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
- 2 On the Publishing tab, click a sort option under Content Lists.
Rhythmyx displays the Content List Administrator.
- 3 Click New Content List.
Rhythmyx displays a blank Edit Content List page.
- 4 Enter the **Name** and optional **Description** of the Content List.
- 5 Enter the **URL** of the Content List.
- 6 Choose the **Edition Type** of the Content List from the drop list.
- 7 Click [**Save**] to save the new Content List record.

Deleting a Content List Registration

To delete a Content List registration:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
- 2 On the Publishing tab, click Content Lists.
Rhythmyx displays the Content List Administrator.
- 3 Click the delete button  in the row of the Content List you want to delete.
- 4 Rhythmyx displays a confirmation message. Click [**OK**] to delete the Content List. Click [**Cancel**] to abort the delete action.

Editing a Content List Registration

To change the registration information of a Content List:

- 1 On the Rhythmyx Content Explorer, click the Publishing tab.
Rhythmyx displays the Publishing Administrator.
- 2 On the Publishing tab, under Content Lists, click a sort option.
Rhythmyx displays the Content List Administrator.
- 3 Click the name of the Content List whose registration information you want to change.
Rhythmyx displays the Edit Content List page, with the current data for the Content List record.
- 4 Modify the fields you want to change.
- 5 Click [**Save**] to save the new Content List record.

Incremental Publishing

Incremental Publishing publishes only Content Items that you have changed since they were last published and new Content Items. Unlike Full Publishing, Incremental Publishing does not publish every Content Item in a Public State.

Your initial publishing run that publishes content to your Site for the first time will perform Full Publishing. However, scheduled publishing runs that regularly update your site after your initial publishing run should perform Incremental Publishing. Since fewer items are published during Incremental Publishing, you use fewer system resources and decrease processing time.

Incremental Content Lists

A Content List for Incremental Publishing or Incremental Unpublishing uses the same Content List application with the same *mappings* (see "[Mapping a Content List Resource](#)" on page 49) as the Content List for a Full Publish or Full Unpublish; however, the post-exit `sys_IncrementalContentFilter` must be attached to the content list resource. `sys_IncrementalContentFilter` takes two parameters:

- `queryrequest` specifies the name of the request that checks if items on the Content List have already been published or unpublished.
- `switchparameter` specifies the name of the parameter that turns on incremental publishing or unpublishing when it is set to `yes`. (Note: The switch allows you to use the same resource for full and incremental publishing.)

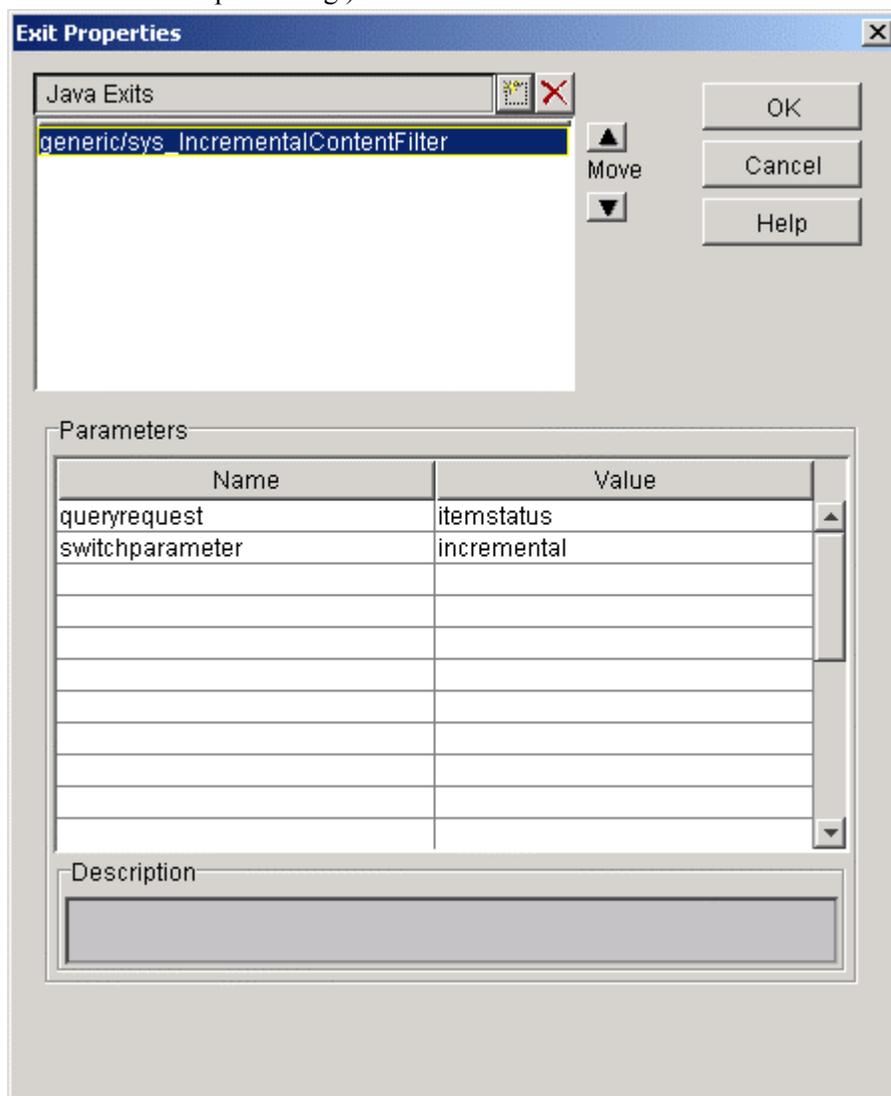


Figure 27: `sys_IncrementalContentFilter` mapping

You must add the `queryrequest` specified in the parameters to your Content List application, and set the `switchparameter` specified to `yes` in your Content List URL, for example:

- `/Rhythmyx/rx_pubContentLists/contentlist_generic.xml?variantid=101&incremental=yes`

For more information about using the `sys_IncrementalContentFilter` exit including instructions on setting up the `queryrequest` resource selector and mapper, see *sys_IncrementalContentFilter* (on page 62).

sys_IncrementalContentFilter

Context:

Java/global/percussion/generic

Description:

This extension filters a content list, removing items which have already been published or unpublished. It performs an internal request to find the entry in the `RXSITEITEMS` table that corresponds to the current content item. If a valid entry is found, it removes the item from the content list. If no valid entry is found, it leaves the item in the content list.

This exit lets you use the same query resource for both full and incremental content lists. The second parameter, `switchparameter`, is optional.

If you specify the `switchparameter` name as `incremental` in the extension registration and your content list resource is `rx_pubContentLists/contentlist_generic.xml`:

- the resource returns a "filtered" content list if you include the parameter in the content list URL and set it to `yes`, for example:
`/Rhythmyx/rx_pubContentLists/contentlist_generic.xml?variantid=101&incremental=yes`
- the resource returns a "full" content list if you do not include the parameter in the content list URL (or you include it but do not set it to `yes`), for example:
`/Rhythmyx/rx_pubContentLists/contentlist_generic.xml?variantid=101`

If you do not specify the `switchparameter` name in the extension registration, the resource always returns a "filtered" content list.

You must create the internal request that finds the item in `RXSITEITEMS`. Add it to the content list application by performing the following steps:

- 1 Open the Content List application in the Rhythmyx Workbench.
- 2 Drag `<Rhythmyx root>/DTD/contentlist.dtd` onto the application window.
- 3 Rename the request `itemstatus`.
- 4 Open the Resource Editor and add the `RXSITEITEMS` table to the backend `datatank`.

5 Open the Selector and define the following conditions for the query:

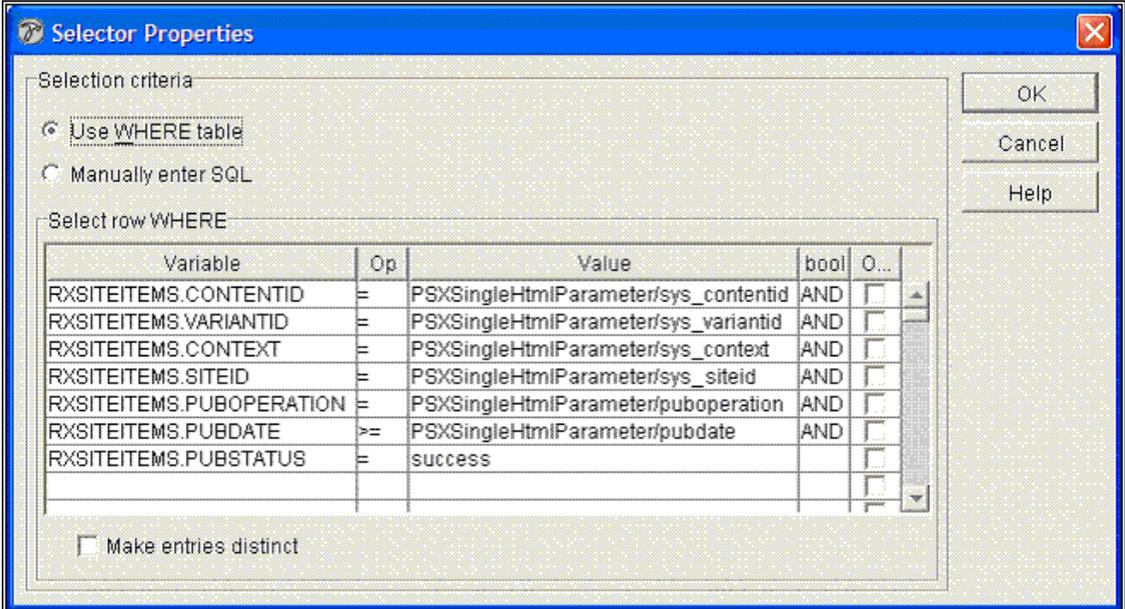


Figure 28: Selector to retrieve Items from Rxsiteitems table.

This selection returns a single row (one content item) from the Rxsiteitems table. REVISIONID is not required because only one revision of a Content Item should be present on a site.

6 Open the mapper and map Rxsiteitems columns to their equivalents in contentlist.dtd. The mapping should resemble the following graphic, but exact mappings are not important, because the exit only tests the presence or absence of a result document.

It is important that you check **Return empty XML** at the bottom of the mapper. If there is no match on the query, the exit expects an empty document (which appears as `</contentitem>`), not a document with empty subnodes (such as `<contentitem><title/> <contenturl/> . . . </contentitem/>`). If it receives a document with empty subnodes, it will attempt to process it, which will result in an error.

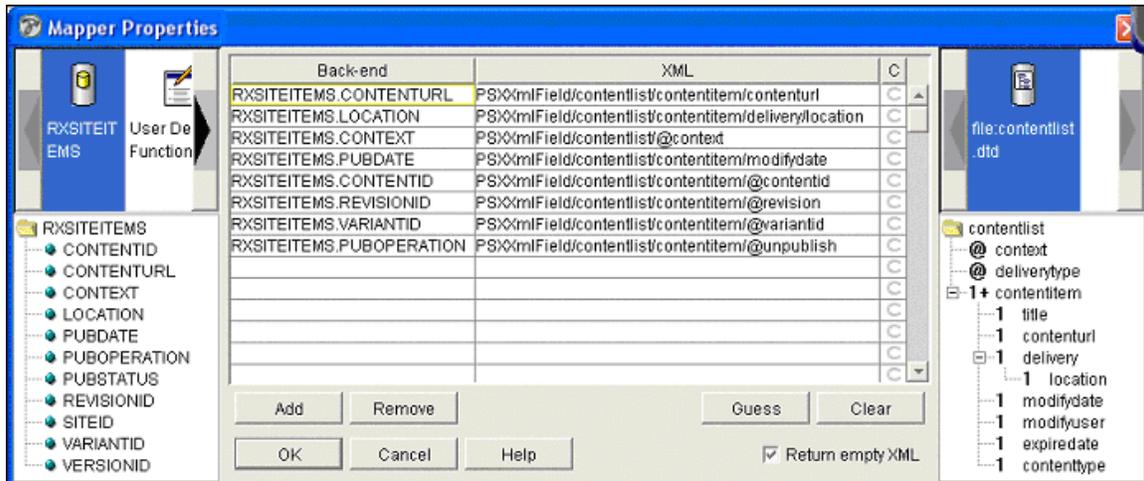


Figure 29: Mapping the support application for `sys_IncrementalContentFilter`

- When you add the exit to a content list resource, set `queryrequest` equal to `<application name>/itemstatus` and set `switchparameter` as specified above.

Class name:

`com.percussion.extensions.general.PSIncrementalContentFilter`

Interface:

`com.percussion.extension.IPSResultDocumentProcessor`

Parameters:

Name	Data Type	Description
<code>queryrequest</code>	String	Name of internal request for site item lookup
<code>switchparameter</code>	String	Optional. Name of the HTML parameter for switching the filter on and off. Any name is valid. If the parameter is included and set to <code>yes</code> , the filter is turned on. If the parameter is included but omitted from the content list URL or set to any value other than <code>yes</code> , the filter is turned off. If the parameter is not included, the exit always returns a filtered content list.

CHAPTER 4

Editions

An Edition specifies a set of Content Lists and the sequence in which to publish them. The sequence is important for three reasons:

- First, because a page is composed from multiple items, and these items themselves may be groupings of other items, more elemental items should be published first, followed by higher level items, and finally by pages. In other words, images should be published before articles because articles will typically include images. Articles should be published before indices because index items typically include article items.
- Second, different Content Lists typically have different publishing rules, which specify the response to errors generated in earlier content lists, such as omitting an item if a child item is not published, publishing the item without the child item, or publishing the item with an error message.
- Finally, sequence is critical when you unpublish and publish to the same URL. It is possible for one Content List to both unpublish and publish, but if you do not control the order correctly, you could publish new content and then immediately unpublish it. To avoid this problem, build separate content lists to unpublish content from the content lists that publish the content. When defining the Edition, sequence the unpublish content lists before publish content lists so only old content is unpublished, rather than inadvertently unpublishing new content.

	Publish	Edition Name (id)	Description	Destination Site (id)	Edition Type
	X	Sample Edition (1)	Sample Edition	Sample Site (1)	
	X	Daily Edition (2011)	Daily run edition	Production Site (201)	

Figure 30: Editions Page

Edition Types

Each Site typically consists of multiple Editions. Currently, Rhythmyx defines four types of Editions:

Edition Type	Icon	Description
Automatic		Editions created by pre-defined Content List queries
Manual		Editions created from special Content Lists composed by hand by a contributor, editor, or Webmaster
Recovery		Editions that republish an earlier Edition of a Site or its contents
Mirror		Editions that copy content from one Site to another; for example, copying content from a QA server to a production machine

Publishing an Edition

You can publish an edition manually from the Edition editor by clicking the Publish button  in the row of the Edition you want to Publish.

	Publish	Edition Name (id)	Description	Destination Site (id)	Edition Type
		Sample Edition (1)	Sample Edition	Sample Site (1)	
		Daily Edition (301)	Daily run edition	Production Site (301)	

Figure 31: Manual Publish icons beside Editions

Rhythmyx displays the Publishing page while publishing.

Publication Status Details

Publisher Status: Edition is in progress

Currently Publishing:

Edition: Full Enterprise Investments

Content list: Site Root Full (1 of 1 lists)

Figure 32: Publishing Page

The Status page automatically refreshes every 15 seconds; you can also click the [Refresh Status] button to refresh it. The page shows which of the content lists Rhythmyx is publishing, in the format Content List Name (*number of total lists*) For example, Articles (6 of 13 lists).

Edit Edition Properties Page

Use the Edit Edition Properties page to *define* (see "Defining a New Edition" on page 69) and *maintain* (see "Editing an Edition" on page 72) Editions and to manage the content lists assigned to each Edition.

Edit Edition Properties

Edition(id): Full Enterprise Investments(301)

*Edition Name: Full Enterprise Investments

Description: Publish all items in public state

*Destination Site: Enterprise Investments

Edition Type: Automatic

Recovery Publication (id): (Recovery only)

Mirror Source Site: --Choose-- (Mirror only)

Save Cancel

Edit Edition: Allowed Content [Add Content List](#)

Content List Name (id)	Sequence	Authorization Type	Context	Preview
✗ Site Root Full (310)	1	Site Folder Content	Publish	

Figure 33: Edit Edition Properties Page

Field Descriptions

Edition Name Mandatory. The name of the Edition.

Description Optional. Description of the Edition.

Destination Site Drop list. The Site to which the Edition will be published.

Edition Type Drop list. The type of Edition. Options are *Automatic*, *Manual*, *Recovery*, and *Mirror*. Unless you are an advanced user, use this page to create Automatic Editions only.

Recovery Publication Only required if the Edition Type is *Recovery*. The ID of the Publication you want to recover.

Mirror Source Site Only Required if the Edition Type is *Mirror*. The Site that is the source of the content to publish to the Destination Site.

Edit Editions Allowed Content Table of Content Lists included in the Edition.

Defining a New Edition

Note: Unless you are an advanced user, use this screen to create Automatic Editions only.

To define a new Edition:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
- 2 On the Publishing tab, click Editions.
Rhythmyx displays the Edition Editor.
- 3 Click New Edition.
Rhythmyx displays the Edit Edition Properties page.
- 4 Enter the **Name** and optional **Description** of the Edition.
- 5 Choose the **Edition Type** from the drop list. Options are *Automatic*, *Manual*, *Recovery*, and *Mirror*. Unless you are an advanced user, you should select *Automatic*.
- 6 Choose the **Destination Site** from the drop list. Options are all Sites defined in your system.
- 7 If the **Edition Type** is *Recovery*, enter the Publication ID of the Edition you want to recover.
- 8 If the **Edition Type** is *Mirror*, choose the Source Site from the drop list. Options are all sites defined in your system.
- 9 Add Content Lists to the Edition.
- 10 Click [OK] to save the Edition record.

Edition Content Lists

Each Edition consists of a series of Content Lists that will be published. You can add Content Lists to an Edition, edit a Content List assignment, or delete a Content List from an Edition.

To add a Content List, in the lower portion of the Edit Edition Properties page, click [Add Content List](#).

To view a Content List that already exists, click the Content List name.

Rhythmyx displays the Add/Edit Edition Content List page.

Figure 34: Add/Edit Edition Content List Page

Field Descriptions

Content List The content list being assigned to the Edition. Options are all content lists associated with the Edition Type that have not yet been assigned to an Edition.

Sequence Optional. The sequence in the content list will be processed.

Authorization Type Drop list. Determines how the edition will treat related content. Options are:

- *All Content*: publishes all related content, whether or not it is in a Public State.
- *All Public Content*: publishes only related content that is in a Public State.
- *Content On Site*: publishes only content that has already been published to the Site.

Context Mandatory. Specifies the directory structure to which the content will be published. Each context is a unique directory structure. The specific details of the directory structures depend on the implementation. Options are 0, 1, 2, and 3.

Adding Content Lists to an Edition

To add a Content List to an Edition:

- 1 In the Editions Editor, open the Edition to which you want to add a Content List.
- 2 Click [Add Content List](#).
Rhythmyx displays the Edition Content List page.
- 3 Choose the **Content List** you want to add to the Edition from the drop list. Options include all existing Content Lists associated with the Edition Type that have not already been assigned to an Edition.
- 4 In the **Sequence** field, enter the integer describing the order in which the Content List should be processed by the Edition (1, 2, 3, 4...).

- 5 Choose the **Authorization Type** for the Content List.
- 6 Choose the **Context**.
- 7 Click [**Add**] to add the Content List to the Edition.

Editing a Content List Assignment to an Edition

To edit a content list assignment to an Edition:

- 1 In the Edit Editions Properties page, click the name of the content list that you want to edit. Rhythmyx displays the Edition Content List page.
- 2 You can modify the **Sequence**, **Authorization Type**, and **Context**.
- 3 Click [**Save**] to save your changes.

Previewing Content List Output

To preview the output of a Content List:

- 1 In the Edit Edition Properties page under the **Edit Edition: Allowed Content** table, locate the Content List.
- 2 In the row for the Content List click  in the Preview column. Rhythmyx opens the XML file that includes the Content List output.

Deleting a Content List From an Edition

To delete a Content List from an Edition:

- 1 In the Editions Editor, click the name of the Edition from which you want to delete the Content List. Rhythmyx displays the Edit Edition Properties page with the current data for the edition.
- 2 Click the delete button  in the row of the Content List you want to delete.
- 3 Rhythmyx displays a warning message. Click [**OK**] to confirm the delete action. Click [**Cancel**] to abort the delete action.

Editing an Edition

To edit an Edition:

- 1** In the Rhythmyx Content Explorer, click the Publishing tab.
- 2** On the Publishing tab, click a sort option under Editions.
Rhythmyx displays the Edition Editor.
- 3** Click the name of the Edition you want to edit.
Rhythmyx displays the Edit Edition Properties page, with the current data for the Edition.
- 4** Make your changes. The **Edition name** and **Destination Site** fields are mandatory. You can also add, edit, or delete a Content List.
- 5** Click [**OK**] to save the Edition record.

Deleting an Edition

To delete an Edition:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
- 2 On the Publishing tab, under Editions, click a sort option to display the Editions Editor. Rhythmyx displays the Edition Editor.
- 3 Click the delete button  in the row of the Edition you want to delete.
- 4 Rhythmyx displays a confirmation message. Click **[OK]** to delete the Edition. Click **[Cancel]** to abort the delete action.

Implementing a Custom AuthType

Authorization Types (or Auth Types) provide a mechanism for filtering the related Content Items for which link URLs will be published when publishing content. This functionality basically defines what Snippets will appear on a Page. AuthType has two major purposes:

- to prevent broken links that point from a Public Content Item to related content that is not Public; and
- to prevent the inclusion of embedded snippets of non-Public content in otherwise Public output pages and snippets.

Rhythmyx comes with three predefined Auth Types:

- All Content (authtype=0): Output will be assembled with all related Content Items, regardless of whether they are Public or not.
- All Public Content (authtype=1): Output will be assembled with only those related Content Items that are Public.
- Content On Site (authtype=2; deprecated): Output will be assembled with only those related Content Items that have previously been published successfully (in other words, the related Content Item is listed in the publishing log tables for the Site to which you are publishing.

Creating a Custom AuthType (in brief)

- 1 Define a new Keyword value under Authorization Types for the new AuthType.
- 2 Add a new value to the AuthType properties file. This value must match the Choice Value of the new Authorization Type Keyword. The format is:
`authtype.[numeric value for authtype]= [application name]/[resource name]`
- 3 Create an application with a resource to define the LinkURL for the related content.

Adding new Keywords

When a Content List is associated with an Edition, the user selects an Authorization Type.

*Content List	--Choose--
Sequence	
*Authorization Type	--Choose--
*Context	--Choose--

Save Cancel

Figure 35: Associating an AuthType to a Content List

AuthType and Content Lists, though logically related, are not tied to each other in any way except for their proximity to one another in the Allowed Content List dialog. Items defined by an AuthType are exclusive of the list of Items generated by the Content List. It is important to understand this separation. It is possible to generate a list of Items to publish (Content List) but include links to Items (AuthType) that are not published. Adding a new AuthType begins with the creation of a Keyword to populate the Authorization Type dropdown described above.

Creating a new Keyword

- 1 Log into the Content Explorer.
- 2 Click on the System tab.
- 3 Click on Keywords [By Name].
- 4 Click on [Authorization Types].
- 5 Click on [Add Choice].
- 6 Define all necessary Values for the Keyword fields:

Choice Label - The value displayed in the Authorization Types dropdown.

Description - Descriptive text explaining the use of the AuthType.

Choice Value - The value used to select the AuthType resource. This value is tested against properties listed in Authtypes.properties. This value should be an integer greater than 300. Using values greater than 300 avoids the likelihood of using predefined system values.

Sort Order - The Authorization Types list overrides the alphabetical listing of choices in the dropdown list.

- 7 Click on [Save] to insert the new Authorization Type.

Creating a Content Assembly Support Application

The list of Items that will represent the AuthType are defined in custom Query Resource. This resource is responsible for the generation of the list of Items to be included, then the building of a properly formed linkurl. This linkurl is based on the casSupport.dtd.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT casSupport (linkurl*)>
<!ELEMENT linkurl (#PCDATA)>
<!ATTLIST linkurl
  contentid CDATA #REQUIRED
  relateditemid CDATA #IMPLIED
  rxcontext CDATA #REQUIRED
  slotid CDATA #REQUIRED
  slotname CDATA #REQUIRED
  sys_authtype CDATA #REQUIRED
  variantid CDATA #REQUIRED
  sys_siteid CDATA #IMPLIED
  sys_folderid CDATA #IMPLIED
>
```

The creation of this resource can be started with the casSupport.dtd or by cloning and existing AuthType resource. Cloning resources limits the likelihood of mistakes and will be the method described in this section. It is possible to manually map the values to each of the attributes defined in the DTD if necessary.

Creating a Custom AuthType Application

- 1 Log into the Workbench.
- 2 Select the Applications tab.
- 3 Locate and open the sys_casSupport application.
- 4 Select and copy [ctrl-c] the casSupport_1 resource.
- 5 Open a new blank application and paste [ctrl-v] the resource into the new application.
- 6 Right-click the application. Rhythmyx displays a popup list. Select [Request Properties] and provide a new **Request name** for the resource. The naming convention [application type][Application name]_[authtype value] should be used.

Example: casCustomAuthType_301

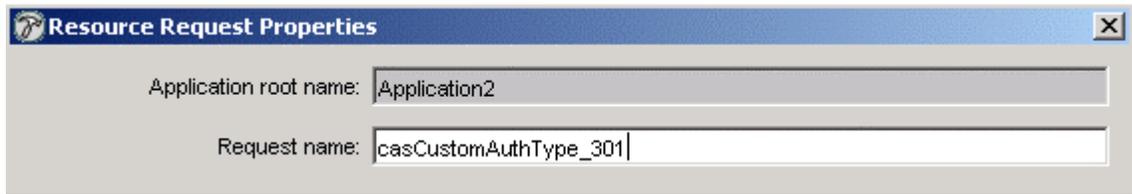


Figure 36: Naming the Custom AuthType Resource

- 7 Click [OK] to close the Request Properties dialog.
- 8 Right-click the application. Rhythmyx displays a popup list. Select [Properties].
- 9 Open the Selector and provide the necessary Selection Criteria to generate the desired list of Items. The cloned resource uses manual SQL to select all Items in Workflow States defined as "y" or "i". Generally, the minimum Selection Criteria includes Items that match the current Items RELATEDCONTENT.CONTENTID and RELATEDCONTENT.REVISIONID values.

Variable	...	Value	bool
RXRELATEDCONTENT.CONTENTID	=	PSXSingleHtmlParameter/sys_contentid	AND
RXRELATEDCONTENT.REVISIONID	=	PSXSingleHtmlParameter/sys_revision	

Figure 37: Minimum Selection Conditions

- 10 Close the Selector and open the Pager. Though this will not affect results, a Pager must be present and define atleast a single valid value for sorting Items returned by the Selector.
- 11 Close the Pager and open the Mapper. This cloned resource includes all the necessary mappings for a properly formed <linkurl>. It is possible to build different <linkurl> values based on conditional mappings. The casSupport_0 resource conditionally displays one of three possible <linkurl> values depending on whether or not the Item is being previewed by the person who currently has the item checked out or if the Item is not checked out at all. This shows the current revision for all those previewing the Item except for the person who has the Item checked out. If the person previewing the Item is also the person who has the Item checked out, the active version is displayed.
- 12 Make any desired changes to the mapping and close the dialog.

- 13 From the Insert Menu, select [Page] to associate a default stylesheet to the resource.

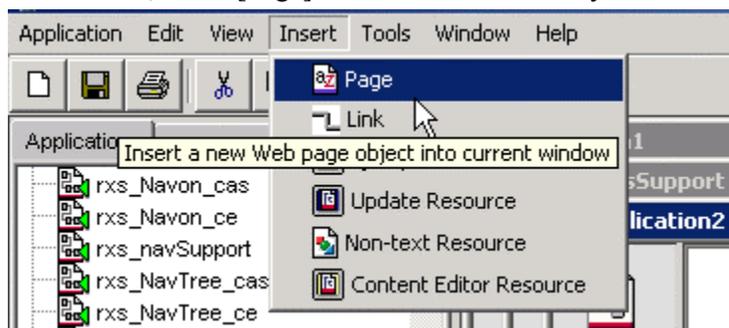


Figure 38: Adding a Page to a Resource

- 14 Link the Page to the resource by dragging a connection from the Page to the Resource.

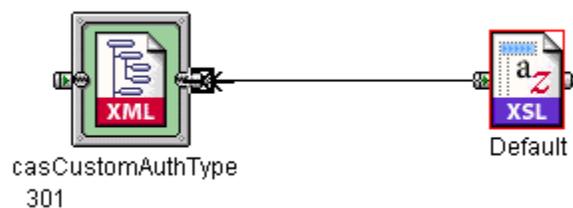


Figure 39: Connecting a Page to a Resource

- 15 Save the Application. During the first [Save] the developer will be prompted for an application name. The general format for naming an application is [department]_[application type][Application description]

Example: rxs_casCustomAuthType

Note: The prefix sys_ should not be used for custom applications.

- 16 [Start] and [Close] the application.

Maintaining the Authtypes.properties file

Once the Developer defines a new Authorization Type keyword, the mapping of this value to an application and resource needs to be established in the Authtypes.properties file. This file marries the numeric Authorization Type value to a specific Application and resource responsible for creating the list of Items to be included in a list of <InfoLink>s. These <InfoLink> values populate an Item's XML to describe the list of Related Content Items (Snippets).

Note: This process requires the restart of the Rhythmyx Server.

- 1 On the machine hosting the Rhythmyx Server, locate the Authtypes.properties file in the Rhythmyx/rxconfig/Server directory.
- 2 Open the file and add a new entry for the newly registered Authorization Type. This entry should include a comment to describe the AuthType usage. By default, AuthTypes 0 and 1 are defined.

```
#Auth type implementation resource for All Content
authtype.0 = sys_casSupport/casSupport_0
#Auth type implementation resource for All Public Content
authtype.1 = sys_casSupport/casSupport_1
```

- 3 The format for this line entry is:

```
authtype.[Keyword Value for Authtype] = [Application name]/[Resource name]
```

The application and resource names must match the names used during the creation of the custom application for the generation of the InfoLink URL. The Keyword Value must match the value defined for the AuthType when creating the Authorization Type Keyword.

- 4 [Save] and [Close] the file.
- 5 Restart the Rhythmyx Server.

CHAPTER 6

Developing and Managing Contexts and Link Generation Schemes

Context

Context is a Rhythmyx mechanism that provides one Variant the ability to produce different markup outputs that achieve the same rendering when the Variant is published to (or previewed from) different site environments.

Different site environments may place common resources used for rendering, such as images and stylesheets, in different locations. The Location Scheme of the Context specifies how the locations of the Published resources are defined within each site environment, and thus the different linking paths between pages. The location scheme may map to actual physical locations in a file system or database, or may only represent a scheme for finding the resource, such as a URL and query string that resolves to the resource.

Different Contexts may require different formatting. You can choose to use different stylesheets to generate the formatting for each Context, or you can define a set of Context variables to use in conditional statements that control formatting within a single stylesheet. For example, in the context of a dynamic application server, the background color for a Variant must be unspecified (because the color is inherited from tables generated by the application server that surround the variant when it is rendered). When viewing the same Variant on the preview site, the color must be defined. A Context Variable for color allows one Variant to work in both Contexts.

Context also gives you the flexibility to produce different formatting markup to achieve similar effects as a Variant processes different content. For example, a site might use content translated into different languages. You might want content to appear the same width and shape regardless of language. Different languages have different space requirements, however, so the specific table formatting markup used for one language might not fit another language as well. In this case, you could define each language as a Context and define different table formatting and spacing variables for each language Context. The Variant then produces a set of pages that appear the same, but use different markup for each language.

The context is indicated by a number:

Context	Meaning
0	Context 0 is the preview context. The file system for context 0 resides on the Rhythmyx Server.
1, 2, 3	Context 1, 2, and 3 are production publishing contexts. The file systems for these contexts reside on machines other than the Rhythmyx server. Use context 1 for a main Web site and contexts 2 and 3 to publish on additional file systems. For example, use contexts 2 or 3 to publish a new version of a Web site that is in progress or to publish the site on a backup server

Context Editor

The Publishing Administrator includes a Contexts Editor for adding and modifying contexts. Rhythmyx stores the context names and the context descriptions in the RXCONTEXT table. To get to the Contexts Editor, go to the Publishing Administrator. Under Contexts, click the By Name or By ID sort.



Contexts	
New Context	
Context(id)	Description
 Preview(0)	Preview Context
 Publish(1)	Publish Context

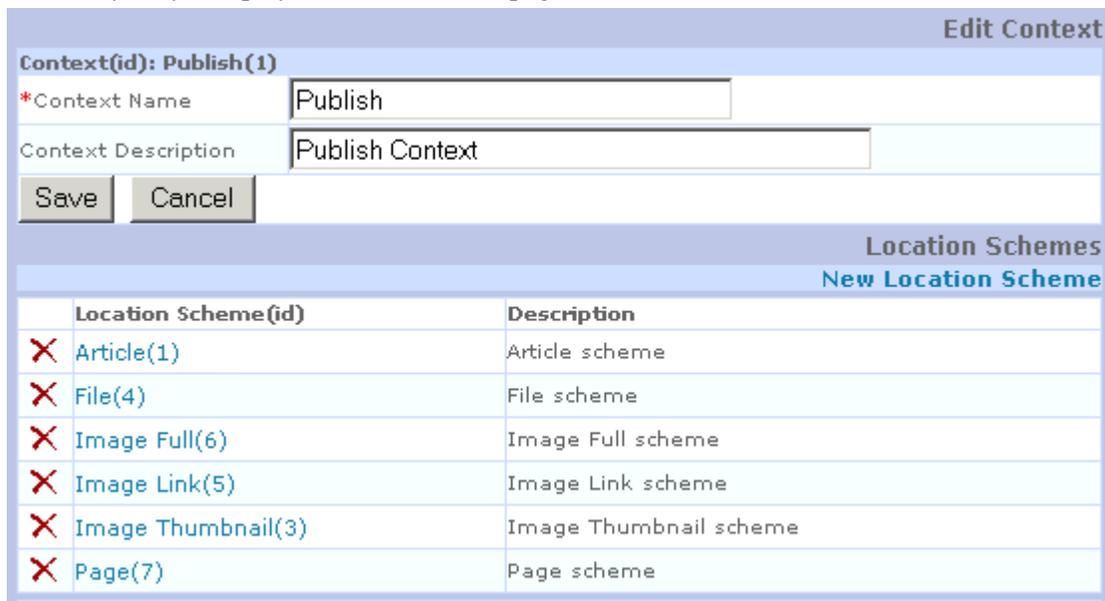
Figure 40: Context Editor

By default, Rhythmyx defines the Preview and Publish contexts. You cannot modify or delete the Preview context from the Contexts Editor.

Adding Contexts and Location Schemes

To add a context:

- 1 In the Publishing Administrator, click the [By Name](#) or [By ID](#) sort under Contexts in the Navigation Bar.
Rhythmyx displays the Contexts Editor.
- 2 In the upper right corner of the Contexts Editor, click [New Context](#).
Rhythmyx displays the Edit Context page.



Edit Context	
Context(id): Publish(1)	
*Context Name	<input type="text" value="Publish"/>
Context Description	<input type="text" value="Publish Context"/>
<input type="button" value="Save"/>	<input type="button" value="Cancel"/>
Location Schemes	
New Location Scheme	
Location Scheme(id)	Description
 Article(1)	Article scheme
 File(4)	File scheme
 Image Full(6)	Image Full scheme
 Image Link(5)	Image Link scheme
 Image Thumbnail(3)	Image Thumbnail scheme
 Page(7)	Page scheme

Figure 41: Edit Context Page

- 3 In the **Context Name** field, enter a name for the context; this field is required.
- 4 In the **Context Description** field, you may enter a description for the context. This field is optional.
- 5 Click [**Save**].

Rhythmyx returns to the Contexts Editor, which displays the new context.

		Contexts
		New Context
	Context(id)	Description
✗	Database(301)	Database Context
✗	Preview(0)	Preview Context
✗	Publish(1)	Publish Context

Figure 42: Contexts Editor displaying new context

- 6 Click the [Context\(id\)](#) link to define a location scheme for the context.

Editing Contexts and Location Schemes

To edit a context:

- 1 In the Publishing Administrator, click the [By Name](#) or [By ID](#) sort under Contexts in the Navigation Bar.

Rhythmyx displays the Contexts Editor.

- 2 Click the [Context\(id\)](#) of the context that you want to edit. You cannot edit Preview context from the Contexts Editor.

Rhythmyx displays the Edit Context page for the selected context.

- 3 Change the **Context Name** or the **Context Description**.
- 4 To edit the location scheme, click the [Location Scheme\(id\)](#).
- 5 Click [**Save**].

Rhythmyx saves the edits and returns to the Contexts editor.

Deleting Contexts and Location Schemes

To delete a context:

- 1 In the Publishing Administrator, click the [By Name](#) or [By ID](#) sort under Contexts in the Navigation Bar.

Rhythmyx displays the Contexts Editor.

- 2 Click the delete icon  beside the **Context(id)** of the context that you want to delete.

Rhythmyx deletes the context.

Developing and Managing Link Generation Schemes

Each Context in Rhythmyx requires a different scheme to generate the links between pages. To generate the links, the `sys_GeneratePubLocation` UDF is included in the content list application. Map the UDF to the `contentlist/delivery@location` attribute.

Scheme generators are extensions of the `com.percussion.extensions.IPSAssemblyLocation` interface. The `sys_GeneratePubLocation` UDF calls the scheme generator specified for the context and content type.

In the Contexts Editor, after you have saved a context once, you can add location schemes, which specify the scheme generator. Add location schemes for each variant of each content type associated with the context.



Edit Context			
Context(id): Publish(1)			
*Context Name	Publish		
Context Description	Publish Context		
Save	Cancel		
Location Schemes			
New Location Scheme			
	Location Scheme(id)	Used by Type:Variant(id)	Description
✗	Article(1)	Article:Published HTML Page(1)	Article scheme
✗	File(4)	File:File Binary(9)	File scheme
✗	Image Full(6)	Image:Full Image (19)	Image Full scheme
✗	Image Link(5)	Image:Image Link (17)	Image Link scheme
✗	Image Thumbnail(3)	Image:Thumbnail Only(18)	Image Thumbnail scheme

Figure 43: Edit Context page with link for adding a Location Scheme

Default Location Scheme Generators

Rhythmyx is delivered with four location scheme generators:

- `Java/global/percussion/contentassembler/sys_casDefaultAssemblyLocation`

This implementation takes three parameters:

- `root`

- path
- suffix

The locations will be assembled as "root + path + contentid + suffix".

- `Java/global/percussion/contentassembler/sys_casConcatAssemblyLocation`

This implementation takes a variable number of parameters. The location is generated by concatenating all parameters: "param[0] + param[1] + ... + param[n]".

- `Java/global/percussion/contentassembler/sys_casConcatWithIdAssemblyLocation`

This implementation takes at least 2 parameters. The first parameter specifies the index, where to append the contentid. All other parameters will be concatenated in the order received. In other words, if parameter 0 is received as 2: "param[1] + param[2] + contentid + param[3] + ... + param[n]".

- `Java/global/percussion/contentassembler/sys_casGenericAssemblyLocation`

This implementation calls a query resource to generate the location scheme. For details see "*Defining Complex Location Schemes* (on page 91)".

For further information, consult the *Javadocs* (see Rhythmyx Javadocs URL - [\Javadocs\index.html](#)) for these classes.

Defining Scheme Generators for Contexts

To define a location scheme for a context:

- 1 Go to the Contexts Editor.
- 2 Click the Context(id) of the context that needs a location scheme.
Rhythmyx displays the Edit Context page with location schemes table.
- 3 Click [New Location Scheme](#).
Rhythmyx displays the Edit Location Scheme page.

Figure 44: Edit Location Scheme page

- 4 In the **Name** field enter a name for the location scheme (required).
- 5 In the **Description** field, enter a description for the location scheme (optional).

- 6 In the **Generator** field, enter the path of the location scheme generator (required).
- 7 In the **Content Type** drop list, select the content type to which this location scheme applies (required).
- 8 In the **Variant Type** drop list, select the variant to which this location scheme applies.
- 9 Click [**Save**].
- 10 Rhythmyx returns to the Edit Context page and displays the new location scheme.

Location Schemes	
New Location Scheme	
Location Scheme(id)	Description
 Test Location Scheme(301)	Test Location Scheme

Figure 45: Edit Context page with new location scheme

- 11 Click on the Location Scheme(id) to add location scheme parameters.

Editing Scheme Generators in Contexts

To edit a location scheme for a context:

- 1 In the Edit Context page, click the Location Scheme(id) of the location scheme that you want to edit.
Rhythmyx displays the Edit Location Scheme page.
- 2 Change the **Name**, **Description**, **Generator**, **Contact Type**, and/or **Variant Type**. The **Name**, **Generator**, **Contact Type**, and **Variant Type** fields are required.
- 3 To define a new location scheme parameter, click [New Location Scheme Parameters](#). To edit a location scheme parameter, click the Name(id) of the parameter that you want to edit.
- 4 Click [**Save**].
Rhythmyx saves the changes and returns to the Edit Context page.

Deleting Scheme Generators in Contexts

To delete a location scheme:

In the Edit Context page, click the delete icon  beside the Location Scheme(id) of the location scheme that you want to delete.

Rhythmyx deletes the location scheme.

Location Scheme Parameters

Location scheme generators may require parameters. In the Contexts Editor, after you have saved a location scheme for a context once, you can define the location scheme parameters from the Edit Location Scheme page.

Two types of parameters are available: *string* and *backend column*.

- Use the type *string* if you need a single, hard-coded value for the parameter.
- If you need a dynamic value, choose the type *backend column*. You must specify the backend column (using the format `BACKEND_TABLE_NAME.BACKEND_TABLE_COLUMN`; for example, `CONTENTSTATUS.CONTENTSTARTDATE`) where the dynamic value is stored. You can specify any column of a Content Type's metadata tables. Rhythmyx retrieves the first value from the column where the Content ID and the Revision ID match the Content Item being processed. Each parameter you specify for the Location Scheme makes its own database query. (NOTE: Columns from child editors are not valid for use in Location Schemes generated by concatenation, such as when using the `sys_cas_DefaultAssemblyLocation` or `sys_casConcatAssemblyLocation`. If you need to use a column from a child editor table in your Location Scheme, see *Defining Complex Location Schemes* (on page 91).)

The following graphic shows the parameters defined for the default location scheme generator `sys_casConcatAssemblyLocation`.

Edit Location Scheme

Location Scheme(id): Test Location Scheme(301)

*Name

Description

*Generator

*Content Type

*Variant Type

Location Scheme Parameters

New Location Scheme Parameter

	Name(id)	Type	Sequence	Value
✗	Root	String	1	/Marketing/Articles/
✗	Middle	BackendColumn	2	CONTENTSTATUS.CONTENTSTARTDATE
✗	File	String	3	.html

Figure 46: Edit Location Scheme page with parameters defined

Adding Scheme Generator Properties

To define a location scheme parameter:

- 1 Click the Location Scheme(id) of the location scheme that requires a parameter.

Rhythmyx displays the Edit Location Scheme page with a Location Scheme Parameter table.

Figure 47: Edit Location Scheme page with Location Scheme Parameter table

- 2 Click New Location Scheme Parameters.

Rhythmyx displays the Edit Location Scheme Parameter page.

Figure 48: Edit Location Scheme Parameter page

- 3 In the **Name** field, enter a name for the parameter (required).
- 4 In the **Type** drop list, choose the parameter type (defaults to *string*).
String -- select to hard code the parameter value.
BackendColumn -- select to map the parameter value to a table value.
- 5 In the **Sequence** field, enter the numerical order in which the location scheme generator uses this parameter. The first sequence number is 0. (required).

- 6 In the **Value** field, enter the value of the parameter if the Type is *String*, or enter the backend table column (in the format BACKENDTABLENAME.BACKENDTABLECOLUMN) if the type is *BackendColumn* (required). For details about using the *backend column* option, see "[Location Scheme Parameters](#) (see "[Deleting Scheme Generators in Contexts](#)" on page 87)".
- 7 Click [Save].

Rhythmyx returns to the Edit Location Scheme page and displays the new location scheme parameter.

Edit Location Scheme

Location Scheme(id): Test Location Scheme(301)

*Name

Test Location Scheme

Description

Test Location Scheme

*Generator

Java/global/percussion/contentassembler/sys_casDefaultAssemblyLocation

*Content Type

Article

*Variant Type

Published HTML Page

Save

Cancel

Location Scheme Parameters

New Location Scheme Parameter

	Name(id)	Type	Sequence	Value
✘	Path	String	1	/Marketing/SnowRemoval/

Figure 49: Edit Location Scheme page with new location scheme parameter

Editing Scheme Generator Properties

To edit a location scheme parameter:

- 1 In the Edit Location Scheme page, click the Name(id) of the location scheme parameter that you want to edit.

Rhythmyx displays the Edit Location Scheme Parameter page.

- 2 Change the **Name**, **Type**, **Sequence**, and/or **Value**. All of the fields are required.
- 3 Click [Save].

Rhythmyx saves the changes and returns to the Edit Location Scheme page.

Deleting Scheme Generator Properties

To delete a location scheme parameter:

In the Edit Location Scheme page, click the delete icon  beside the Name(id) of the location scheme parameter that you want to delete.

Rhythmyx deletes the location scheme parameter.

Defining Complex Location Schemes

The standard location scheme generators, `sys_casDefaultAssemblyLocation`, `sys_casConcatAssemblyLocation`, and `sys_casConcatWithIdAssemblyLocation`, generate the location scheme by concatenating a set of parameters that you define when you create the Location Scheme.

In some cases, however, you may want to generate the Location Scheme more dynamically. For example, you may define the assembly location of pages based on the Community in which they are created, or the assembly location may be defined by the section of the Web site to which the user assigns the Content Item. In these cases, the standard location scheme generators will not work because their parameters are hard-coded. Instead, you will need to use the `sys_casGenericAssemblyLocation` generator, which uses a query resource to generate the location scheme.

To use the `sys_casGenericAssemblyLocation` generator, you will need to create a query resource that queries the database (most commonly the Content Type table of the Content Type for which you are defining the location scheme) to retrieve one or more portions of the path in the location scheme. The output document from this resource would be the location scheme.

For example, suppose a Web site defines the assembly location for its content based on the Community of the Content Item. Suppose further that in some Communities, content creators can assign pages to subsections of the Community section. First, you must define the query resource that generates the location scheme.

Typically, these query resources are part of a Publication support application or a URL builder application. The DTD for the resource does not require a formal structure, and a common practice is to use a simple alphabetical DTD, which you can define in the mapper. The following mapper shows an example based on the example above:

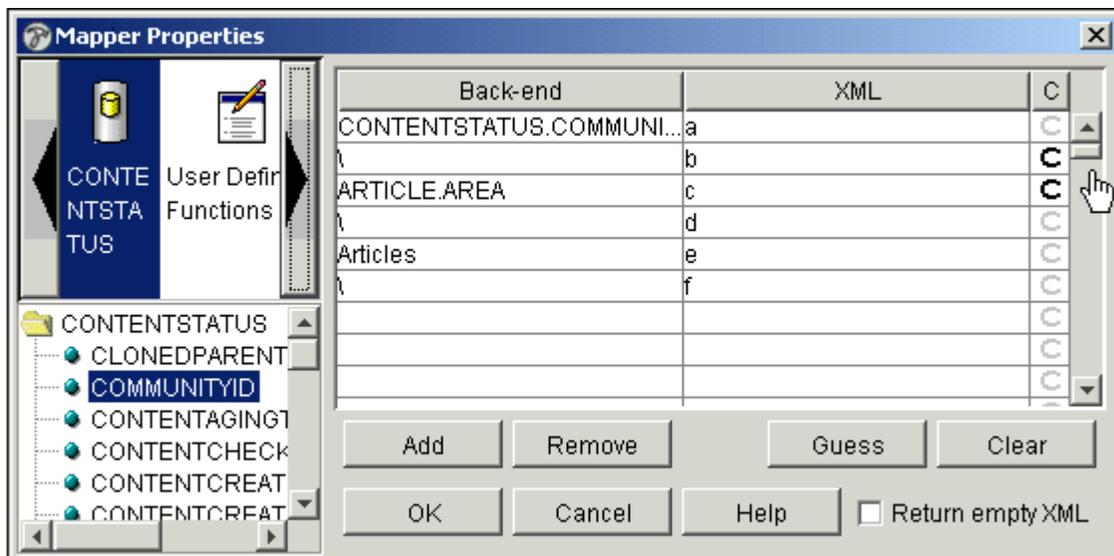


Figure 50: Example mapping of an alphabetical DTD

Since we are building a location scheme, we must be sure to include the backslashes required in the path.

Note that this mapping calls the CommunityID of the Content Item. More than likely, you would want to use the Community's name rather than its ID in the location scheme. You would probably want to use a UDF to transform the ID into the text name.

Note also that the second row, mapping a literal backslash ("\") to the node , and third row, mapping the value of the ARTICLE.ARE column to the node <c>, each include conditional statements. In the example, only some Communities assign pages to subsections. Therefore, we need conditional statements in the mapper that exclude these nodes if the parameter that passes requests these values is null.

You can also use XSL to transform the request data. If so, you must include the `sys_xdTextToDom` pre-exit to convert the incoming request data.

When defining the location scheme for the content, you would need to assign the `sys_casGenericAssemblyLocation` scheme generator. This scheme generator requires a single parameter, the URL of the query resource that generates the location scheme.

Reviewing Publications

Rhythmyx stores the results of each Publication run in the database. When you access the Publishing Administrator, the default page is the Publication Log (Publog), which displays these results. The record displayed for each run includes the following information:

- The date/time stamp of the publication run; clicking the link opens the *Publication Map* for the run.
- The name of the Content List in the Run; clicking the link opens the Content List registration.
- The name of the Edition in the Run; clicking the link opens the Edition registration.
- An icon indicating the type of Edition; hovering your mouse over the icon displays the Edition Type.
- An indicator of the Publication Status:
 - Success - Content List published successfully
 - Error - Content List did not publish successfully
 - Stopped by User - User stopped publishing of content list before it was complete.
 - In Progress - Content List is currently publishing.

Clicking the link opens an error log for the Publication run.

- Content Item statistics, including the number of Content Items that
 - were inserted ()
 - were updated()
 - were removed()
 - were skipped()
 - generated errors () (Note: if this column contains a value other than 0, the value is a link to a detailed log of errors for the publication run.)

You can sort the Publication Logs by

- date
- Site
- Publication ID
- Edition Name
- Edition Type
- error
- recovery

To sort logs, click the link under Publication Log for the type of sort you want to use.

Internet (301)						Publications by Site				
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status						
2005-02-25 15:11:26.0 (348:349)	Site Root Full (310)	Full Internet		Error	87	22	0	4	1	
2005-02-25 15:12:05.0 (349:350)	Site Root Full (310)	Full Internet		Success	1	109	0	4	0	

NOTE: If you specify that chunks of a Content List are to be published by specifying **Max results per page** in the Result Pager, Rhythmyx updates the Publication log each time it publishes a chunk of Content Items. Refreshing the Publication log page while a publication is running displays an updated entry for the chunk of Content Items recently published.

Publication Details: Publication Maps

When you click on the date/time stamp of a Publication in the Publication Log, Rhythmyx displays the publication details as a virtual map of the output of the run. Rhythmyx displays the same map of the current Publication when you click the File Tree icon in the Site Editor.

The virtual map is a tree diagram showing the location of each content item published in the Publication run, whether Rhythmyx succeeded or failed to publish the content item, and any links from the content item to another content item.

Use the Virtual Edition Map to see how a page appears on the Web Site by clicking on the Filename. To see if the Web site is properly displaying your Rhythmyx configuration, click on the CMS Link to view the page as assembled by the Rhythmyx content assembler.

To View the Virtual Edition Map:

- 1 Go to the Publication Log:

Publications by Site						
Corporate Investments (303)						
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status		
2005-07-06 09:34:26.0 (407:407)	Site Root Full (310)	Full Corporate Investments		Error		
2005-07-06 09:35:28.0 (408:408)	Site Root Full (310)	Full Corporate Investments		Success	168	0 0 0 0
Dynamic Preview Site (305)						
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status		
2005-06-27 10:45:47.0 (394:394)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Success	1	0 0 0 0
2005-06-27 14:56:20.0 (398:398)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Success	0	1 0 0 0
2005-06-27 15:00:43.0 (399:399)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Success	1	0 0 0 0
2005-06-28 10:31:41.0 (400:400)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Error		
2005-06-28 10:33:59.0 (401:401)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Error		

Figure 51: Publishing Administrator, Publications page

Click the time/date stamp of the Publication that you want to view.

2 Rhythmyx displays the Publication Details (Virtual Edition Map).

Filename	Operation	Status	CMS Link
C:/			
Rhythmyx/			
Publisher/			
webapps/			
Xroads40/			
images/			
	publish	success	System Menu
	publish	success	Image - Products
	publish	success	System Menu
	publish	success	Image - News
	publish	success	System Menu
			Image - Events
			System Menu
			Image -
			Downloads
			System Menu

Figure 52: Virtual Edition Map

- 3 Click on a **Filename** to view the page on the Web site. Click on a **CMS Link** to view the page as assembled by the Rhythmyx Content Assembler.

Viewing Detailed Logs

Rhythmyx keeps detailed logs of events during Publishing. You can access either a log filtered for errors or a complete log.

To view a log filtered for errors, click on the value in the error column of the Publication Logs whose results you want to view. Rhythmyx displays a detailed log consisting only of errors:

Log session start time: Fri Feb 25 15:11:26 EST 2005		
Time	Level	Message
6003922	ERROR	HTTP error for the request URL: http://bobj9955/Rhythmyx/ris_Home_cas/page.html?sys_revision=1&sys_siteid=301&sys_authtype=101&sys_contentid=466&sys_variantid Error code: 404
6003938	ERROR	
6003938	ERROR	Failed to publish content item.
<pre> com.percussion.publisher.client.PSContentFetchException: HTTP error for the request URL: http://bobj9955/Rhythmyx/ris_Home_cas/page.html?sys_revision=1&sys_siteid=30 at com.percussion.publisher.PSUtils.fetchContent(Unknown Source) at com.percussion.publisher.client.PSContentItem.publish(Unknown Source) at com.percussion.publisher.client.PSContentItem.process(Unknown Source) at com.percussion.publisher.client.PSContentPublisher.process(Unknown Source) at com.percussion.publisher.client.PSContentPublisher.execute(Unknown Source) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source) at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source) at java.lang.reflect.Method.invoke(Unknown Source) at org.apache.soap.server.RPCRouter.invoke(RPCRouter.java:146) at org.apache.soap.providers.RPCJavaProvider.invoke(RPCJavaProvider.java:129) at org.apache.soap.server.http.RPCRouterServlet.doPost(RPCRouterServlet.java:354) at javax.servlet.http.HttpServlet.service(HttpServlet.java:760) at javax.servlet.http.HttpServlet.service(HttpServlet.java:853) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:247) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:193) at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:256) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:643) at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480) at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995) at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:643) at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480) at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995) at org.apache.catalina.core.StandardContext.invoke(StandardContext.java:2416) at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:180) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:643) at org.apache.catalina.valves.ErrorDispatcherValve.invoke(ErrorDispatcherValve.java:171) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:641) at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:172) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:641) at org.apache.catalina.authenticator.SingleSignOn.invoke(SingleSignOn.java:376) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:641) at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480) at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995) at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:174) at org.apache.catalina.core.StandardPipeline\$StandardPipelineValveContext.invokeNext(StandardPipeline.java:643) at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:480) at org.apache.catalina.core.ContainerBase.invoke(ContainerBase.java:995) at org.apache.coyote.tomcat4.CoyoteAdapter.service(CoyoteAdapter.java:223) at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:801) at org.apache.coyote.http11.Http11Protocol\$Http11ConnectionHandler.processConnection(Http11Protocol.java:392) at org.apache.tomcat.util.net.TopWorkerThread.run(ThreadPoolEndpoint.java:565) at org.apache.tomcat.util.threads.ThreadPool\$ControlRunnable.run(ThreadPool.java:619) at java.lang.Thread.run(Unknown Source) </pre>		
6003953	ERROR	Items failed: 1

Figure 53: Detailed Publication Log showing only errors

To view a complete detailed log, click on the value in the Status Column. Rhythmyx displays the complete detailed log for the publication run.

Log session start time Fri Feb 25 15:11:26 EST 2005

Time	Level	Message
5997125	INFO	Starting publisher log. Version: Rhythmyx Publisher 5.6; Build:20050204. Date: Fri Feb 25 15:11:26 EST 2005
5997156	DEBUG	Begin Content List.
5997156	DEBUG	Begin content item.
5997156	DEBUG	Getting content item from URL : http://boj:9955/Rhythmyx/bs_Category_cas/p_category.html?sys_revision=1&sys_siteid=301&sys_authtype=101&sys_contentid=485&sys
5997172	DEBUG	PSFilePublisherHandler - Publishing item with contentid: 485
5997172	DEBUG	Creating file c:\inetpub\wwwroot\Mortgages and Home Finance\page485.html
5997234	DEBUG	Finished content item.

Figure 54: Section of a complete detailed Publication Log

Purging the Publication Log

Over time, you will build up a large backlog of publication logs. In order to keep the publication log manageable, you will want to purge it periodically. Purging the publications log deletes the log records permanently from the database, so you want to be sure that you have no need of a log record before you purge it.

To purge the publication log:

- 1 In the Rhythmyx Content Explorer, click the Publishing tab.
Rhythmyx displays the Publishing Administrator.
- 2 On the Publishing tab, click the Purge Pub Log option you want to use. (Note: The only option in the default version of the tab is By Site.)
- 3 Rhythmyx displays the Purge Publication Log page.

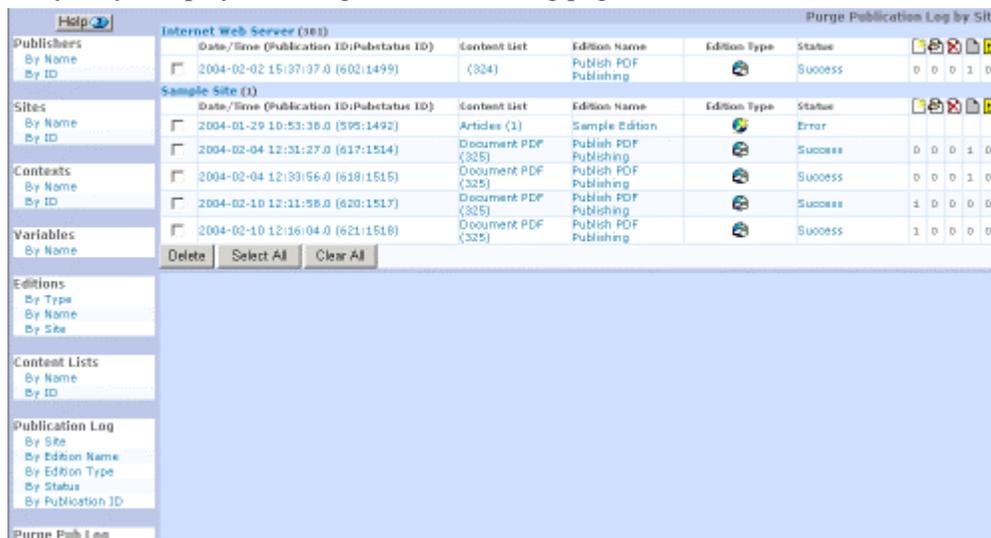


Figure 55: Purge Publication Log Page

- 4 Check the logs you want to purge. To purge all logs, click [**Select All**]. To clear your selections and start over, click [**Clear All**].
- 5 Click [**Delete**].
- 6 Rhythmyx will display a confirmation message. Click [**OK**] to confirm the delete action. Click [**Cancel**] to abort the delete action.

Monitoring Publication of Localized Content

Use the *Virtual Edition Map* (see "[Publication Details: Publication Maps](#)" on page 95) to monitor the publication of localized content.

If your Publishing Model is site-centric (publishes localized content to unique sites or destinations), you will have unique Sites and Editions for each Locale. Review the Virtual Edition Map for each Edition to determine whether the content of the Edition published correctly.

If your Publishing Model is content-centric (publishes all content to a single site or destination), you use a single Edition that includes the pages for all Localized versions of your content. After publishing this Edition, Rhythmyx creates a single Virtual Edition Map that shows the success or failure of each page. Click the different pages in the Virtual Edition Map to see if the different versions within the Edition published correctly.

See the *Internationalizing and Localizing Rhythmyx* document for more information about localization.

If Rhythmyx publishes your Editions or Edition, but does not publish some content items, *republish the failed content* (see "[Republishing Failed Content](#)" on page 101).

Republishing Failed Content

When you review the Publication Log and Virtual Edition Map for a publication, you may find that Rhythmyx published your edition but did not publish some content items. After you *resolve the problems* (see "[Appendix: Troubleshooting](#)" on page 109) causing the publication failure of these content items, you can republish just these items to your site by publishing an incremental Edition. You do not have to republish the entire Edition. See Incremental Content Lists for Publishing for help determining the selection criteria for the incremental content list.

Implementing Plugins

Publisher Plugins are Java programs that implement the `IPSPublisher` plugin interface. Rhythmyx configures them as Publisher parameters. You can create your own Publisher Plugins to customize a Rhythmyx Publisher.

Rhythmyx ships with 2 plugins that are required parameters for all Publishers:

- **FTP** -- The FTP plugin opens up an FTP connection to the target server and pushes the data
- **FileSystem** -- The FileSystem plugin works on a local filesystem. The Publisher is installed on the target server. The publisher waits to be fed a content list, then it pulls the content items from the Rhythmyx server and writes them to the local filesystem.

Implementing a User-Created Plugin

To publish an edition using a Publisher Plugin that you have created, complete the following steps:

- 1 **Register the plugin in the Rhythmyx CMS** (see "[Registering a Publisher Plugin](#)" on page [105](#)).
- 2 **Add the plugin class to the Tomcat server classpath** (see "[Including the Java Class in the Tomcat Server File](#)" on page [106](#)).
- 3 **Map to the plugin in your content list resource** (see "[Mapping to the Plugin in a Content List Resource](#)" on page [107](#)).

Registering a Publisher Plugin

Register a Publisher Plugin by adding it as a parameter to the Publisher with which you want to associate it.

To register a publisher plugin:

- 1 In the Publishing Administrator, open the Publisher with which you want to associate the plugin.
- 2 Click Add User Param.
Rhythmyx opens a blank Edit Config Param page.
- 3 In the **Name** field, enter a name for the publisher plugin. You will map this Name to the `@deliverytype` parameter in the content list.
- 4 In the **Value** field, enter the name of your new plugin class. Include the package name, and exclude the `.class` extension.
- 5 Optionally, enter a description of the plugin in the **Description** field.
- 6 Click [**Save**].

Including the Java Class in the Tomcat Server File

If you create a plugin, you must add the java class associated with your plugin to the server.

To add the new java class associated with your plugin to the Tomcat server:

- 1 Add the new plugin class to:
`<Rhythmyxroot>/AppServer/webapps/RxServices/WEB-INF/lib.`
- 2 Restart Tomcat.

Mapping to the Plugin in a Content List Resource

To map to your Publisher plugin in the content list resource:

- 1 Open the content list application in the Workbench.
- 2 Map the `deliverytype` parameter to the Name you assigned when registering the plugin, for example, `filesystem` or `database`.

Note: The Name is case-sensitive.

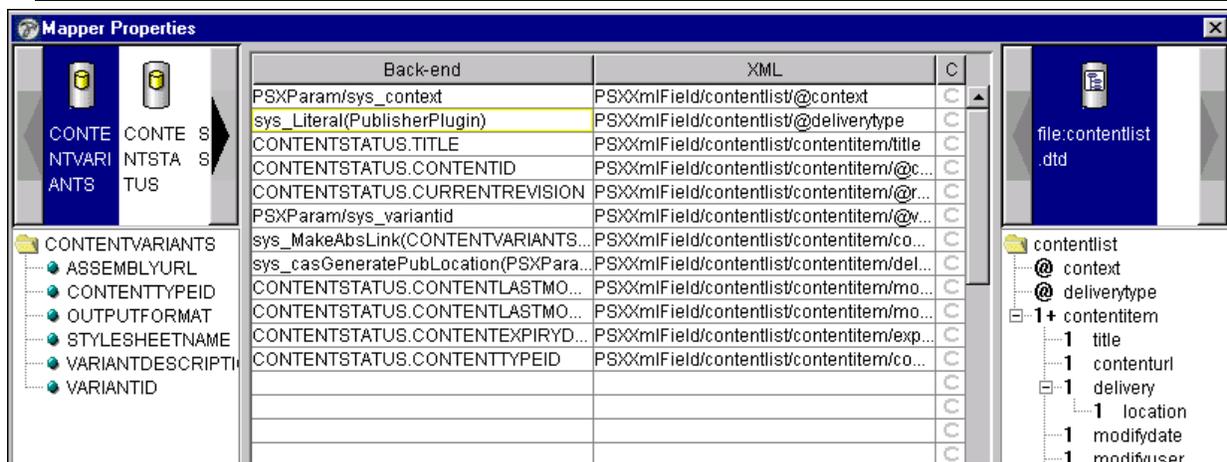


Figure 56: Mapper Properties in Workbench

- 3 Click [OK].

CHAPTER 9

Appendix: Troubleshooting

Troubleshooting Publishing

This section provides you with documentation for troubleshooting Rhythmyx Publishing problems. It includes the following:

- *Error List with links to resolutions for each error* (see "[Errors](#)" on page 111).
- *A table describing the different types of Publisher Logs* (see "[Error Logs in the Publisher](#)" on page 120)
- *Help for troubleshooting content lists.* (see "[Troubleshooting Content Lists](#)" on page 121)
- *Publishing FAQs.* (see "[Publishing FAQs](#)" on page 123)

Errors

Publication Log indicates that files have published correctly, but links to the files on the Web server are incorrect. (see "[Incorrect links to published files on Web server](#)" on page 112)

Error document displays "Access is denied" message. (see "[Access is Denied Error](#)" on page 113)

Publishing process resulted in an error, but the Publication Log and Error Document are empty. (see "[Publication Error Log is Empty](#)" on page 114)

Publication Log indicates that files were published correctly, but the files do not render correctly in the browser. (see "[Files Do Not Render Properly in Browser](#)" on page 115)

Error document displays "error opening socket" message (see "[Error Opening Socket](#)" on page 116).

RXSITEITEMS table is not updating correctly. (see "[RXSITEITEMS Is Updating Incorrectly](#)" on page 118)

Content is publishing without related content. (see "[Content is Publishing without Related Content](#)" on page 119)

Incorrect links to published files on Web server

Error:

Publication Log in the Publishing Administrator indicates that files have published correctly, but links in the *Publication Map* (see "[Publication Details: Publication Maps](#)" on page 95) to the files on the Web server are incorrect.

Resolution:

Fix the Publishing Root Location or the Site Address (URL) on the Edit Site Page. The Publishing Root Location must be an existing directory that an HTTP server listed in Site Address (URL) serves. If you use FTP publishing, the Publishing Root Location must be relative to the FTP account specified under FTP information. If you publish to a file system, the Publishing Root Location must be an absolute path (for example, c:\inetpub\wwwroot or /usr/local/web).

Access is Denied Error

Error:

Status link in Publication Log displays error document with "Access is denied" message.

Example Error Documents:

```
[Debug] Creating file
c:/inetpub/wwwroot/ftppublish/articles/art303.html
Error: Could not create directory 'c:' on server
FTP server message: 550 c:: Access is denied.
```

```
[Debug] Creating file ftpublish/articles/art303.html
Error: Could not create directory 'ftppublish' on server
FTP server message: 550 ftpublish: Access is denied.
```

Resolution:

The error document displays this error message if the FTP account does not have access to the directory listed in the Publishing Root Location on the Edit Site Page. For Publishing to work, the FTP account must have access to the directory listed in Publishing Root Location or privileges to create it under the FTP root.

Check the following:

- Directory Permissions (Does account have read-only access to this directory?)
- Account Permissions (Is account permitted to create directories?)
- Publishing Root Location (Is Publishing Root Location relative to the FTP account's root directory?)

Publication Error Log is Empty

Error:

Publishing Log in the Publishing Administrator displays "Error" for status, and the columns for items inserted, updated, removed, skipped, and with errors are empty. The status link accesses an empty error document.

Example Error Document:

```
Rhythmyx Publisher 4.0; Build:20020226 Thu Apr 04 15:40:06 EST 2002 -  
begin log
```

Resolution:

This error occurs when the Publisher fails before it can establish a connection with the target server.

Common causes are:

- **FTP IP Address** is incorrectly defined on Edit Site Page.
- **FTP Port Number** is incorrectly defined on Edit Site Page.
- **FTP User ID** is incorrectly defined on Edit Site Page.
- **FTP Password** is incorrectly defined on Edit Site Page.
- FTP Service is not available.
- Firewall is blocking ports between Publisher and Web Server.
- The server running the publisher cannot resolve the host name used for the Site (if you used a host name instead of an IP address). To fix this problem, create or correct the entry for the host in the local host file, or create or correct the entry on the host file at the DNS server.

Files Do Not Render Properly in Browser

Error:

Publication Log in Publishing Administrator indicates that publishing was successful, but the files do not render correctly in the browser.

Resolution:

The assembly application is not generating the html file as you intended. Check the application.

Error Opening Socket

Error:

Publication Log indicated error opening socket.

Example Publication Log 1:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <publicationstatus pubstatusid="316">
  <status>Error</status>
  <message>error opening socket: Connection refused: connect</message>
</publicationstatus>
```

Example Publication Log 2:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <publicationstatus pubstatusid="319">
  <status>Error</status>
  <message>error opening socket: pubserver</message>
</publicationstatus>
```

Resolution:

The Rhythmyx Server was unable to create a connection with the Publisher Service. Do the following:

- 1 If the Publisher (<http://publisher:port>) is not running, start the Publisher.
- 2 Determine if the Publisher Service is deployed properly on Tomcat:
- 3 Open a browser and enter: <http://publisher:port/soap/admin/index.html>, where publisher is the name or IP address of the publishing server's machine and port is the number of the port on which the publishing server listens for requests.
- 4 The browser displays the XML-SOAP Admin page.
- 5 Click **List**.
- 6 The browser displays the Service Listing page. Open the Service Listing Dialog. If you do not see a deployed service named: `urn:www.percussion.com/Publisher`, the SOAP service is not properly deployed on Tomcat. Check to make sure that you have a `deployedservices.ds` file in both `<Rhythmyxroot>/AppServer` and `<Rhythmyxroot>/AppServer/bin`.

If `deployedservices.ds` only exists in one of these locations, copy it from the other directory and restart the Publisher

If `deployedservices.ds` is missing, shut down the Publisher and reinstall the Publisher Component.

If this does not work, delete your Rhythmyx Publisher Registry key.

Check if a firewall exists between the Rhythmyx Server and Publisher server.

RXSITEITEMS Is Updating Incorrectly

Error:

The RXSITEITEMS table is not updating correctly.

Resolution:

The RXSITEITEMS table has 4 key columns: SITEID, CONTENTID, CONTEXT, and VARIANTID.

The Content List must supply the Publisher with these values. Check your content list mappings to make sure the content list is being populated correctly. (See *Troubleshooting Content Lists* (on page [121](#)) for more detailed troubleshooting help.

Content is Publishing without Related Content

Error:

Content is publishing without Related Content.

Response:

Check the ACLs of all your Content List and Assembly Applications in the Rhythmyx Workbench (the ACL is the Key icon on the left side column of the application workspace).

- Default should have full Run-Time access;
- The Admin Role (and any Rhythmyx Developer Roles) should have full Run-Time and Design access;
- Anonymous should not have any access.

Allowing Anonymous access will create an Anonymous session. When the application makes a request to another Rhythmyx application, the Anonymous session will be passed too. If that application does not allow Anonymous access, the request will fail. All Rhythmyx system applications do not allow Anonymous access, and few situations exist when Anonymous access is appropriate.

Error Logs in the Publisher

Different parts of the Publisher have different error logging. The part of the system that encounters a failure determines which log records the error.

System Component	Definition	Where system component logs Publishing errors
Content Lists	The content lists are ordinary Rhythmyx Query applications.	Content lists write serious errors to the PSLOGDAT table (if it is enabled), and they may generate trace files
PubHandler	The PubHandler is a loadable handler that manages the Publishing process. It reads the content lists and generates SOAP messages to the Publisher.	Generally, Loadable Handler errors go to the Rhythmyx server console.
Tomcat server	The default Publisher server.	The Tomcat server has logs. These are generally in the <code><Rhythmyxroot>/AppServer/logs</code> (or <code>/tomcat/logs</code>) directory. There are several different logs, including access logs. None of the Rhythmyx Publisher's output goes to these logs. You can change these logs in the <code>server.xml</code> in the <code>tomcat/conf</code> directory.
Publisher	--	The publisher's output goes directly to the <code><Rhythmyxroot>/AppServer/webapps/publogs</code> directory. There is one log for each edition, and the edition number (and Publishing run number) are in the file name. These are the logs that are linked from the Publication Details Map. If anything fails (or is configured incorrectly) before the SOAP servlet calls the Publisher, there will be no log.
RXPUBSTATUS	Backend Rhythmyx table	If all of the Publishing steps work, Rhythmyx adds a row to the RXPUBSTATUS table. When this row is added, a new line appears in the Publication Log window.

Troubleshooting Content Lists

Error:

`contentlist/@context` is not populating properly.

Response:

The Publisher passes this value to the Content List when it requests it as the HTTP parameter `sys_context`. Make sure that you have mapped this node to a *Single HTML Parameter* with a value of `sys_context`.

CHAPTER 10

Publishing FAQs

Q) How do I know if the publisher is running correctly?

A) In your browser, enter: `http://pubserver:port`. The publisher should display a splash screen if it is running correctly. *Note: The default port of the Publisher is 9980.*

Q) To which Web servers can Rhythmyx publish?

A) Rhythmyx can publish to any Web server that serves pages from a file system and/or a database.

Index

A

Access is Denied Error • 111, 113
 Adding a Publisher Parameter • 24
 Adding Content Lists to an Edition • 70
 Adding Contexts and Location Schemes • 83
 Adding new Keywords • 76
 Adding Publisher Authentication • 25
 Adding Publisher Authentication to the
 Publisher Server • 25
 Adding Scheme Generator Properties • 89
 Appendix
 Troubleshooting • 101, 109

C

Certificate Authority • 34
 Configuring a Publisher • 21
 Configuring Firewalls to Facilitate Publishing •
 23, 26, 27
 Configuring Publishing Components in the User
 Interface • 8
 Content is Publishing without Related Content •
 111, 119
 Content Lists • 47
 Context • 82
 Context Editor • 83
 Copy Site Page • 40, 42, 44
 Copying a Site Registration • 42, 43, 44
 Creating a Content Assembly Support
 Application • 77
 Creating a Content List Application • 48
 Creating a Mirror Edition Content List • 54, 55
 Creating Self-Signed SSL Certificates • 31, 34
 Creating SSL Certificates • 29, 31

D

Debugging Your SSL Publishing
 Implementation • 37
 Default Location Scheme Generators • 85
 Defining a New Edition • 54, 68, 69
 Defining a New Site • 43
 Defining Complex Location Schemes • 86, 88,
 91

Defining Scheme Generators for Contexts • 86
 Deleting a Content List From an Edition • 71
 Deleting a Content List Registration • 59
 Deleting a Publisher Parameter • 25
 Deleting a Publisher's Registration • 20
 Deleting a Site • 40, 43, 44
 Deleting an Edition • 73
 Deleting Contexts and Location Schemes • 84
 Deleting Scheme Generator Properties • 90
 Deleting Scheme Generators in Contexts • 87, 90
 Descriptions of Publisher Parameters • 20, 21,
 22, 24, 26, 29
 Developing and Managing Contexts and Link
 Generation Schemes • 81
 Developing and Managing Link Generation
 Schemes • 85
 Digital Certificate • 34, 37

E

Edit Content List Page • 58
 Edit Edition Properties Page • 68
 Edit Site Properties Page • 40
 Editing a Configuration Parameter • 20, 24
 Editing a Content List Assignment to an Edition
 • 71
 Editing a Content List Registration • 59
 Editing a Publisher's Registration • 20
 Editing a Site • 40, 43, 44
 Editing an Edition • 68, 72
 Editing Contexts and Location Schemes • 84
 Editing Scheme Generator Properties • 90
 Editing Scheme Generators in Contexts • 87
 Edition Content Lists • 70
 Edition Types • 66
 Editions • 65
 Enabling SSL in the Publisher Server • 22, 29
 Error Logs in the Publisher • 110, 120
 Error Opening Socket • 111, 116
 Errors • 110, 111

F

Files Do Not Render Properly in Browser • 111,
 115

H

How the Publishing Process Works • 6

I

Implementing a Custom AuthType • 75
 Implementing a User-Created Plugin • 104

- Implementing Plugins • 103
- Including the Java Class in the Tomcat Server File • 104, 106
- Incorrect links to published files on Web server • 111, 112
- Incremental Content Lists • 48, 61
- Incremental Publishing • 60
- Installing a Remote Rhythmyx Publisher • 13
- Installing and Configuring Publishing System Components • 13

K

- Keystore • 34, 37

L

- Location Scheme Parameters • 88

M

- Maintaining the Authtypes.properties file • 80
- Mapping a Content List Resource • 49, 61
- Mapping to the Plugin in a Content List Resource • 104, 107
- Mirror Editions • 54
- Monitoring Publication of Localized Content • 100

N

- New/Edit Publisher Page • 19

P

- Previewing Content List Output • 71
- Publication Details
 - Publication Maps • 45, 93, 95, 100, 112
- Publication Error Log is Empty • 111, 114
- Publisher Logging • 25
- Publishing an Edition • 67
- Publishing FAQs • 110, 123
- Publishing from Different States • 54
- Publishing in Rhythmyx • 5
- Publishing with SSL • 29
- Purging the Publication Log • 99

R

- Registering a Mirror Site Content List • 54, 56
- Registering a New Content List • 59
- Registering a New Publisher • 20
- Registering a Publisher • 19
- Registering a Publisher Plugin • 104, 105
- Registering Content Lists • 58
- Republishing Failed Content • 100, 101

- Reviewing Publications • 93
- Rhythmyx Publishers • 11
- Rhythmyx Publishing Administrator • 9
- Rhythmyx Publishing Manager • 10
- RXSITEITEMS Is Updating Incorrectly • 111, 118

S

- Setting Up Rhythmyx for Publishing • 7
- Setting Up Scheduled Publishing • 28
- Setting Up SSL Certificates Issued by Certificate Authorities • 34
- Setting Up SSL in the Rhythmyx Server • 31, 34
- Site Maintenance Dialogs • 40
- Site Maintenance Procedures • 43
- Sites • 39
- SSL • 37
- SSL Glossary • 37
- sys_IncrementalContentFilter • 62
- sys_MakeAbsLinkSecureEx • 31, 36

T

- Troubleshooting Content Lists • 110, 118, 121
- Troubleshooting Publishing • 110

U

- Unpublishing a Mirror Edition • 57
- Using a Custom SOAP URL • 25
- Using SSH Tunneling with FTP Publishing • 26

V

- Viewing a Virtual Edition Map • 43, 45
- Viewing Detailed Logs • 97