

Concepts Guide

5.7

Copyright © 1999-2005 Percussion Software.
All rights reserved

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>).

Percussion Software

600 Unicorn Park Drive

Woburn, MA USA 01801

781.438.9900

Internet E-Mail: technical_support@percussion.com

Website: <http://www.percussion.com>

Contents

Document Introduction 5

The Basics of Content Management	6
Content Management in Rhythmyx	7
Rhythmyx and Item-based Content Management.....	8
Using the Concepts Guide	9

Rhythmyx Concepts 11

Content Items	11
Content Types.....	11
Templates	12
Variants	13
Content Editors.....	14
Active Assembly	15
Content Explorer	17
Folders.....	18
Views.....	19
Relationships	19
Publishing.....	20
Workflow.....	21
Roles.....	22
Communities	23
Locales.....	23
Sessions	23

Rhythmyx Logical Architecture and Processing **25**

Content Engine	26
Relationship Engine.....	31
Workflow Engine.....	35
Assembly Engine.....	38
Publishing Engine.....	41

Clients and Interfaces **45**

Workbench.....	46
Server Administrator	49
Multi-Server Manager.....	56
Content Explorer.....	57
Web Services API.....	58

Convera Full-Text Search Engine **59**

Rhythmyx Modules **61**

Connectors	63
Enterprise Content Connector	63
Word Connector	64
BEA WebLogic Portal Connector	65
IBM Websphere Portal Connector.....	66
Oracle Portal Connector	66
Rhythmyx Express Portal	67

Rhythmyx FastForward for Web Content Management **69**

Physical Architecture, Deployment, and Scaling **71**

Rhythmyx Physical Architecture.....	72
Deployment Scenarios	74
Rhythmyx System Components	74
Multi-tiered Environment.....	76
Configuration Options for Development, Testing, and Production Tiers.....	78
Scaling the Rhythmyx Publishing Environment.....	80
Guidelines for Expanding a Rhythmyx System.....	81
Tomcat Web Application Server	82
Database.....	83

Data Protection **85**

Security.....	86
Backup.....	87
Archiving.....	87
Purging	87
Backup Recommendations	87

System Requirements **89**

Server Side	89
Client Side	90
Publisher	90
Rhythmyx Repository	91
Optional Rhythmyx Full Text Search Engine components:	91
Portal Connectors	91

Index **93**

Document Introduction

Rhythmyx is an Enterprise Content Management solution that provides a combination of out-of-the-box capabilities and customization options that enable companies to build high-impact, tailored functionality that evolves as their needs grow. A robust, scalable system, Rhythmyx manages Web and portal content, documents, digital assets, and scanned images. To provide greater efficiency and flexibility, Rhythmyx optimizes content delivery for multiple channels and enables customers to maximize the value of their content by providing content reuse capabilities.

Concepts Guide

This guide provides an overview of Rhythmyx for managers, system administrators, and implementers who are new to the product. It introduces the concepts and architecture of Rhythmyx, and provides guidance for installation and deployment.

After reading this document, you should have a basic understanding of Rhythmyx. Users who require more detailed information after reading this document may find the following Rhythmyx documents useful:

- For complete documentation of Workflows in Rhythmyx, see the document *Maintaining Workflows in Rhythmyx* or see the *CMS Online Help*, accessible from the CMS interface.
- For complete documentation of Publishing in Rhythmyx, see the document *Implementing Publishing in Rhythmyx* or see the *CMS Online Help*, accessible from the CMS interface.
- For complete documentation of System Administration in Rhythmyx, see the *CMS Online Help*, accessible from the CMS interface.
- For complete documentation of Rhythmyx Content Explorer, see the *Content Explorer Online Help*, accessible from Content Explorer.
- For help planning and designing your Rhythmyx system, see the document *Modeling a CMS*.
- For a guide to Rhythmyx's Multi-Server Manager, see the document *Multi-Server Manager*.
- For a guide to Rhythmyx's Enterprise Content Connector, see the document *Enterprise Content Connector*.
- For a guide to Rhythmyx's BEA WebLogic Connector, see the document *Rhythmyx Connector for BEA WebLogic 8.1 Portal*.
- For a guide to Rhythmyx's IBM WebSphere Connector, see the document *Rhythmyx Connector for IBM WebSphere Portal*.
- For information about using WebDAV with Rhythmyx, see the document *Implementing WebDAV in Rhythmyx*.

The Basics of Content Management

Content management refers to a set of applications and capabilities that enable easier maintenance and publishing of items of data, such as documents or Web content. Content Management can refer just to Web content (Web Content Management or WCM) or documents (Document Management or DM), but usually refers to a broader set of content that is published to a variety of locations (Enterprise Content Management or ECM).

Enterprise Content Management (ECM) enables organizations to:

- manage information resources that are created and stored in countless places;
- deliver multiple types of targeted content to specific audiences;
- support easier content reuse throughout the content lifecycle.

An ECM system offers a strategic framework for storing enterprise content and communicating that content through multiple channels to multiple audiences. After being published, content may be delivered to customers through various types of applications including e-commerce, sales automation, customer relationship management, supply chain management, or employee portals.

Content Management in Rhythmyx

The Rhythmyx Content Management System manages content through the automation of content creation, maintenance, and delivery, enhanced with features such as content reuse, delivery to multiple sites, intelligent relationships between content, and simple content assembly for business users. One of Rhythmyx's key features is content reuse, which it achieves by separating content and formatting, enabling users to create a variety of outputs using the same data. Business users can assemble the same Content Items multiple times using different formats and include the Content Items in the output formats of other content. This enables Rhythmyx output pages to contain many Content Items assembled together.

Another key feature of Rhythmyx is its ability to publish content to multiple sites. Rhythmyx allows any number of applications to be used to present published content to consumers since the content delivery mechanism (a portal, Web server, or other application) is not connected or “coupled” to Rhythmyx. This “decoupled delivery” not only offers companies flexibility in choosing delivery applications, but also simplifies the process of delivering content to multiple media.

Rhythmyx provides all of its users with the interfaces and mechanisms they need to accomplish their tasks:

- Content contributors (business users) perform their work through the Content tab of Content Explorer, which provides a graphical user interface for creating content and assembling it into outputs. In addition, content contributors can author content through Microsoft Word or any other application that produces files in formats uploadable to Rhythmyx.
- Implementers and CMS Administrators use the functionality provided on Content Explorer's Workflow, Publishing, and System tabs to configure Rhythmyx's components to meet their companies' needs, and use Rhythmyx's Server Administrator to maintain the Rhythmyx Server, security, and user settings.
- Designers customize their companies' applications and interfaces through the Rhythmyx Workbench and use Rhythmyx's Multi-Server Manager to move components between development, testing, and production Rhythmyx Servers.

Rhythmyx and Item-based Content Management

Rhythmyx provides item-based content management, breaking all content into granular pieces that can be assembled, used and reused in many ways. With item-based content management, users can contribute, manage, preview and rearrange any type of content, regardless of how it will be assembled and used in the future.

Rhythmyx output pages can contain many Content Items assembled together. For example, a job posting may contain an image, a job description, a benefits description, and a legal disclaimer. Rhythmyx treats each of these as a basic item. Page-based systems force users to treat this content as a single entity, causing them to reenter the same information into their system on each page that displays it. Rhythmyx allows content to be broken into its basic Content Items, which are then managed individually. Therefore, if the legal department needs to change the disclaimer on the bottom of all job postings, the change need only be applied once to the source item. The change is then automatically reflected across all job postings and anywhere else the disclaimer is used.

Using the Concepts Guide

Different users of Rhythmyx may find particular sections of this document especially useful:

- If you are going to implement Rhythmyx or you are a Web master who will administer it, the following sections will be particularly useful to you:

Rhythmyx Logical Architecture and Processing (on page [25](#))

Clients and Interfaces (on page [39](#))

Convera Full-text Search Engine (on page [39](#))

Rhythmyx Express Portal (on page [39](#))

Connectors (on page [39](#))

Rhythmyx FastForward for Web Content Management (on page [39](#))

- If you will be managing a Rhythmyx implementation, the following sections include information that you will want to know:

Rhythmyx Concepts (on page [11](#))

Rhythmyx Logical Architecture and Processing (on page [25](#))

Convera Full-text Search Engine (on page [39](#))

Rhythmyx Express Portal (on page [39](#))

Connectors (on page [39](#))

Rhythmyx Modules (on page [39](#))

Rhythmyx FastForward for Web Content Management (on page [39](#))

- If you are a system administrator who is adding Rhythmyx to your network and software infrastructure, refer to the following sections:

Clients and Interfaces (on page [39](#))

Physical Architecture, Deployment and Scaling (see "[Physical Architecture, Deployment, and Scaling](#)" on page [39](#))

Data Protection (on page [39](#))

System Requirements (on page [39](#))

CHAPTER 2

Rhythmyx Concepts

To help you understand Rhythmyx, this section lists and defines some of the system's major concepts.

Content Items

Content Items are the basic units of content in Rhythmyx. A Content Item may be a page or a portion of a page, or a chunk of data stored in a database. By defining output in terms of Content Items and related collections of Content Items, Rhythmyx allows users to modify only the parts of a page or other output that change and to reformat individual Content Items for multiple uses.

Content Items treat both traditional body content and metadata such as content titles and start dates as body content. Any type of data in the Content Item can be included in the Content Item output.

Rhythmyx stores the Content Item data in database tables and formats it prior to publishing it to a *Site* (see "[Sites](#)" on page 20). Each Content Item can be displayed in any number of formats and reused in multiple locations throughout a Site. Content Items can also include links to other Content Items which can be included in their outputs.

For more information, see *Content Engine* (on page 26).

Content Types

In Rhythmyx, a Content Type defines the fields and structure for storing information contained in a Content Item. The Content Type definition also controls the behavior of the *Content Editor (Web form)* (see "[Content Editors](#)" on page 14) used to create and modify the Content Items of that type. The implementer of each Rhythmyx system defines a set of Content Types that store the different types of information displayed on the organization's Web Site(s). For example, one Content Type might store uploaded images while another could store promotional text. In addition to body content fields, each Content Type also includes fields for metadata associated with the main content.

For more information, see *Content Engine* (on page 26).

Templates

In Rhythmyx, as with other content management systems, templates are used to define the possible layout and look and feel of content outputs apart from the content itself. Rhythmyx templates are typically designed in HTML and then automatically transformed into XSLT stylesheets for use in Content Assembler applications. However, templates behave fundamentally differently in Rhythmyx than in other Content Management Systems. In Rhythmyx, business users select and combine fragments of formatted content together to produce output pages and documents. Each template thus defines only a portion of the possible output that could be generated by the system when combined with others. To control the interaction of each template with all others in the system, Rhythmyx uses *Variants* (on page 13) which both contain the template (stylesheet) itself as well as the rules for how that portion of output may be combined with others to produce the final output.

Variants

A Variant defines how to produce the formatted output of a Content Item. The Variant defines transformation and formatting rules that are applied to a Content Item as well as the rules for how this Variant interacts with other Variants when aggregating Content Items together. Each Variant is associated with a specific Content Type, although each Content Type may have any number of associated Variants. In Rhythmyx, Variants are built inside Content Assembler applications. These applications use stylesheet templates as well as other instructions to produce the outputs that Variants define when Content Items are previewed or published.

The following graphics show the same Content Item formatted as two different Variant outputs:

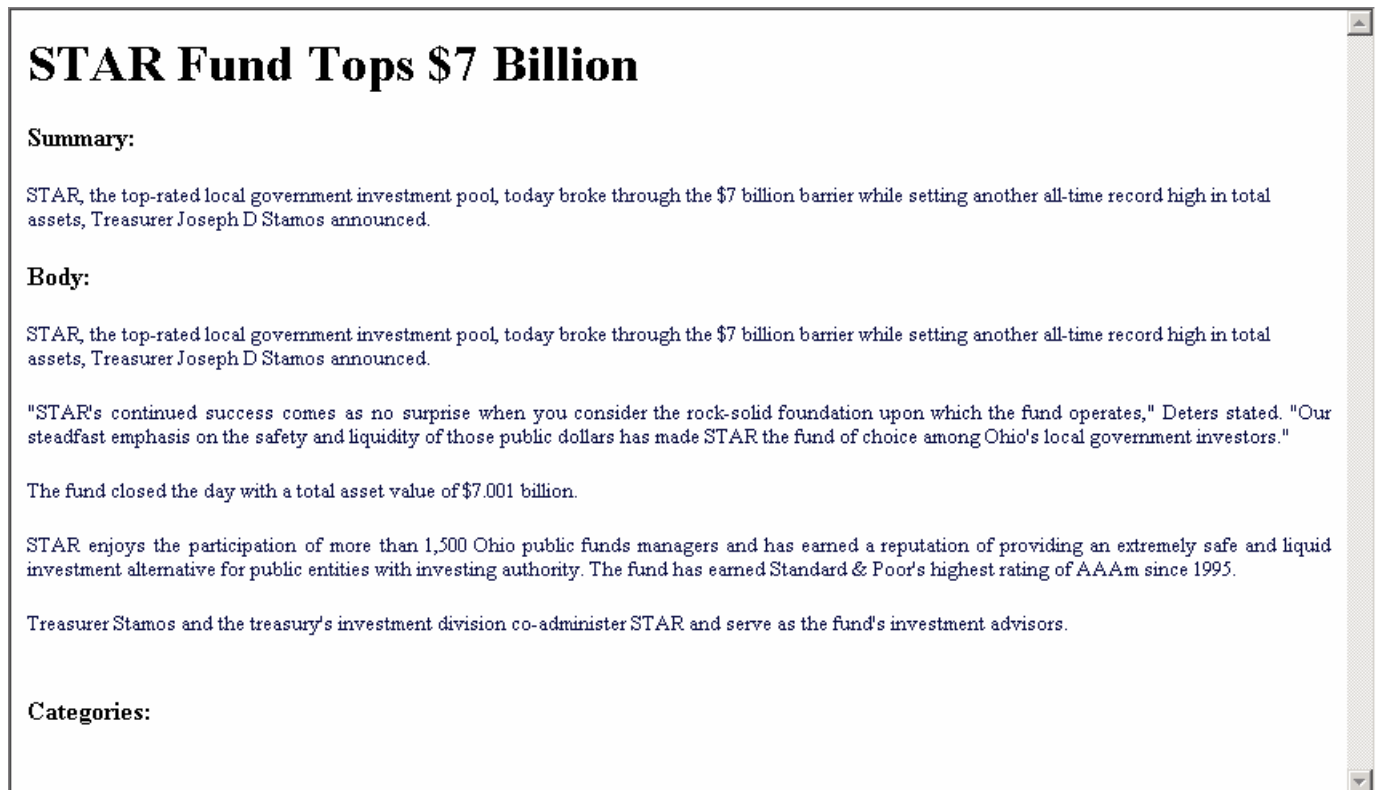


Figure 1: A Content Item formatted by a Page Variant

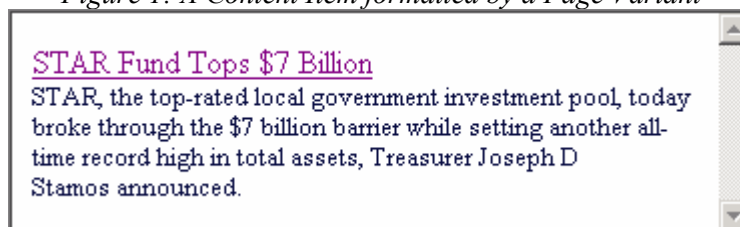


Figure 2: The same Content Item formatted by a Snippet Variant

For more information, see *Assembly Engine* (on page 38).

Content Editors

A Rhythmyx Content Editor is a form for displaying and editing Content Items of a specific Content Type. Content Editors list and display the editable fields of a Content Item.

The screenshot shows a web browser window titled "Rhythmyx - Before planning your estate, plan to see a lawyer - Edit Content - Microsoft Internet Explorer". The page content is titled "Content Properties" and contains the following form fields:

- * System Title:** Before planning your estate, plan to see a lawyer
- * Title:** Before planning your estate, plan to see a lawyer
- * Start Date:** 2004-08-15 00:00:00.C
- Expiration Date:** (empty)
- Reminder Date:** (empty)
- Keywords:** (empty)
- Description:** Why you need a lawyer to help with estate planning

Figure 3: A Content Editor displaying a Content Item

The screenshot shows a web browser window titled "Rhythmyx - Before planning your estate, plan to see a lawyer - Properties - Microsoft Internet Explorer". The page content is titled "Content Properties" and displays a table with the following data:

Content Title (ID)	Creator	Created On	Last Modifier	Last Modified On
Before planning your estate, plan to see a lawyer (338)	admin1	Aug 18, 2004 - 12:00	admin1	Aug 18, 2004 - 12:00
State(ID)	Public	Checked Out	Assignees(Type)	
Draft(1)		by me	<ul style="list-style-type: none"> Admin Author Editor Web_Admin 	
Community		Workflow	Locale	
Internet Community (1002)		Standard Workflow (5)	US English	

Figure 4: A Content Editor's Properties page

Each Content Editor is created in a Content Editor application. Rhythmyx provides a developer interface for designing and customizing the functions of Content Editors.

For more information, see *Content Engine* (on page 26).

Active Assembly

Active Assembly enables business users to assemble collections of related content items together into the formatted pages, documents and other output they wish the CMS to generate. Active Assembly for Pages provides a WYSIWYG browser UI that lets business users assemble content visually into slots. Active Assembly for Documents provides a split pane "tree" view that lets users assemble content hierarchically. In either case, users of Active Assembly perform searches against content in the system, then select the Content Items they wish to combine. They also select the Variant of each Content Item to control how that item will appear in the final assembled output.

When a user opens a Content Item into Active Assembly, and adds another related Content Item into one of its Slots, the two items become linked by an Active Assembly Relationship. For more information, see *Relationships* (on page 19).

When a Content Item is previewed or published in one of its output formats, Rhythmyx Content Assembler applications automatically execute the Active Assembly Relationships in a process referred to as recursive rollup. The most deeply embedded related Content Items are formatted, then the Content Items that include them are formatted, then the Content Items that include these Content Items are formatted, and so on until the page Content Item that includes and displays all the others is formatted. The following diagram illustrates this process:

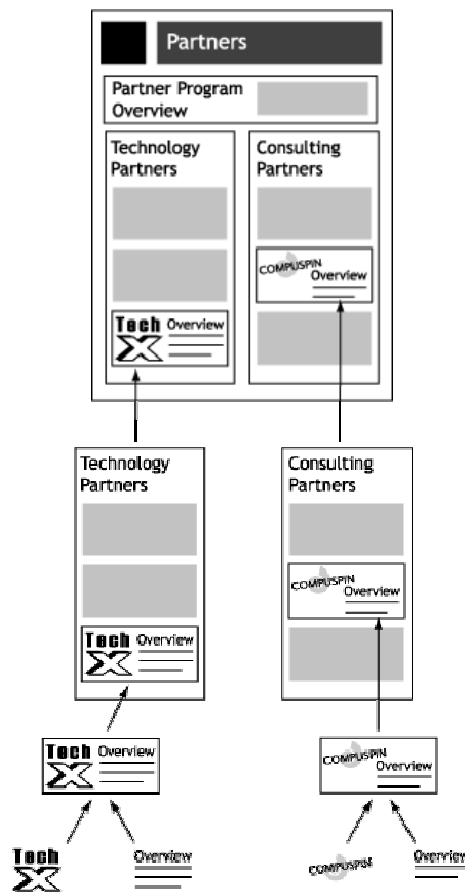


Figure 5: Recursive Rollup

NOTE: Active Assembly Relationships are predefined in Rhythmyx, but implementers can modify their specific behavior. See *Relationship Engine* (on page [31](#)).

Content Explorer

Content Explorer gives Web masters, CMS managers, and business users direct access to Content Items in the CMS including functions such as create, approve, and copy, as well as mechanisms to assist in organizing Content Items such as Folders, Views and Searches.

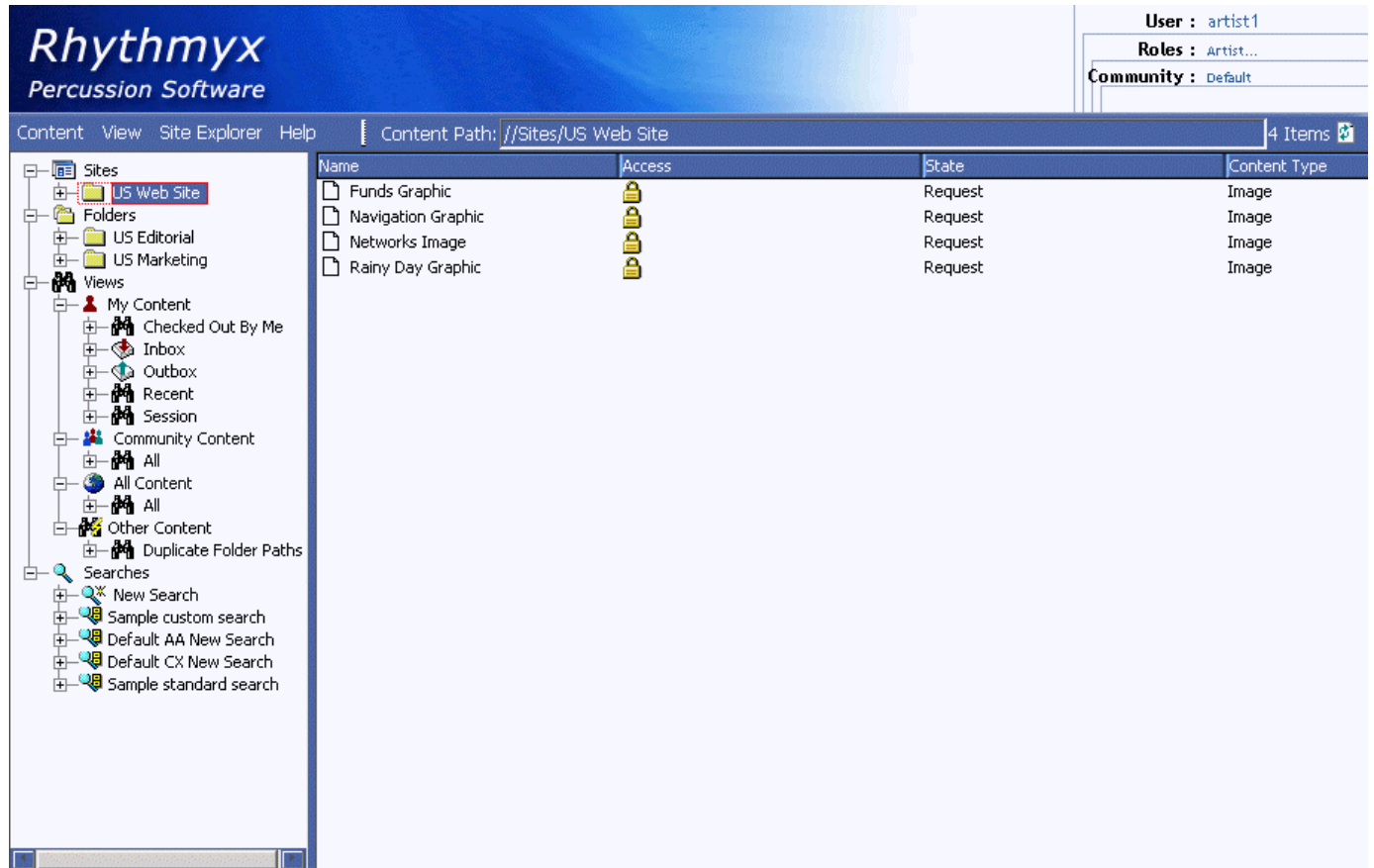


Figure 6: Content Explorer as it appears to a business user

Content Explorer includes a Navigation Tree with Folders, Views, and Searches that present Content Items by user-defined or system-defined categories. When users click on one of these nodes in the Navigation Tree, the Display pane to the right shows its contents.

Users can right-click on a Content Item to access an Action Menu of functions to perform on the Content Item, such as editing and performing Workflow transitions.

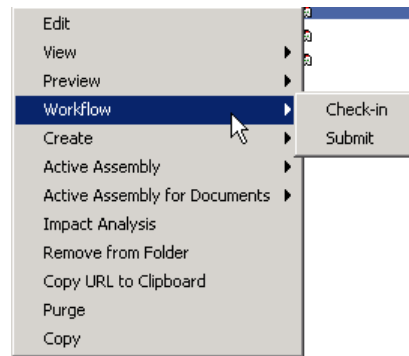


Figure 7: Action Menu

Folders

Rhythmyx Folders in the Content Explorer navigation pane are similar visually to Folders in Windows Explorer. While Windows Explorer folders organize files, Content Explorer Folders organize Content Items. However, underneath, Rhythmyx Folders are very different. In Rhythmyx, Folders are virtual locations for maintaining Content Items. A Content Item is not actually stored in a folder. All Content Items are stored in the CMS Repository (a DBMS). A Content Item "in" a folder is actually linked to the Folder. Folders provide a method for users to arrange Content Items regardless of how they are stored. The same Content Item may appear in multiple Folders at the same time, since what the Folder holds is actually a link to the physical Content Item. Once an Item is altered, it is altered in the CMS as a whole, and will appear changed in all the Folders in which it appears.

The Navigation pane in Content Explorer contains two main Folder nodes, the Sites node and the Folders node. Folders in the Sites node are used to control the structure of Folders on a published Site. Administrators can use Site Folder Publishing to publish the contents of Site Folders to the same folder tree on the Site. Folders in the Folders node can be arranged in any way that is useful to the user; for example Folders could represent projects, or workgroups, or hold all content of a given Content Type.

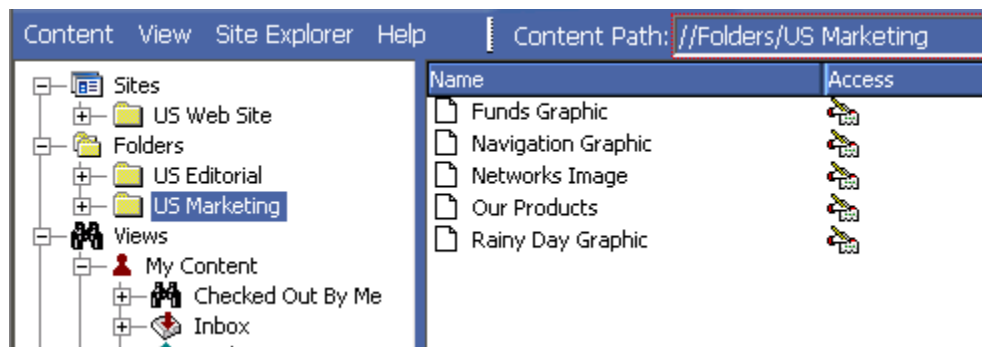


Figure 8: Folders in Content Explorer

Views

Rhythmyx Views in the Content Explorer navigation pane display content sorted by parameters selected by Rhythmyx or a system implementer. Views are useful because they arrange content by categories that would be difficult to maintain in Folders, such as Workflow States and assigned users, and they allow users to see Content Items that are not stored in their folders. Implementers set up the views available to users; users can choose to display different views, but cannot modify any view.

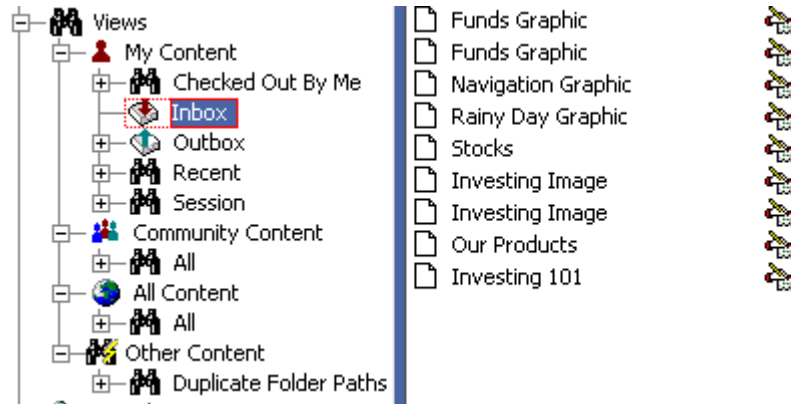


Figure 9: Content Explorer Views

Relationships

A Rhythmyx Relationship is an association between two objects in Rhythmyx (usually between Content Items). One object is the owner and the other is the dependent. For example, in an Active Assembly Relationship, an original Content Item is an owner, and a Content Item that is included in its output is a dependent; in a Translation Relationship, an original Content Item is an owner, and the same Content Item translated into another language is a dependent. A set of properties define the behavior for each type of Relationship in Rhythmyx. For example, in one type of Relationship, the owner and dependent must both be in a public State before either of them can be published; in another type of Relationship, when the dependent is published, the owner is transitioned to an archive State.

Implementers maintain rules governing relationship behaviors in the Rhythmyx Workbench Relationships Editor. For more information, see *Relationship Engine* (on page 31).

Publishing

Enterprise applications that make content available to end users are content delivery applications. These include corporate internet sites, employee intranets, e-commerce applications, and print production systems. Typically, content delivery applications are hosted in a wide variety of application platforms such as enterprise portal systems, Web Servers, personalization servers, or application servers. With an ECM system in place, content applications must interact with the ECM to get the correct content. This interaction may be defined as either "coupled delivery" or "de-coupled delivery." With coupled delivery, each content delivery application must directly request all content from the CMS via an API. The CMS becomes the direct content object store under all the applications. With de-coupled delivery, each content delivery application defines its own private content object store. The CMS is configured to transfer content from its Repository into the different content object stores defined under each of the content delivery applications. Whenever content in the CMS changes, the CMS publishes the appropriate changes into all the content delivery applications. This lets the delivery platform deliver only the appropriate managed content without an active connection to the CMS.

Rhythmyx uses de-coupled delivery. Rhythmyx uses a Publisher to transfer content to each delivery platform's repository or file system, or through platform specific APIs via Publisher plug-ins. Rhythmyx's decoupled delivery significantly simplifies the use of multiple delivery platforms, because each platform does not have to interface with the CMS, nor do all applications have to agree upon a common definition or schema for storing and delivering content objects.

Sites

A Site defines a location where Rhythmyx Content Items are published. A Site may be a file system, an ftp location, or a database repository. Sites can also refer to multiple virtual locations on the same physical machine (such as a virtual Web folder or drive).

Site Folders in the Content Explorer navigation pane can dictate the structure of a Site. Administrators can use Site Folder Publishing to publish the contents of a Site Folder to the same folder tree on the Site.

Editions

The Rhythmyx publishing process uses Editions to let administrators set up and manage the types of publishing activities that can occur. Editions primarily define the content that is to be published and the Site where it is to be transferred. Publishing of an Edition begins via a scheduler program, a workflow or system action, or an end user manually initiating an Edition from a button in the browser interface.

An Edition specifies the publishing Site and includes one or more Content Lists to define the content to be published. Each Content List specifies a given subset of Content Items to publish, including the sequence in which to publish them and the directory locations at which to publish them. For example, an Edition specifies that content is published to an Internet Site. It includes three Content Lists. Each Content List represents a different Page Variant, and specifies that all Content Items in a public State that are associated with that Variant be published to the Site. For Site Folder Publishing, it is more common to use a single Content List that specifies that all Content Items under a common Site Folder root be published to the Site specified in the Edition.

Workflow

A Rhythmyx Workflow is a set of States that a Content Item progresses through during its lifetime. Workflows often include the following States:

- **development** - In a development State, users create or modify a Content Item. It may be the initial State of a Content Item, a State in which a contributor regularly adds data, such as artwork, to an existing Content Item, or a State to which an approver returns a Content Item for modification.
- **approval** - In an approval State, administrators or Web Masters approve an existing Content Item for publication or reject it and return it to an earlier State for modification.
- **public State** - Content Items in a public State are ready to be published. During the next publishing run, these Content Items can be selected and published to a delivery Site. Depending on the Workflow assigned to them, Content Items in a public State may have moved through one or more approval States before reaching the public State or may have transitioned directly to the public State after being created.
- **archive State** - Content Items in an archive State are no longer publishable. Most Content Items in a public State reach a point at which they are no longer current; at this point, a user or automatic Transition moves them into an archive State. If a Content Item becomes unusable before it reaches a public State, it may also be transitioned to an archive State.

Each Workflow includes Roles that define user access to content in each State, and Transitions or actions that specify how content can move from one State to another.

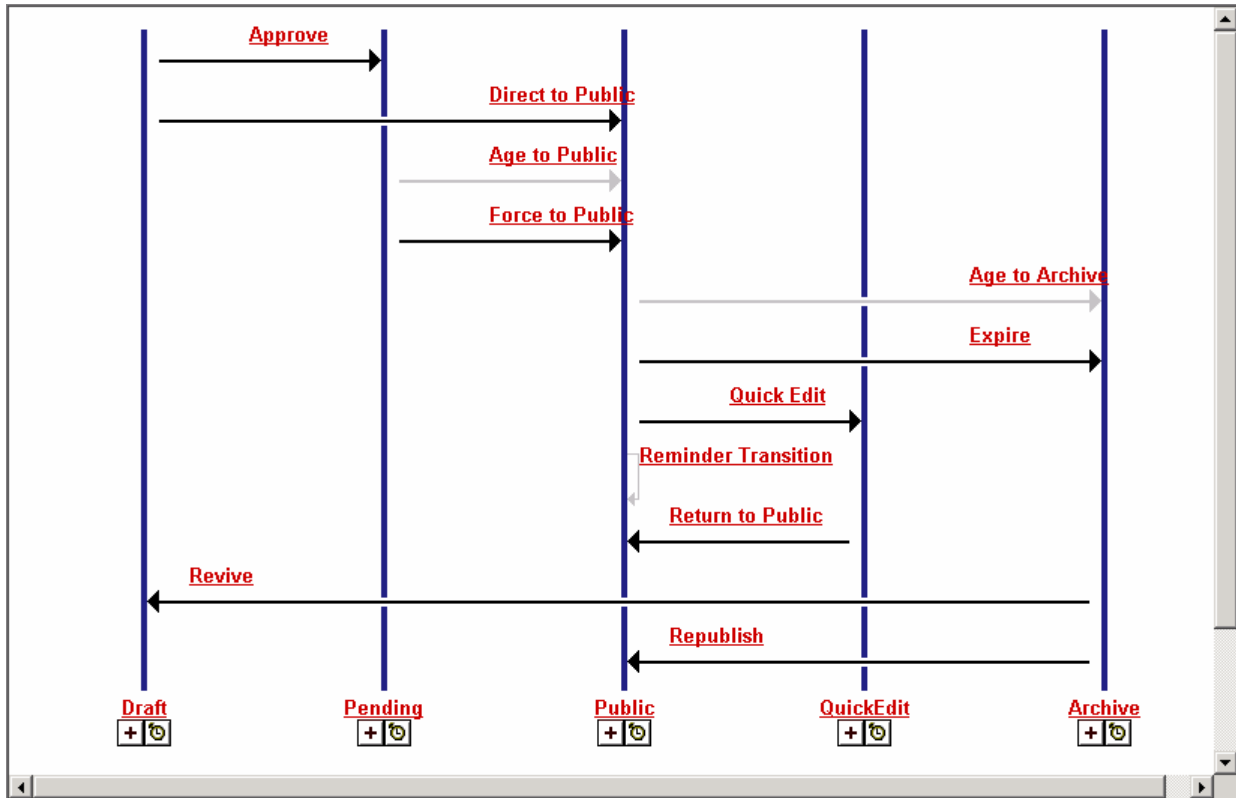


Figure 10: Workflow diagram. Each vertical line represents a State. Each horizontal arrow represents a Transition from one State to another. Transitions may be initiated by users, or by automated events, such as expiration based on date.

See *Workflow Engine* (on page 35) for more information.

Roles

A Role defines a collection of users associated with the same permissions and access in Rhythmyx. In Workflows, Roles are assigned permissions by State to determine which users can access content in each State and which Transitions they can initiate. Roles also determine which portions of Content Explorer a user can view, Folder access, and other privileges in Rhythmyx. Roles are also used to define the members of a *Community* (see "*Communities*" on page 23).

Organizing users into Roles helps administrators manage users that have the same permissions. Instead of managing the permissions for each user, they define a Role and the permissions for it, and then assign users to it. Users assigned to that Role have the permissions specified for that Role. When an administrator assigns a Role to the Access Control List of an application, the users in that Role have access to that application.

Communities

Communities add efficiency to a Rhythmyx System by filtering the Content Editors, Variants, Workflows, Sites, and user interface components available to users. A Community represents a group of Roles that require access to similar information in Rhythmyx. A Role can be associated with more than one Community, and users belong to all of the Communities associated with their Roles. When users log in to the CMS interface, Rhythmyx only displays Rhythmyx components associated with their Communities.

Locales

A Rhythmyx Locale is a user's login language including its regional variation, such as French Canadian or British English. Rhythmyx uses Locales in two distinct areas. First, the user interface Locale (chosen by the user during login) affects the language of labels seen in Content Explorer during the session. Second, each Content Item in the system is marked with a specific Locale. By default, the Locale of newly created items is set to that of the user interface.

Locales enable customers to deliver similar content to audiences that speak different languages. If more than one Locale is available, the system is globalized, and it is possible for users to make Translation Copies of Content Items. For example, a customer may require two different sites for the Canadian market—one site in Canadian English and one site in Canadian French. Content contributors could begin by creating the site in Canadian English, and then translators could create duplicate Canadian French pages by creating Translation Content Items of each English page.

For information about Localizing your Rhythmyx System, see the document *Internationalizing and Localizing Rhythmyx*.

Sessions

A user begins a session by logging in to Rhythmyx. When the user logs out (or the system logs the user out) the session is finished. After logging out, a user cannot return to a previous session, but must log in again and begin a new session.

Each session takes the login username, and if enabled, the default or login Community and Locale as user session properties. Rhythmyx displays these user session properties and the login user's Role at the top of Content Explorer.

The default Locale and Community are configurable. The login Locale only displays at the top of Content Explorer with the other user session properties if the user has more than one Locale. The user can click on Community or Locale to change it for the active session.

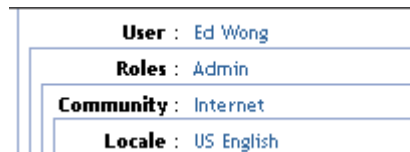


Figure 11: User Session information in Content Explorer

Session properties determine the login user's access to Content Items and Action Menu options and other components of Content Explorer.

CHAPTER 3

Rhythmyx Logical Architecture and Processing

Rhythmyx's components and their processes interact in a variety of ways. However, the majority of Rhythmyx's features and operations are associated with one of five logical engines that make up the CMS:

- Content Engine - Controls the creation and storage of Content Items.
- Relationship Engine - Defines and processes Relationships between Content Items
- Workflow Engine- Defines and controls the life cycles of Content Items.
- Assembly Engine- Controls the formatting of Content Items
- Publishing Engine- Implements the delivery of Content Items.

The topics in this section explain the processing of each engine and the Rhythmyx components that it uses.

Content Engine

The Content Engine includes the components and processes in Rhythmyx that allow users to create and store Content Items. Users can use these components in Rhythmyx to create and modify all of their content or can integrate them with outside applications such as Microsoft Word and Adobe PhotoShop to create content in other applications. Rhythmyx can upload content created in other applications through specially configured folders or one of the Connectors that Rhythmyx provides.

For information about configuring folders to upload content from other applications, see the document *Implementing WebDAV in Rhythmyx*.

Architecture

The main components of the Content Engine are Content Types, Content Editors, Content Items, Content Editor definition files, and the backend repository tables that store local, shared, and system data for Content Items.

A Content Type defines a category of Content Item and the fields included in items of that type. It specifies the Content Editor or form used to create and modify the Content Items of that category. The Content Editor specified by the Content Type defines the body content and metadata fields for the Content Type.

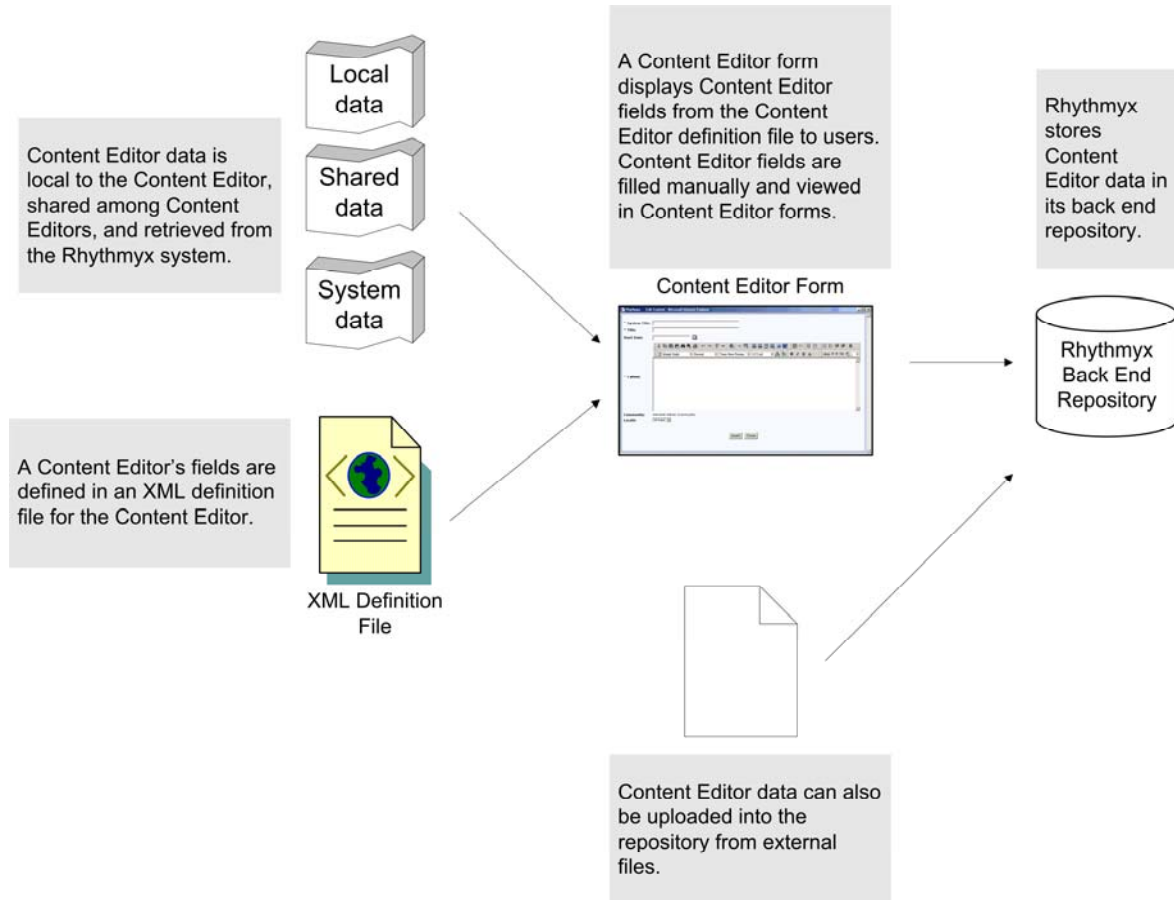
A Content Editor can include local, shared, and system fields which are defined in XML files. Local fields are unique to the Content Type; an implementer defines them. Shared fields are used by more than one Content Editor in the CMS. System fields are used in all Content Editors in the CMS. After the Content Editor and Content Type are created, Rhythmyx creates an XML file that defines all of the Content Editor fields and builds a backend table for storing the values of local fields. The values for shared and system fields are also stored in backend tables.

Local, shared, and system fields may store what is traditionally distinguished as body content or metadata. However, Rhythmyx does not distinguish between body fields and metadata fields. They are defined in the same manner in Content Editor definition files and backend tables, and any body or metadata field can be displayed in a Content Item's output.

Implementers create Content Editors in the Rhythmyx Workbench, either from a generic Content Editor template, a predefined XML file, or a backend table defined for the Content Editor. Once the Workbench creates a resource for the Content Editor, the implementer can access a Properties dialog for modifying it. To perform more detailed customization of the Content Editor, the implementer can directly modify the XML file associated with it.

Users view Content Editors as Web browser forms that list and display the fields and the field contents of a Content Item. Manually entering data into a Content Editor is one way that users can create Content Items in Rhythmyx. After entering the data, the user clicks an Insert button to upload the data into the backend tables that store the Content Item. Users can then reopen the Content Items to view or edit them. Users can also create Content Items in third party applications and then upload them into Rhythmyx as specific Content Types using specially configured (WebDAV-enabled) folders or one of Rhythmyx's Connector applications. After Rhythmyx uploads these Content Items, users can open them in Rhythmyx Content Editors for viewing or editing.

A Content Item is an instance of a Content Type. It includes the body data and metadata for a particular piece of content and is stored as unformatted data in the Rhythmyx database tables representing the Content Type's local, shared, and system fields. When Rhythmyx requests the Content Item, raw data from the database fields are inserted into a virtual XML document for further processing by the CMS. When the processing is done, the contents of the updated virtual XML document are inserted into the database fields for storage.



Processing

The process of creating a Content Item in Rhythmyx can either involve a content contributor making a request to create the Content Item directly in Rhythmyx, or Rhythmyx ingesting a file or XML created in a third-party application and submitted through specially configured folders (WebDAV), a Connector, or through custom Web services API submission.

When a content contributor makes a request from Content Explorer to create a new Content Item, Rhythmix determines the Content Type requested and generates a Web form for the Content Editor. The content contributor fills in the Web form with values for the Content Editor fields and clicks an insert button. When the insert button is clicked, Rhythmix is directed to create the Content Item using the data entered into the form.

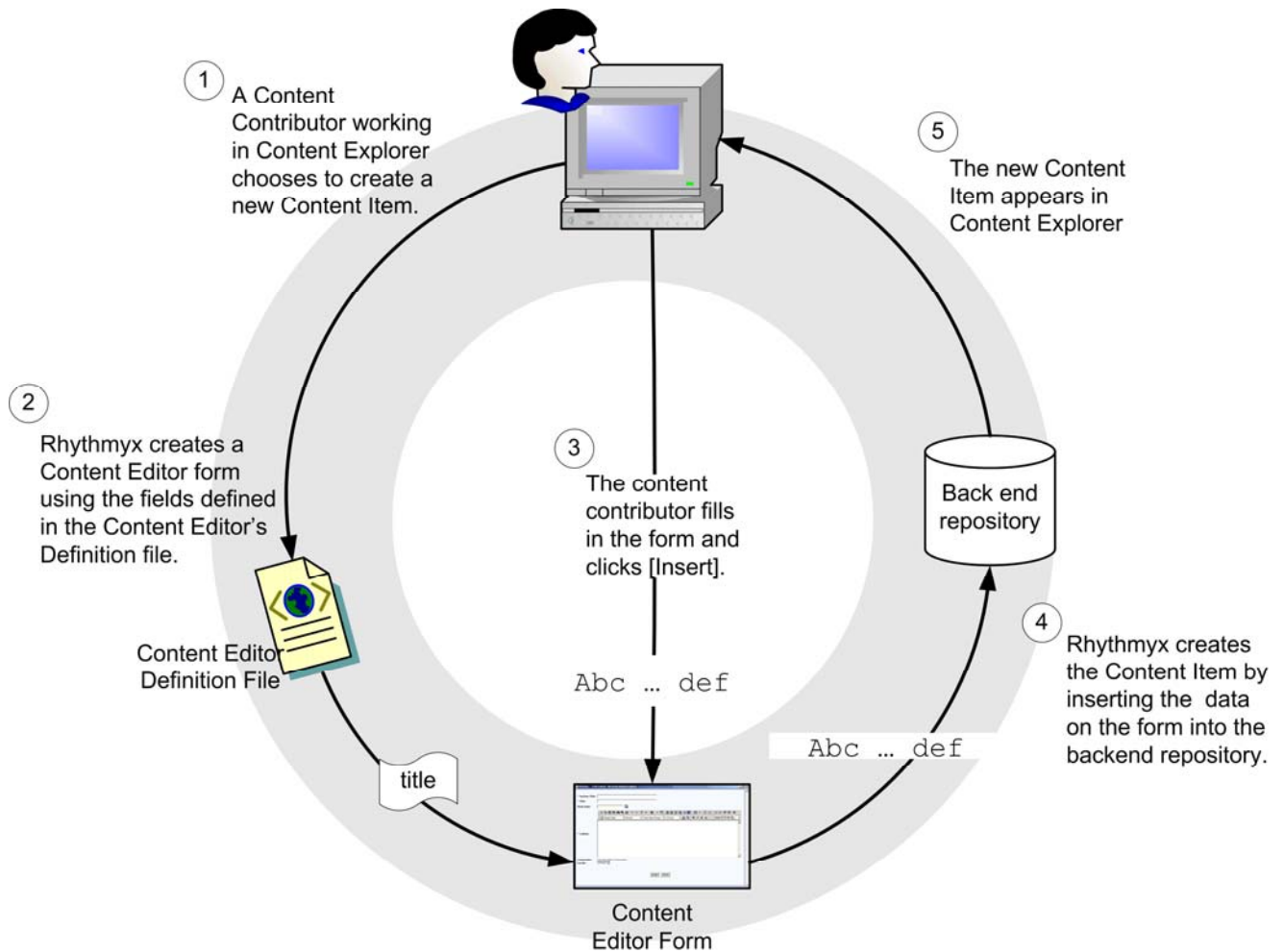


Figure 12: Creating a Content Item in Content Explorer

The process for uploading Content Items into Rhythmyx begins when a user creates a file in a third-party application. If the system uses WebDAV, the user saves the file using a WebDAV-enabled client (such as PhotoShop) or saves the file into a WebDAV-enabled folder in the underlying operating system. Rhythmyx uses mappings from a WebDAV configuration file to associate data from the file with a Content Type and to the fields defined for that Content Type. For information about implementing WebDAV in Rhythmyx, see the document *Implementing WebDAV in Rhythmyx*.

Other Connectors and a Web services API are also available for ingesting content into Rhythmyx. These all submit to the same Content Types using input field definitions and other rules from the same Content Editor definitions as are used by the Web forms and WebDAV submissions.

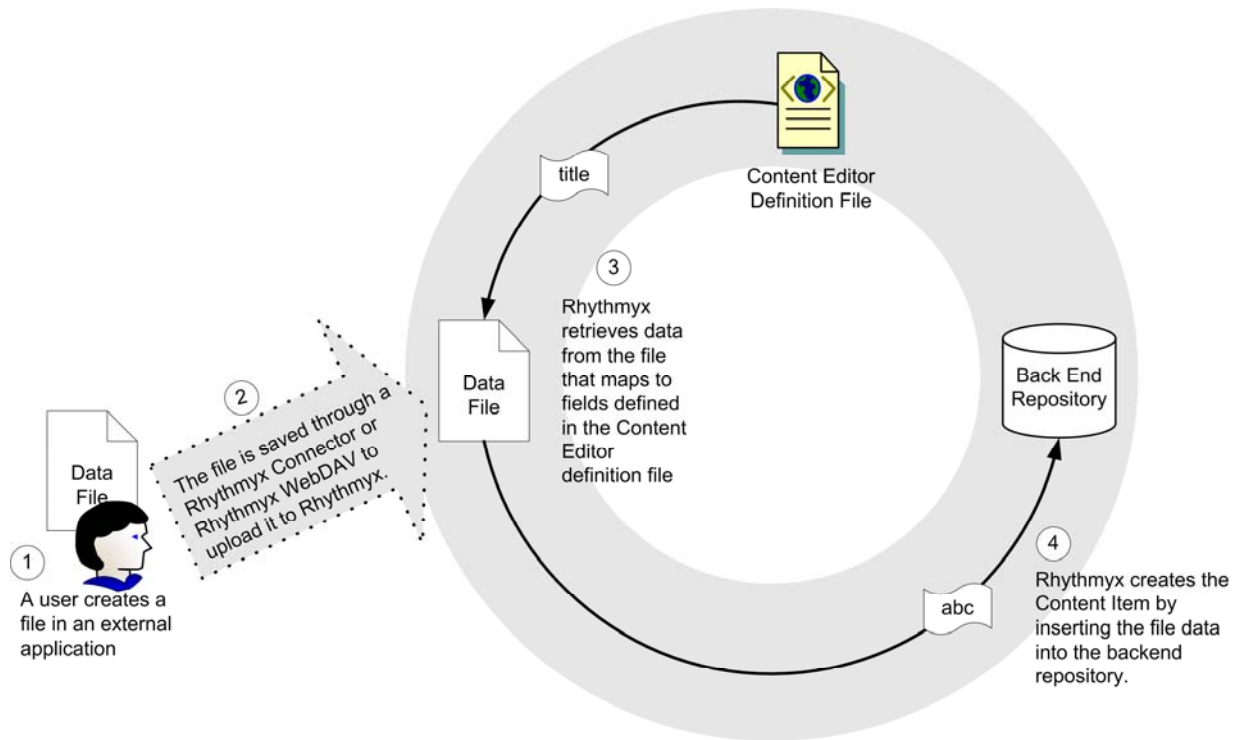


Figure 13: Creating a Content Item in an External Application

Rhythmyx uses the same components to present a Content Item for viewing or editing as it does to create a Content Item. In this process, a content contributor makes a request to view or edit the Content Item. In response to the request, Rhythmyx creates a virtual XML document using definitions from the Content Editor definition file and data from the backend repository tables for the specific Content Item. Rhythmyx applies Content Editor stylesheets that format this XML to present it in the desired form to the content contributor for viewing or editing.

If a Content Item is edited after entering a public State, a new revision of the Content Item is created and becomes the revision displayed as output. The former version of the Content Item is saved with the previous revision number, but in the future, it can be restored as the current revision that is displayed as output.

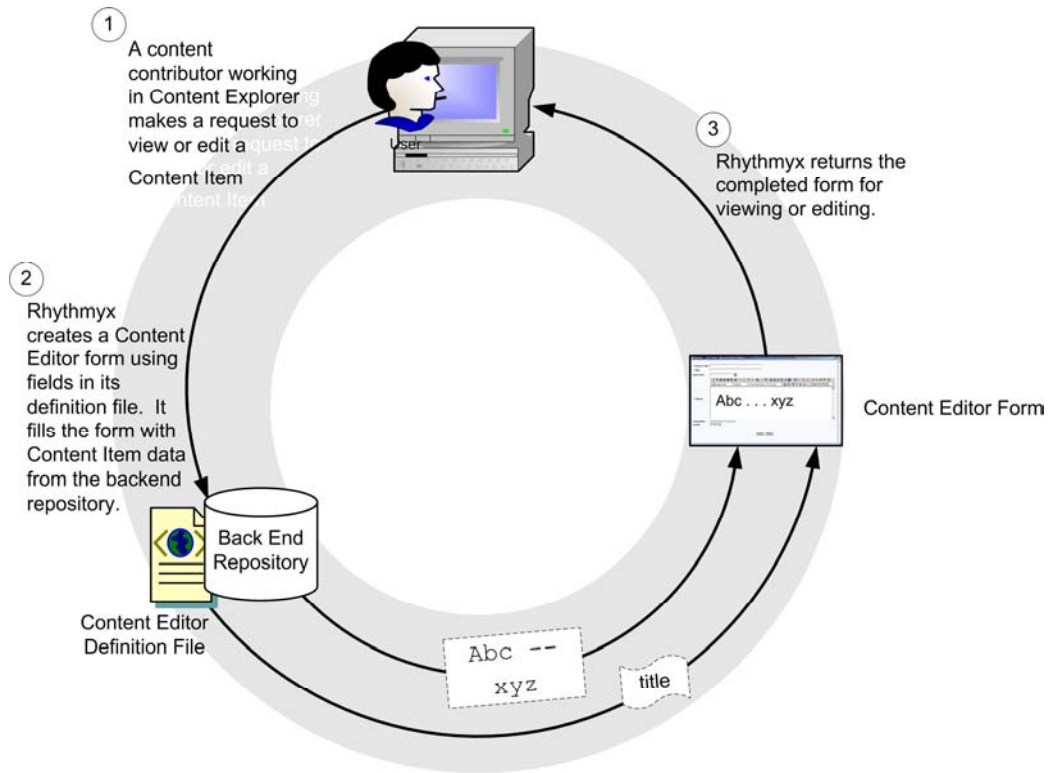


Figure 14: Viewing or Editing a Content Item

Relationship Engine

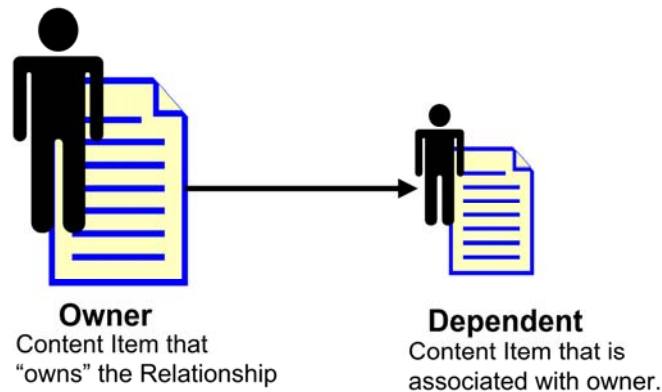
The Relationship Engine ensures that Rhythmyx understands and manages all content associations and enforces all business rules defining those associations. The Relationship Engine includes the interfaces for defining Relationships in Rhythmyx and the processes that implement the proper functioning of these Relationships.

Architecture

A Rhythmyx Relationship is an association between two objects in Rhythmyx (usually two Content Items). One object is the owner and the other is the dependent. The dependent is associated with the owner according to rules or behaviors specified in the Relationship. Four components specify the rules and behavior of a Relationship: properties, cloning options, exits, and effects:

- Properties - Information defining the Relationship such as its Name, Expiration Date, and whether it is used in Active Assembly. Rhythmyx uses this information to determine actions such as when to end the Relationship.
- Cloning Options - Rules defining whether to duplicate the Relationship and how. For example, some Relationships may require that if the Relationship is duplicated, the dependent object must also be duplicated; others may require that just the Relationship be duplicated.
- Exits - Java code and scripts executed when the Relationship is created. For example, an exit may prevent Rhythmyx from creating the Relationship if an instance already exists.
- Effects - Java code and scripts executed when a Relationship is processed. For example, an effect may require that Rhythmyx notify certain users when the Dependent in a Relationship reaches a Public State.

Relationship: Main Components



Other Components

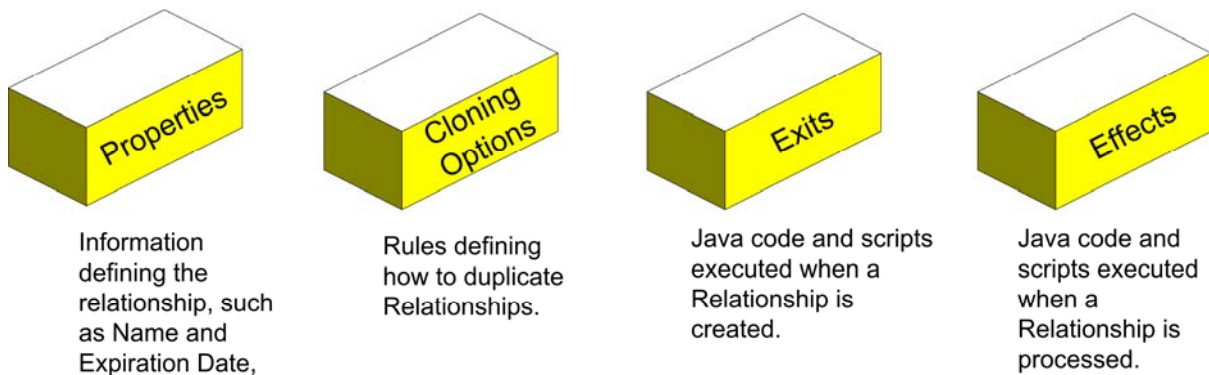


Figure 15: Relationship Architecture

Processing

To begin understanding how Relationships work, let us look at a simple HTML page that consists of some text and a graphic:

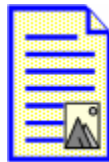


Figure 16: A Page including text and a graphic

In Rhythmyx, this page consists of two Content Items, an Article Content Item that contains the text and an Image Content Item used to manage the Image file. The two Content Items are associated through an Active Assembly Relationship.

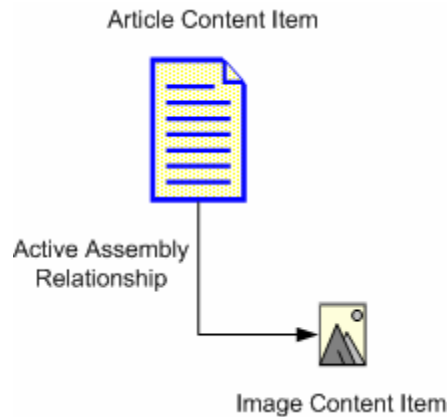


Figure 17: Active Assembly Relationship

The Active Assembly Relationship is a very basic Relationship. It simply points to a dependent Content Item to be inserted into a Slot in a specific Variant of the owner Content Item. At assembly, when the individual Content Items are formatted, an HTML page is formatted with a reference to the image. The browser then renders the HTML as one page.

Now, let us suppose we want to ensure that the Article cannot go Public unless the associated graphic is also ready to go Public. The Active Assembly Relationship does not meet our needs because it does not put any constraints on the two Content Items. Each can go Public independently of the other. If the Article goes public and the image does not, the article will be published, but the Image will be removed. While this ensures non-public content will not be seen, it may make the article less compelling. In some cases, articles might depend on the image to make sense. The Relationship Engine can then be used to manage these cases, where one piece of content is dependent on others. For example, to ensure that the Article cannot go Public unless the associated Image is ready to go Public, we could use the Active Assembly - Mandatory Relationship when linking the Image to the Article.

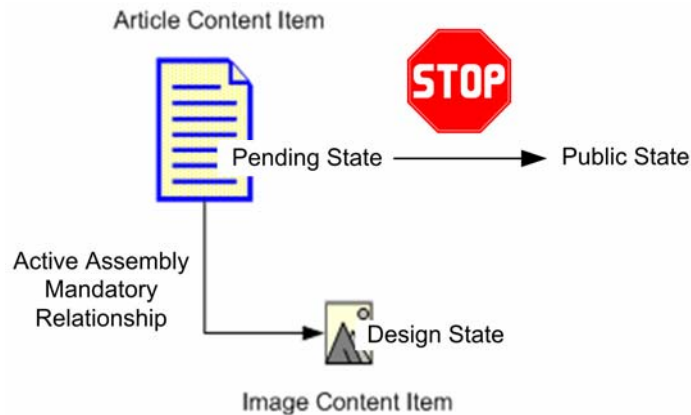
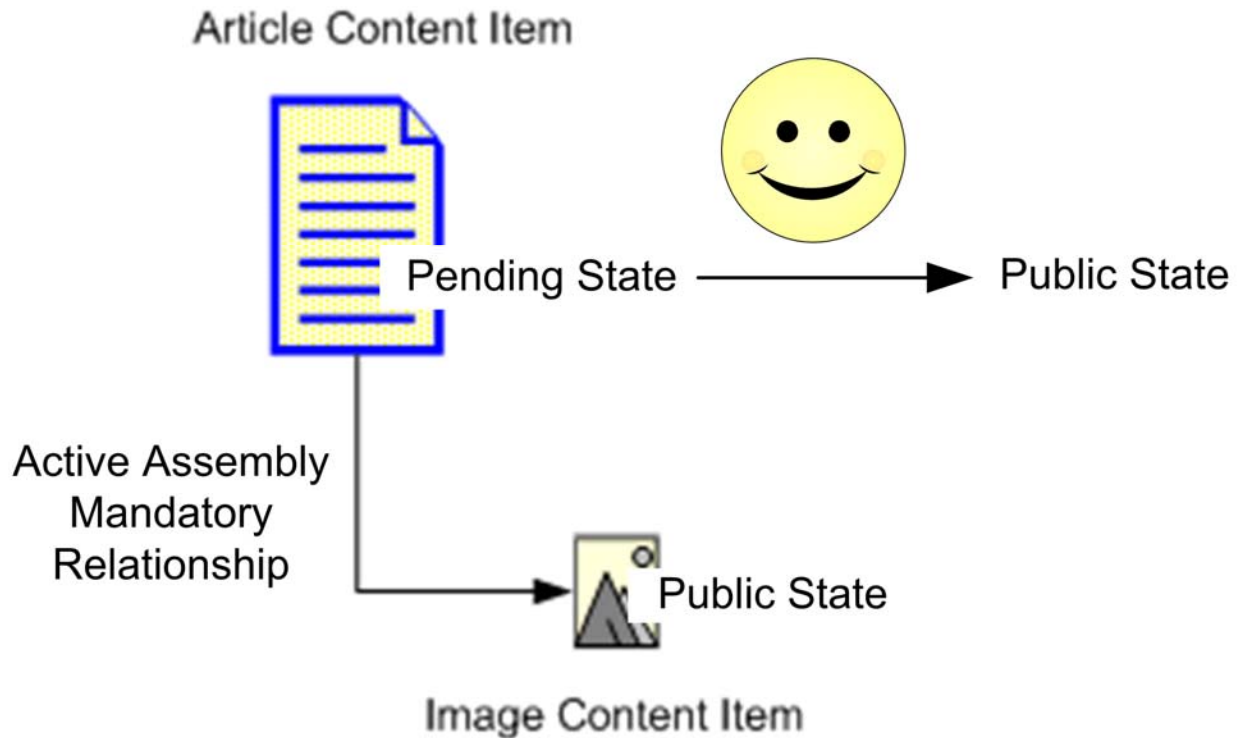


Figure 18: Active Assembly - Mandatory Relationship

The Active Assembly - Mandatory Relationship includes the Effect `sys_PublishMandatory`. This Effect prevents a Content Item from going Public if the associated dependent Content Item in the Relationship is not also Public. In this case, the Effect checks whether the Dependent is Public before allowing the Owner item to go Public. Thus, if the Image Content Item is not Public, we cannot Transition the owning Article to Public. The Article will wait in the Pending State until the Image Content Item has entered the pending State. When this Dependent enters the Pending State, we will be able to Transition the Article to Public. Other relationship Effects could be used to create different types of automation.



Workflow Engine

The Workflow Engine is composed of the components and procedures that define Workflows or business processes in Rhythmyx.

Architecture

Workflows define the stages in the process of creating a Content Item and specify which users can access the items at each stage in the process. The Rhythmyx Publisher looks at the Workflow State to determine whether or not a Content Item is eligible to be published.

Workflows consist of States, Roles, Transitions, Transition Roles, and Notifications.

- States define the different stages of a Content Item.
- While a Content Item is in a certain State, only the people in Roles assigned to that State have access to the item. In Workflows, Roles should define a set of users by the function they perform in the process, such as “Author”, “Editor”, “Designer” and “Administrator”.
- Transitions define routing paths from one state to the next. There can be multiple transitions from one state to other states, and they may go forward, backward, and even "loop back" to the same state. Transitions are triggered manually when an assignee takes some defined "triggering" action on the Content Item, such as pushing the approve button or automatically, when a specified date is reached.
- Transition Roles specify the Roles that must perform a Transition on a Content Item before it is valid, and the number of users who must perform the Transition. For example, the Transition Role could specify that at least one person in the Editor Role and another person from the Marketing Role must perform the Transition before it is completed.
- Notifications or email messages can be assigned to certain Transitions. Rhythmyx can send the Notifications to any email addresses or to those of Roles assigned to the before or after States for the Transition.

Workflow Architecture

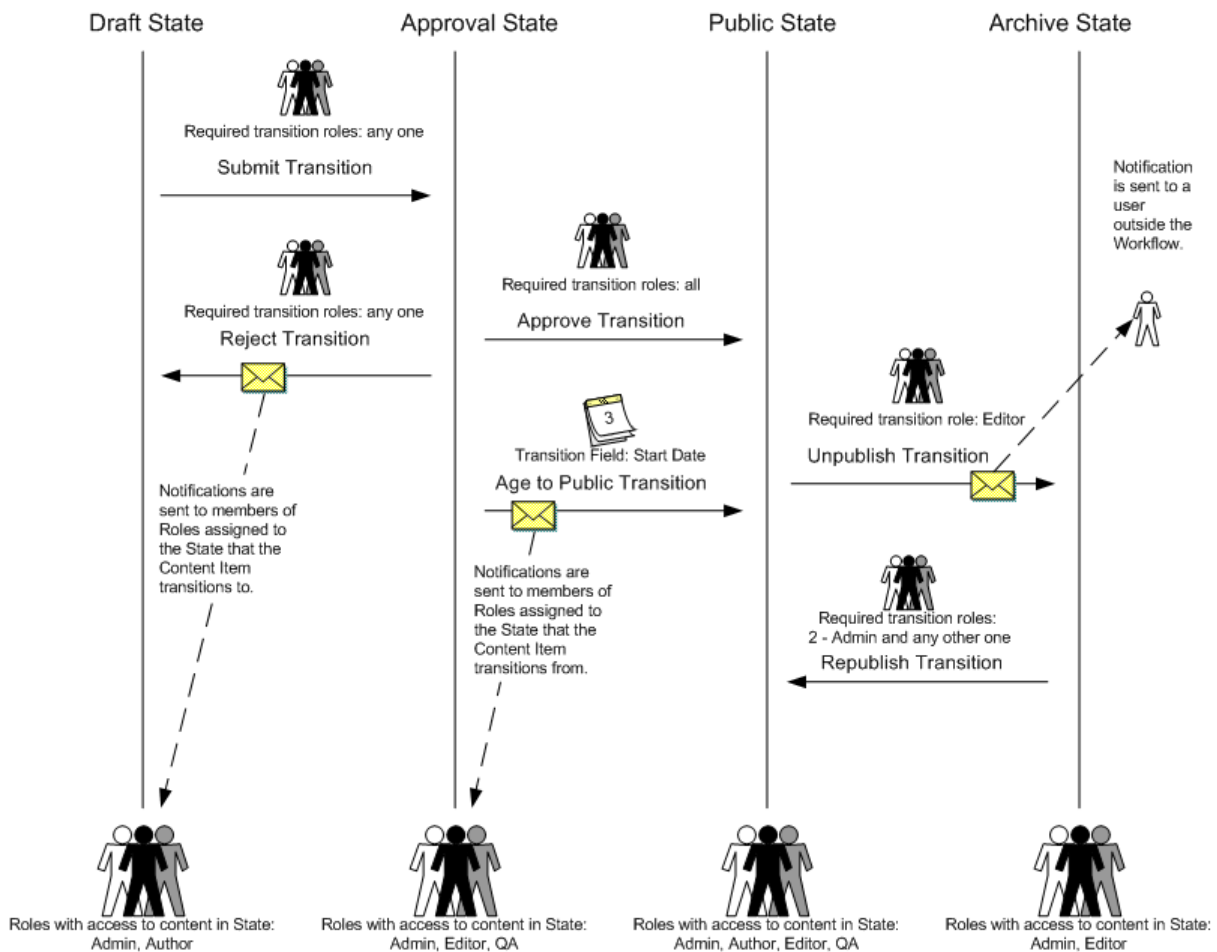


Figure 19: Workflow Engine Architecture

The Workflow architecture diagram, the vertical lines represent the States of this Workflow: Draft, Approval, Public, and Archive. The arrows coming from each State represent its Transitions. They point to the State that the Content Item transitions to. The figures at the bottom of each State represent the Roles that have access to Content Items in the State, and the figures above Transitions represent Transition Roles. Envelopes superimposed on Transition Arrows represent Notifications. Notifications point to the users who receive them when the Transition occurs.

Processing

When a Content Item is created and saved, it automatically enters the first State in its Workflow. Either a Role with access to this State transitions the Content Item to the next State, or an Aging Transition automatically moves it to the next State when a specified date or interval is reached. The Content Item moves through each State in the Workflow in this manner until it reaches a public State. The Rhythmyx Publisher is generally configured to publish the Content Item when it reaches a public State. Once a Content Item is no longer valid or useful on a Web Site, a Role can transition it to an archive State, or it may enter an archive State through an aging Transition. Content Items in the archive State are unpublished from the Web Site and are saved for reference.

The following diagram shows the same Workflow as the diagram in the previous section, but here a Content Item's lifecycle is also recorded. The Content Item is created and saved by a user in the Author Role. Once it is saved, it enters the Draft State. The Author has access to the Draft State and performs a Submit Transition on the Content Item. Since any role may transition the Content Item, it moves to the Approval State. Since no users approve the Content Item before its Start Date is reached, on the Start Date, an Aging Transition moves the Content Item to the Public State. The Aging Transition automatically sends Notifications to all members of the Roles that had access to the Content Item in the Approval State. Eventually, the Content Item is no longer current on the Web Site. An Editor, the required Transition Role, performs an Unpublish Transition and the Content Item moves to the Archive State. The Unpublish Transition sends a Notification to the company auditor that the Content Item is in the Archive State.

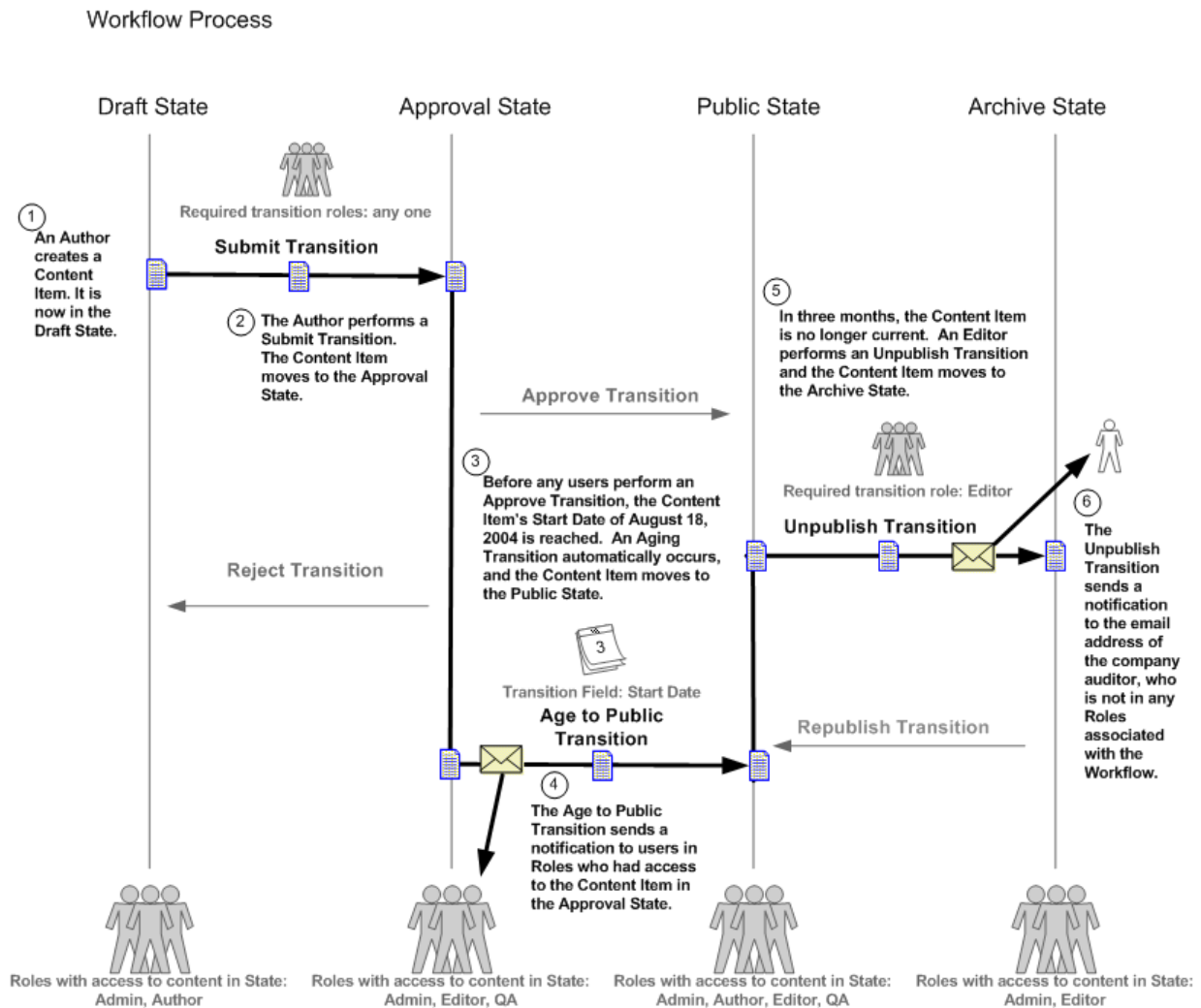


Figure 20: Workflow Processing

Assembly Engine

The Assembly Engine includes the XML and XSL files, applications, and component registrations responsible for the aggregation and formatting of Content Items into various outputs produced by Rhythmyx.

Architecture

The central component of the assembly engine is the Variant. Each Variant includes a template, an associated Content Assembler application, and any number of Slots. Most Variant templates are XSL stylesheets which format Content Item data in one of the ways it may be viewed on a Web Site. Slots are the locations on the output where other Content Items may be included. A Variant is marked as either a Page or a Snippet. Pages represent a complete Web Page and Snippets represent a portion of a Web Page.

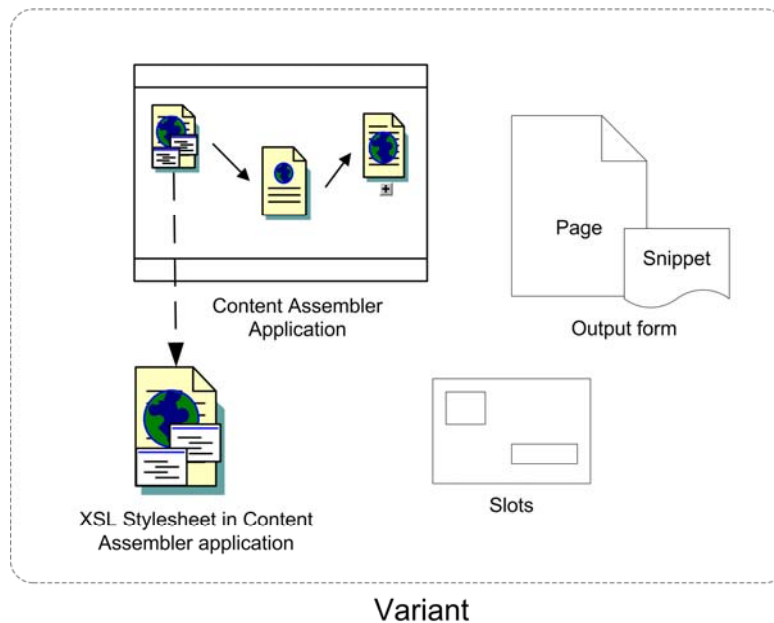


Figure 21: Components of a Variant

The Content Assembler applications associated with Variants perform the process of formatting raw content with templates. The applications extract Content Item data from Rhythmyx tables, insert the data into XML files, and format it using the XSL stylesheet templates. Usually, implementers create one Content Assembler application which generates output for all of a Content Types' Variants; however, more than one assembler application may apply to a single Content Type.

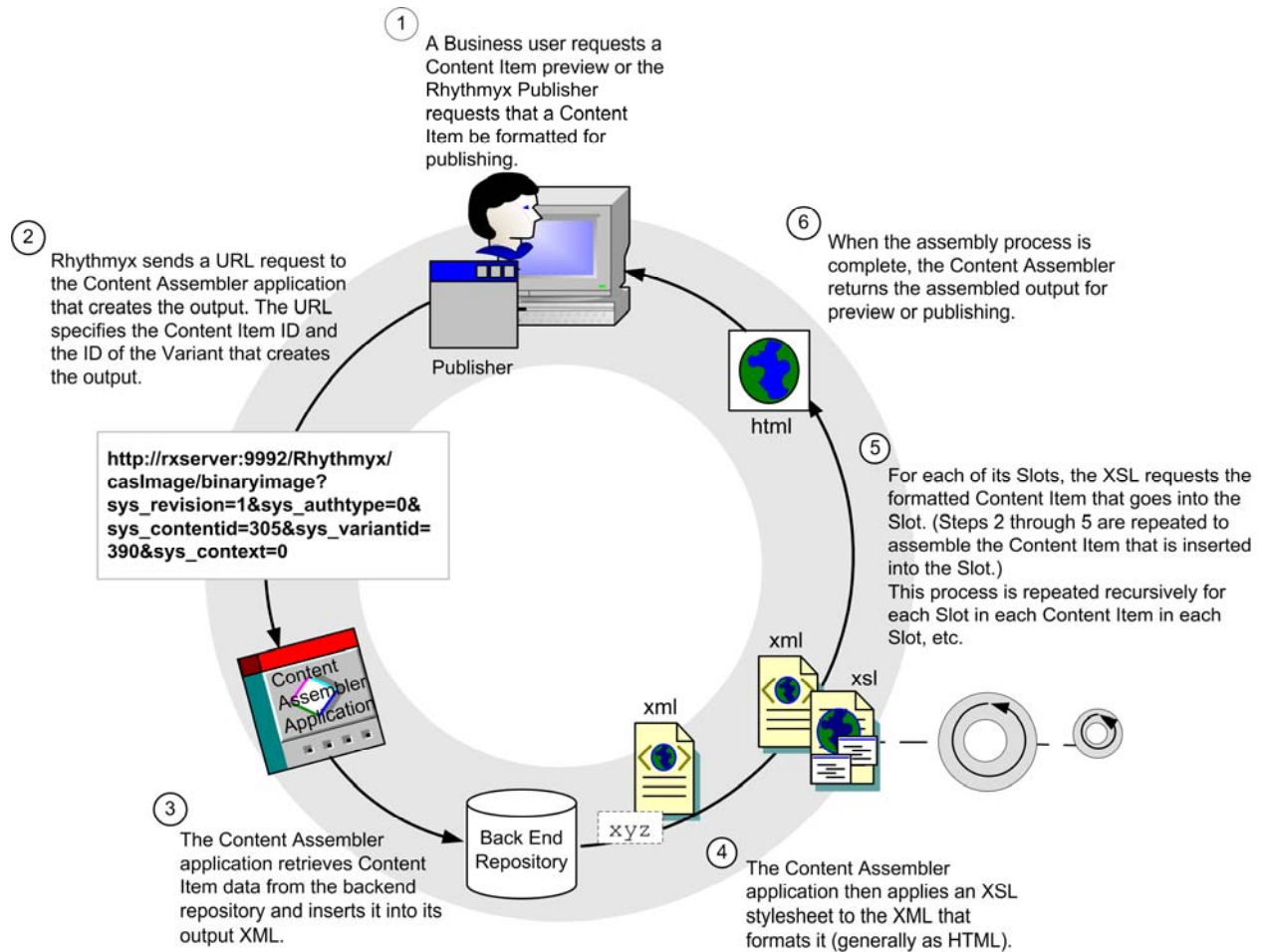
Processing

To assemble a Variant of an item, Rhythmyx applies formatting to the item's local content and inserts Snippets into the Variant's Slots. This process is referred to as recursive rollup because Rhythmyx must also format the embedded Snippet's local content and insert additional Snippets into its Slots, and format the local content of these Snippets and insert Content into their Slots, and so on. When the recursion is complete, all embedded Content is formatted and inserted and Rhythmyx returns the output Page or Snippet.

Assembly is triggered when a business user requests a preview of an assembled output of a Content Item or a Rhythmyx Publisher requests that a Content Item be formatted as a certain output for Publishing. When Rhythmyx receives the request, it sends a URL to the associated Content Assembler application. The URL specifies the application to call, the Content Item to be formatted, and the Variant associated with the output. The Content Assembler application uses this information to obtain the Content Item data (and URLs of related Content Items) from the Rhythmyx backend tables and to select the correct XSL stylesheet for formatting the Content Item. It inserts the Content Item data into an XML file and applies the XSL stylesheet to it. The stylesheet formats the content as output text, in most cases, as HTML. The stylesheet also requests the formatted Content Items that belong in each of the Slots in the output.

Each time a formatted Content Item is requested for a Slot, a URL is sent to the Content Assembler application that creates the appropriate output and that Content Assembler application performs the process outlined above for formatting the related Content Item. This occurs for each related Content Item until all levels of related Content Items are included in the embedded content.

When the entire process is complete, the Content Assembler application returns the originally requested output for preview or processing.



Publishing Engine

The Publishing Engine consists of the registrations, applications, and Java code responsible for the delivery of Rhythmyx Content Items.

Architecture

The main components of the Publishing engine are the Publishing Manager, one or more Publishers, Sites, Content Lists, and Editions:

- A Content List specifies which Content Items to extract from the database for Publishing.
- An Edition specifies one or more Content Lists to publish and the order in which to publish them.
- The Publishing Manager receives a publishing request, selects the Publisher to run the job, and passes the Content Lists to publish to the Publisher. Rhythmyx sends the publishing request to the Publishing Manager when an Edition is published.
- Publishers are applications that reside on the Rhythmyx Server or remote locations, and run off a Tomcat Web application server or another J2EE Web application server. A Publisher runs after it receives a request from the Publishing Manager. The Publisher is responsible for extracting Content Items from the Rhythmyx database, assembling them to produce final content pages, and saving or sending them to their destination.
- A Site defines a location where output will be published. A Site may be a file system or a database or some other destination (for example, a Portal). Rhythmyx can maintain multiple Sites on the same machine.

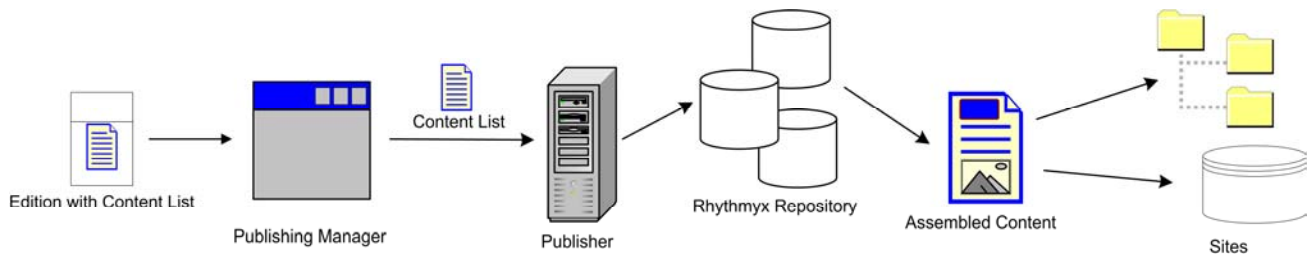


Figure 22: Publishing Engine Architecture

Processing

A System Administrator can 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. The Publishing Manager compiles lists of Content Items to publish, and determines the publishing Site and the Publisher. After it compiles each content list, the Publishing Manager issues an HTTP request via Simple Object Access Protocol (SOAP) to the location where the Publisher resides. The request includes the compiled Content List and directs the Publisher to publish it.

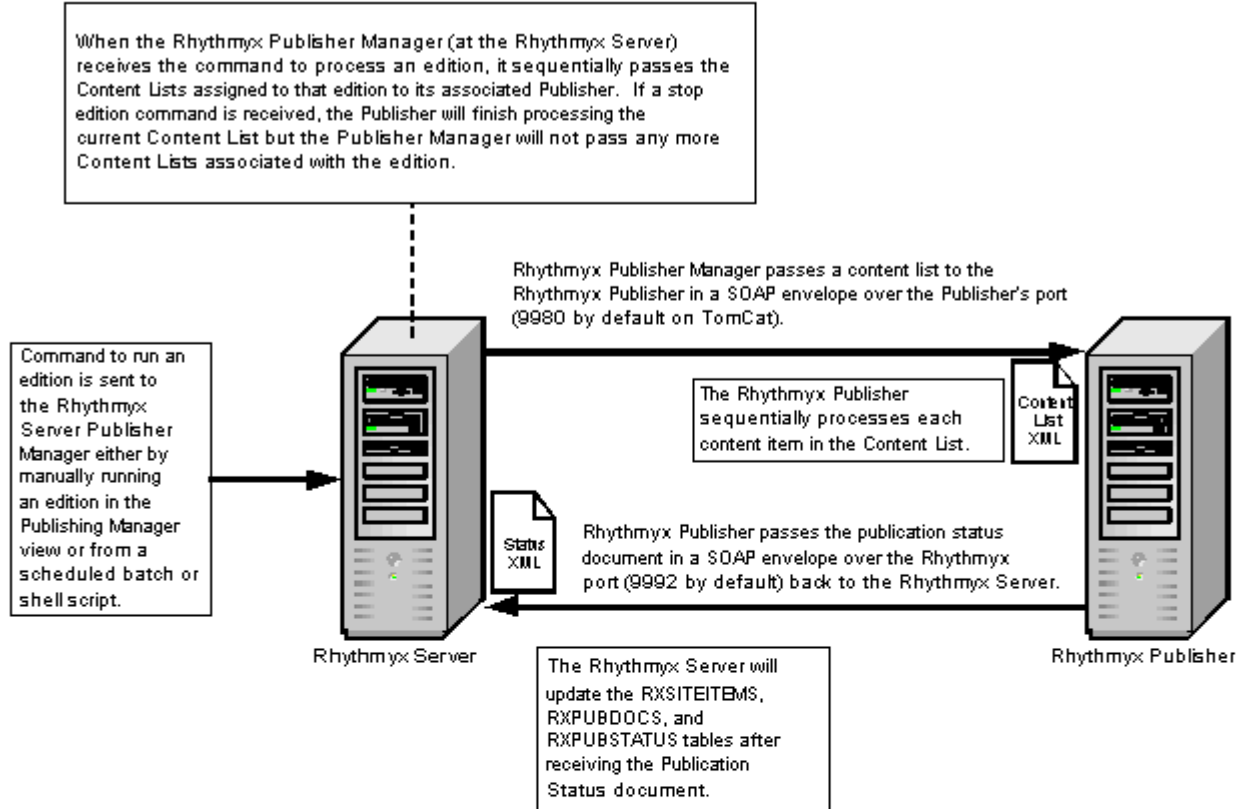
After receiving a request, the Publisher

- retrieves the Content Items specified on the content list from the Rhythmyx server;
- calls the content assembler application to generate the output;

- sends the Page(s) to the publishing Site (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 Publishing Manager updates the Rhythmyx tables.

Rhythmyx Publishing Process



Content List Processing

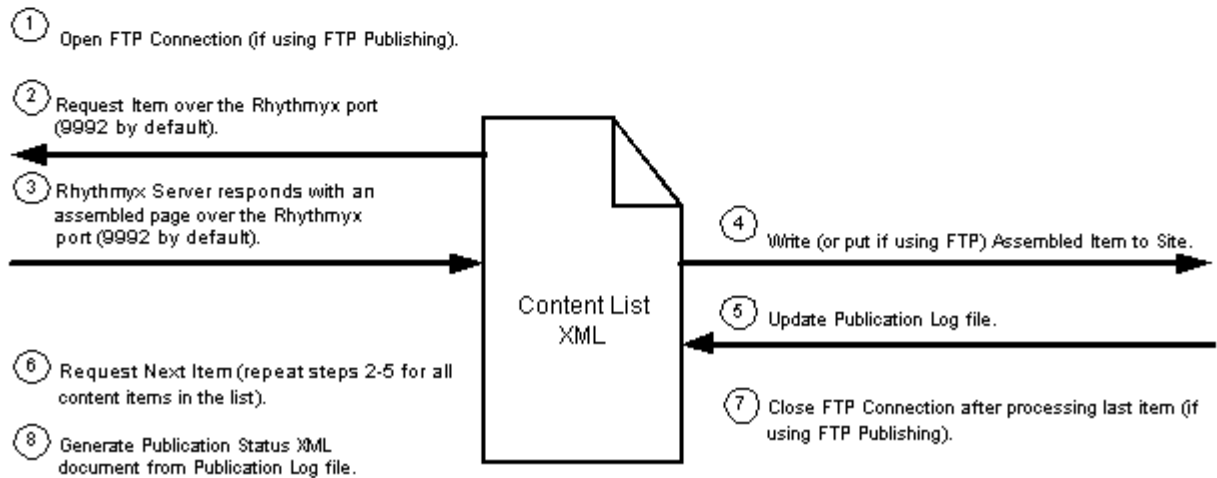


Figure 23: Publishing Process
Publishing Engine Processing

CHAPTER 4

Clients and Interfaces

Rhythmyx provides several clients and interfaces that enable users to perform the functions particular to their Roles. In the Workbench, implementers create applications and customize the Content Explorer interface. In the Server Administrator, administrators perform functions such as configuring back end connections and setting up security. Business Users work in the Content Explorer interface to create Rhythmyx content. Implementers use the Multi-Server Manager to move Rhythmyx components from one server to another, and they customize code with Web Services when sharing information with other applications on the Web.

Workbench

The Workbench is the Rhythmyx environment in which implementers:

- Define applications which process the data that Rhythmyx delivers or receives through HTML pages;
- Configure Content Explorer Action Menus, Display Formats, and Content Views;
- Set up Rhythmyx Relationships.

The Workbench's Application tab includes an Explorer that lists Rhythmyx applications and an Editor for modifying or creating them.

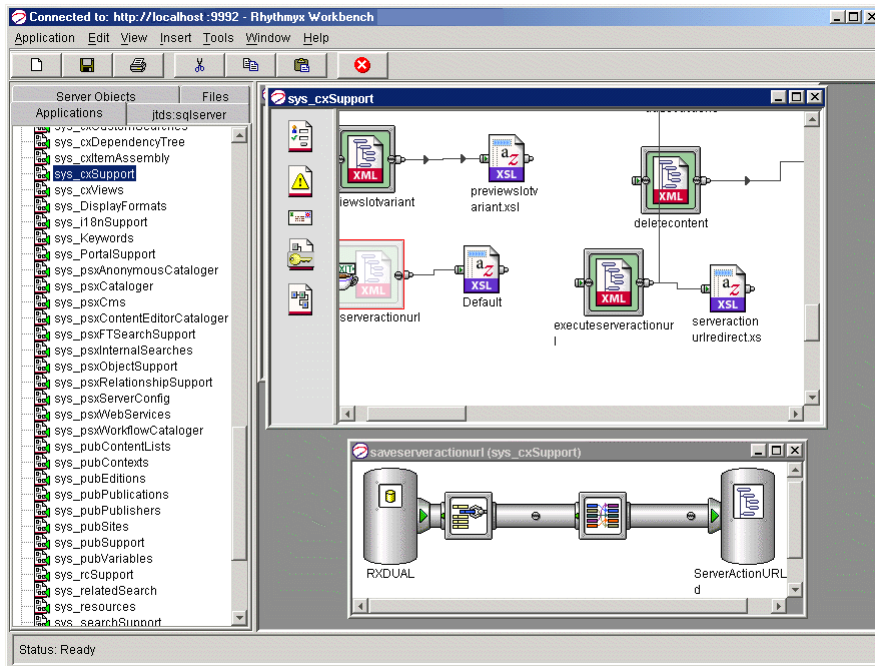


Figure 24: Rhythmyx Workbench

Users click the Files tab to access files to drag and drop on the application, and may create database tabs for accessing backend database tables to drag and drop on the application.

Users can also click the Server Objects tab to access Java exits (Java classes or scripts) to drag and drop on an application or to access the editors for Action Menus, Content Views, Display Formats, and Relationships:

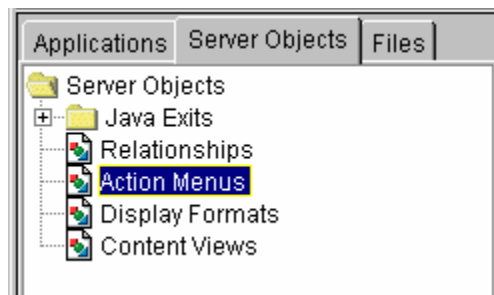


Figure 25: Server Objects Tab of Workbench

To access the editors, users click on the editor name. The editor displays a list of the objects available and a window for viewing and modifying information about the object.

For example, clicking on the Display Formats node displays the Display Formats editor:

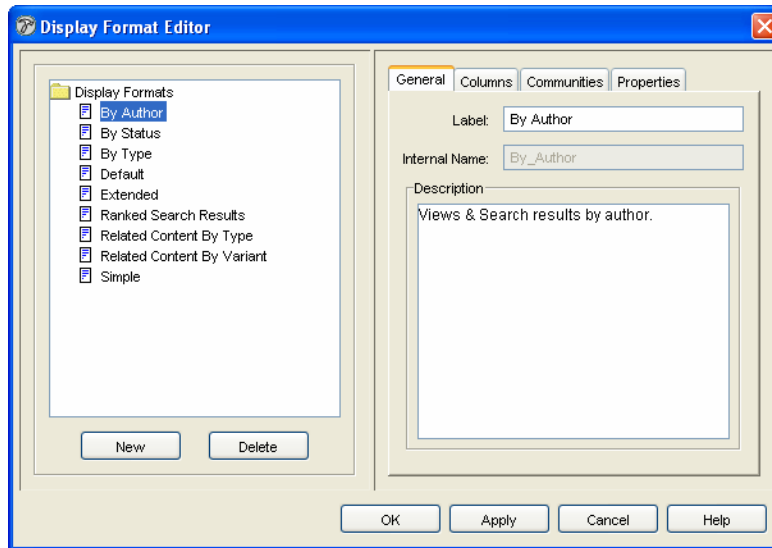


Figure 26: The Display Formats Editor

Server Administrator

The Server Administrator is the Rhythmyx interface that the System Administrator uses to maintain the Rhythmyx Server.

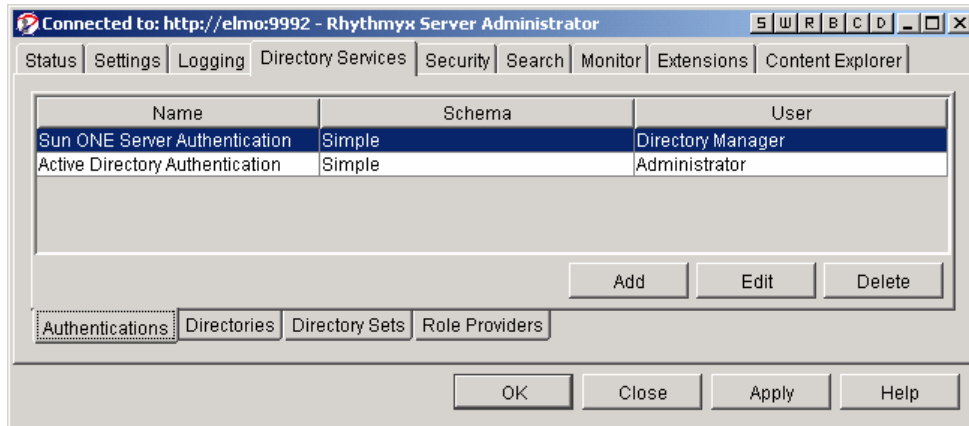


Figure 27: Rhythmyx Server Administrator

The Server Administrator separates administration tasks into nine categories displayed by its upper tabs.

Status

In the Status tab, an administrator can stop and start Rhythmyx applications and monitor their statistics.

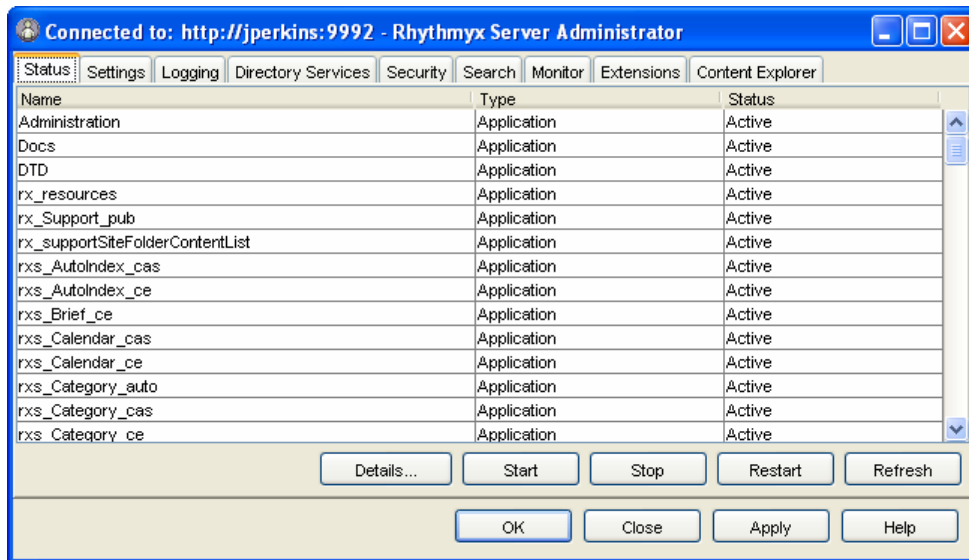


Figure 28: Status Tab in Server Administrator

Settings

The Settings tab lets administrators optimize Rhythmyx's performance by specifying idle time limits, maximum connections, and similar settings for backend servers, the Rhythmyx Server, and Rhythmyx applications. In addition, this tab lets the administrator enable or disable the Server Cache and set its size.

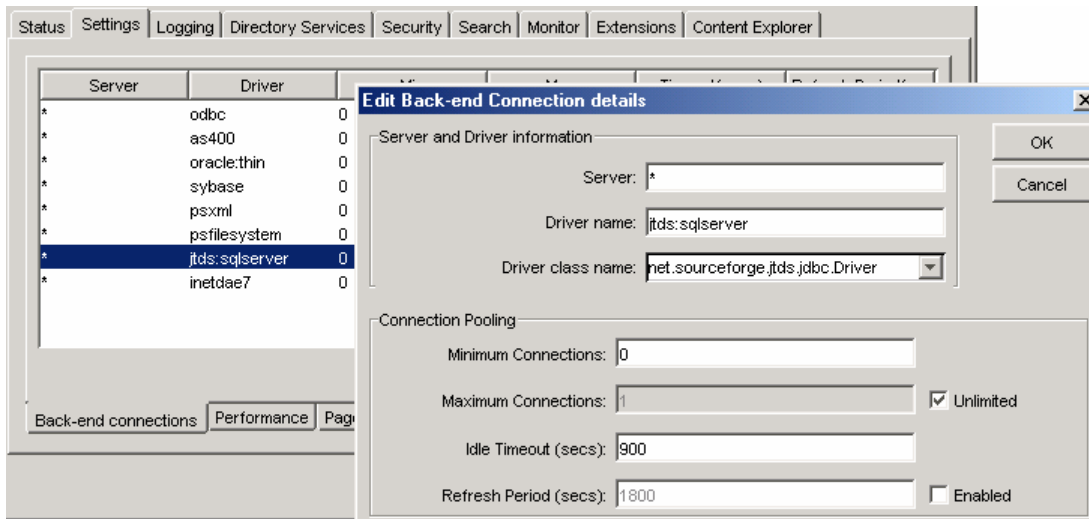


Figure 29: Setting Back-end Connections in the Server Administrator

Logging

In the Logging tab, an administrator can set the types of events to log, specify how long to save log files, and query log records for information.

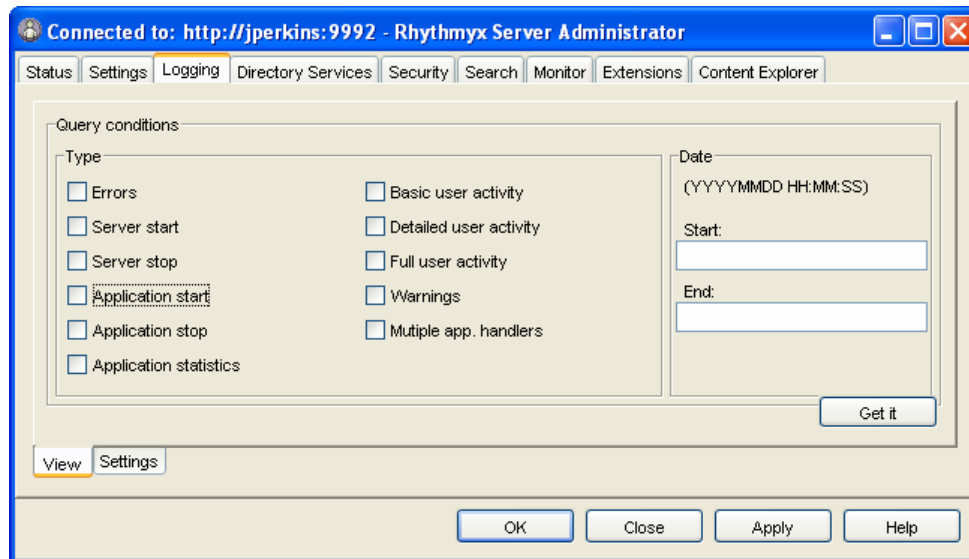


Figure 30: Logging Tab, Server Administrator

Directory Services

Directory Servers are third-party systems that store various types of user data. In Rhythmyx, Directory Services can provide authentication. In the Directory Services tab of the Server Administrator, administrators can register Directory Services, add authentication information for users connecting to a Directory Service, and provide other data necessary for connecting to and using Directory Services.

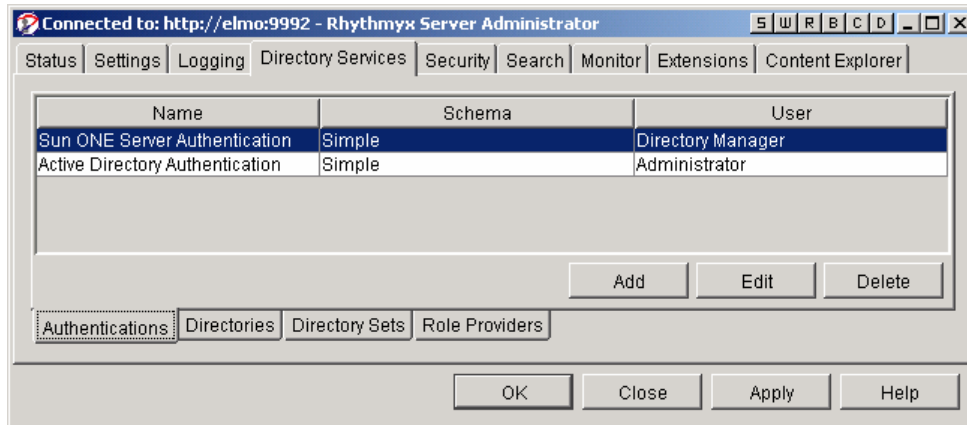


Figure 31: Directory Services tab, Server Administrator

Security

The Security tab allows administrators to set up most of Rhythmyx's security features. It includes sub-tabs for configuring Security Providers, Roles, Server ACLs, Back-end Credentials, Encryption, and Java exit security.

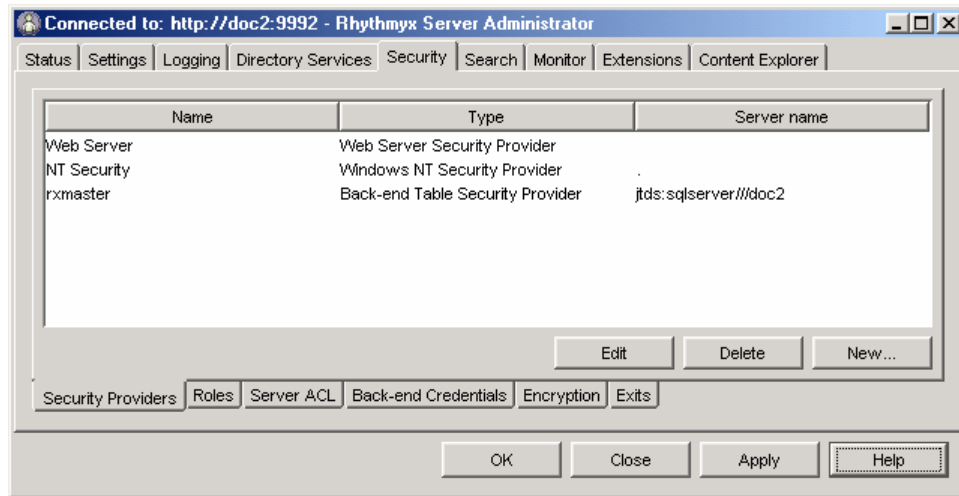


Figure 32: Security tab in the Server Administrator

In the Security Providers sub-tab, administrators register security providers and associate them with servers or users. Security providers perform authentication for servers. For more information about them, see **Data Protection** (on page 39).

In the Roles sub-tab, administrators add or edit Rhythmyx Roles and their members on the Rhythmyx Server. Roles must be defined here before Workflows and Communities can use them.

In the Server ACL sub-tab, administrators define the level of access users and Roles have to the Rhythmyx Server.

In the Back-end Credentials sub-tab, administrators specify user IDs and passwords for accessing the Rhythmyx backend databases.

In the Encryption sub-tab, administrators can configure SSL encryption as the default for Rhythmyx applications. Encryption scrambles data transmitted between servers and only allows trusted users to unscramble it.

In the Exits sub-tab, administrators can enable or disable the Java exit sandbox. Enabling the sandbox restricts the resources that Java exits can access; disabling the sandbox allows Java exits to access all Rhythmyx resources.

Search

The Search tab displays default search configuration values for the Full Text Search engine. It is only available if the Full Text Search is installed and enabled. In this tab, the administrator can change the default values for the Search engine Server, Port, and Config directories. If multiple Rhythmyx Servers exist, the administrator may change which one of them serves as the Admin Master to receive errors and other administrative information. In **Default Search Properties**, the administrator can change the default values for the concept search expansion level and tracing.

For more information, see the section “Search Configuration” in the Server Administrator online help.

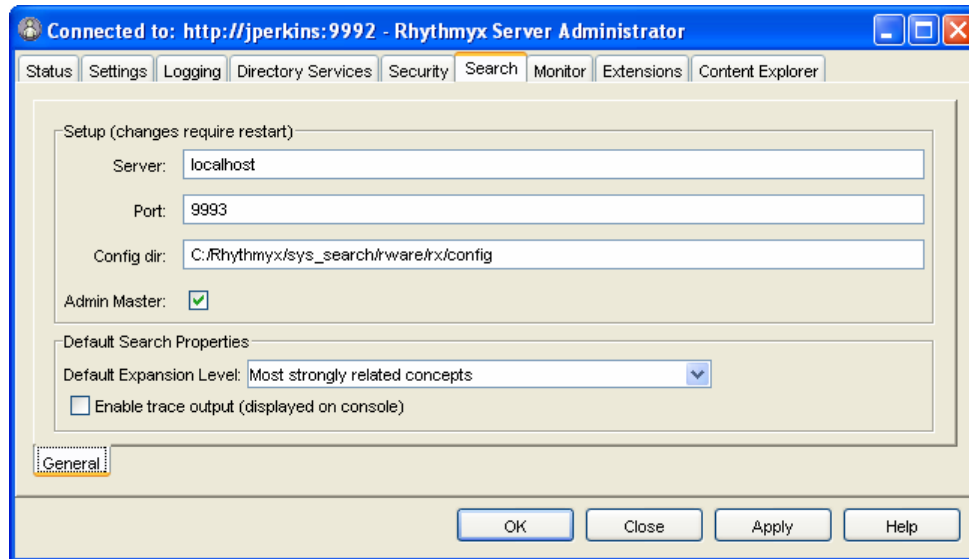


Figure 33: Search tab, Server Administrator

Monitor

The Monitor tab lets administrators who are accessing a Rhythmyx Server remotely enter Server commands and view the Rhythmyx Server response on a console.

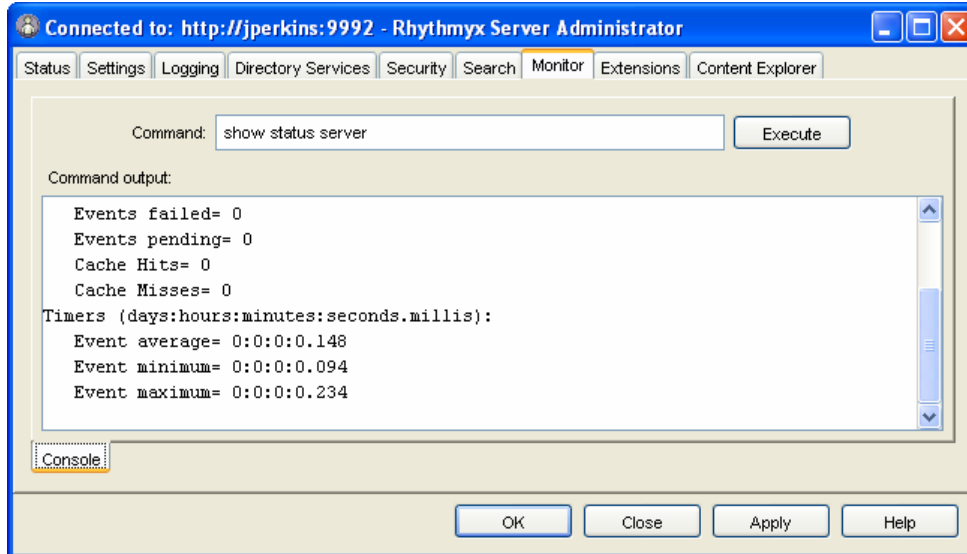


Figure 34: Monitor tab, Server Administrator

Extensions

In the Extensions tab, administrators or implementers can maintain Java extensions and extension handlers.

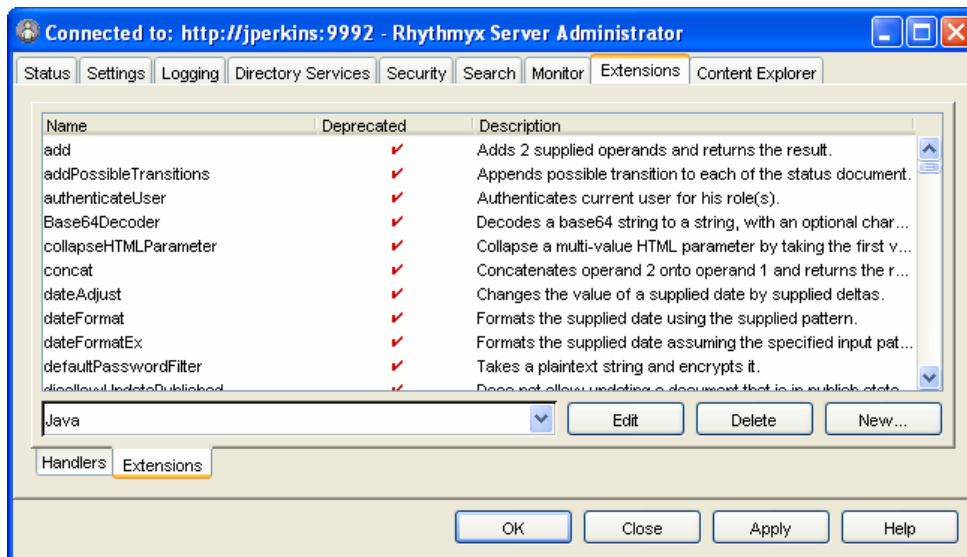


Figure 35: Extensions tab, Server Administrator

Content Explorer

In the Content Explorer tab, administrators can configure which Java Plugin Content Explorer uses and its download location, and choose when to refresh the Content Explorer screen.

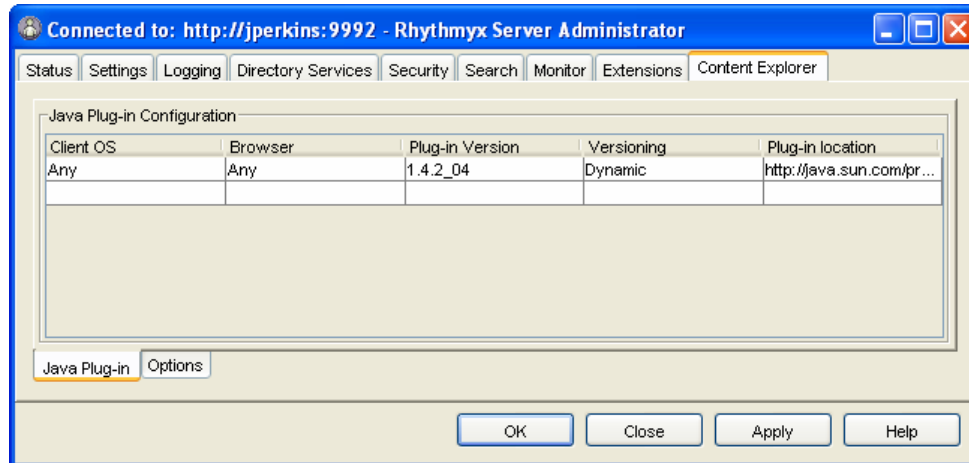


Figure 36: Content Explorer Tab, Server Administrator

Multi-Server Manager

Rhythmyx implementers use the Multi-Server Manager to move a Rhythmyx CMS or portions of a Rhythmyx CMS from a source server to a target server.

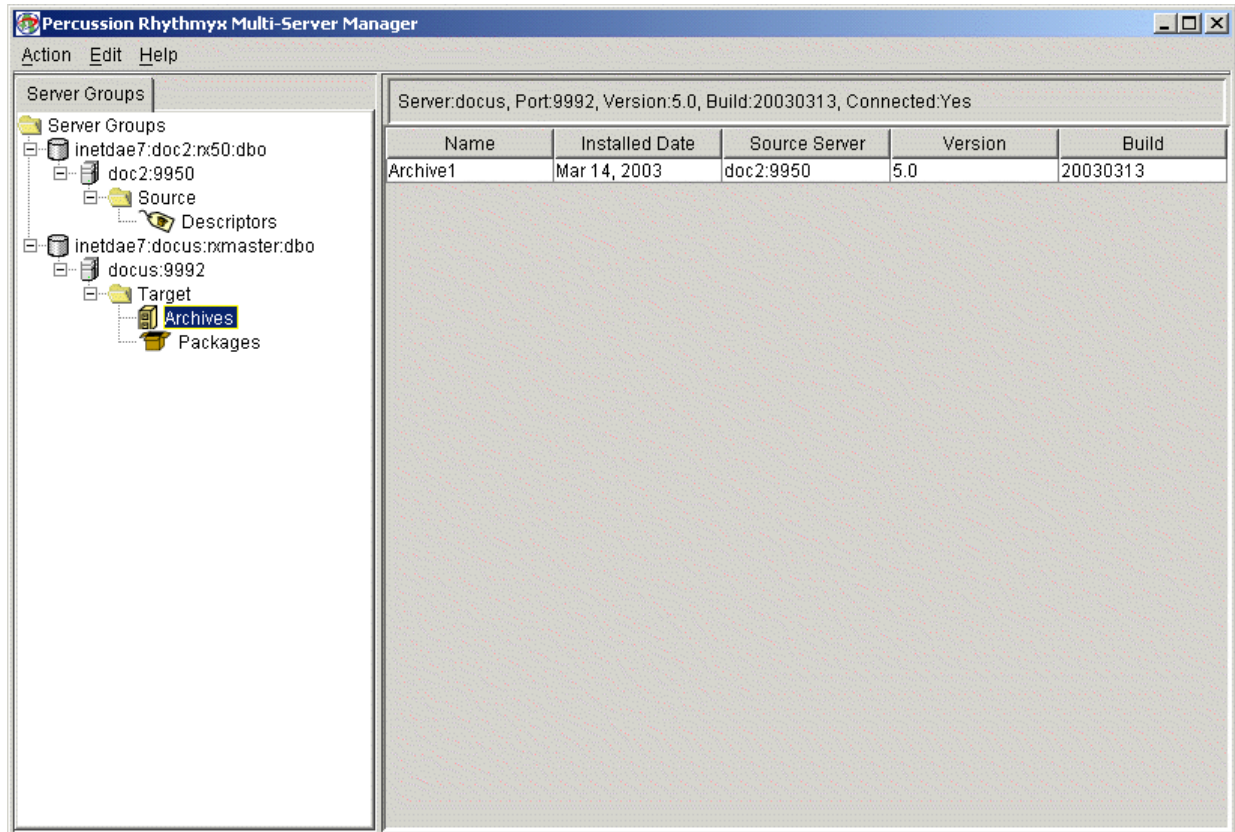


Figure 37: Rhythmyx Multi-Server Manager

The Multi-Server Manager guides users through each step of the installation process including selecting elements to install, packaging the elements in an archive file, and installing the archive file's contents on the target server.

The parts of a CMS that developers can install using the Multi-Server Manager include Action Menus, Content Assemblers, Content Items, Content Lists, Content Types, Folders, Locales, Relationships, Sites, Slots, and Workflows.

The Multi-Server Manager is a stand-alone tool that functions as a client and interacts with the target and source servers. The Rhythmyx Installer loads it with other client tools into the Rhythmyx root directory.

Content Explorer

Content Explorer is the default home page when users log into the Rhythmyx CMS interface. It is the Rhythmyx interface through which end users create, modify, and process Content Items. The Content Explorer interface appears and functions similarly to the Windows Explorer interface: the left pane displays a Navigation Tree and the right pane is a Display pane that lists the contents of the node selected in the Navigation Tree. The Content Explorer also includes a Menu Bar with drop menus, and a banner with user session properties.

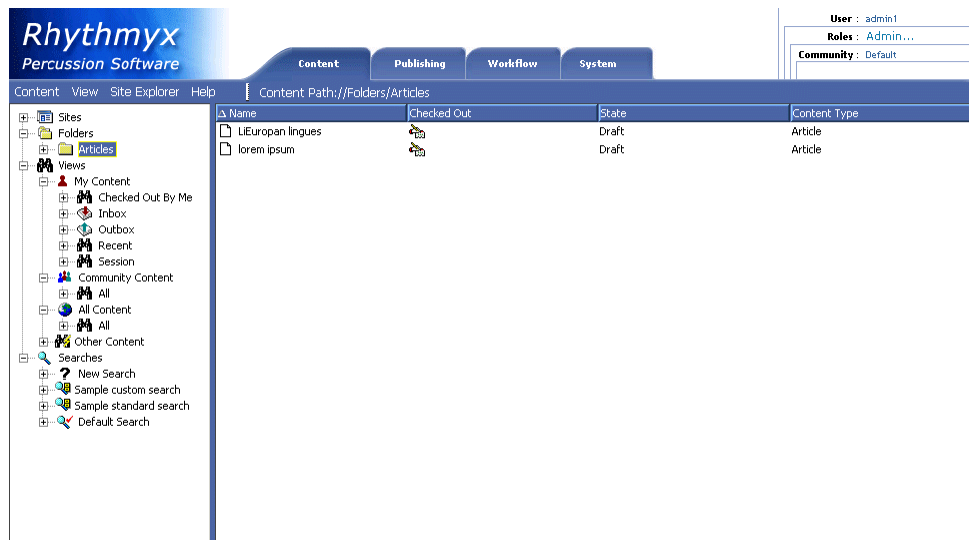


Figure 38: Content Explorer

In the Display Pane, users can select a Content Item node and right-click to open a menu of options for acting on the Content Item.

In the Menu Bar, the Content menu displays options for working with folders, views, and saved searches in the Navigation Tree. Users can access a menu of similar options by right-clicking on a node in the Navigation Tree. The other Menus display options for refreshing the page, changing its appearance, activating Rhythmyx's Site Explorer interface, and accessing Help.

At the top of Content Explorer, the Community and Locale entries link to pages that let users change these properties.

Users' Roles determine the Content Explorer options and functions and the Content Items that they may access.

The tabs at the top of the page are only visible to users who have access to windows other than Content Explorer. They access the Publishing Administrator, Workflow Administrator, and System Administrator, which allow CMS Administrators to register the Publishing, Workflow, and general System components of the CMS. Most business users do not have access to these tabs.

Web Services API

Web Services enable applications or processes to interact based on standard Internet protocols. Rhythmyx includes a Web Services API that serves as an interface between the Rhythmyx Server and remote client applications or other servers. Rhythmyx Web Services let third-party applications access Rhythmyx functionality within their own environment. For example, through Web Services a third party application could create a Content Item or move a Content Item from one Folder to another.

Rhythmyx Web Services operates as a registered SOAP service on any J2EE application server. By default, the Rhythmyx Installer installs it on the default Tomcat server.

The Web Services server receives incoming messages from remote client applications and Rhythmyx Web Services messages from Rhythmyx. The messages are all in XML format. The Web Services server processes the messages, and sends client requests to the Rhythmyx server and Rhythmyx responses to the client application. The graphic below shows the general architecture of Rhythmyx Web Services:

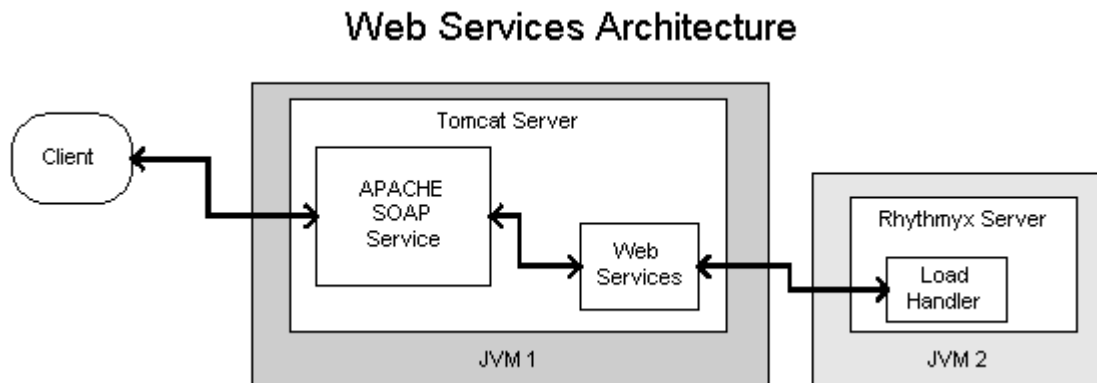


Figure 39: Web Services Architecture

Each message to Rhythmyx includes a header with authentication information and a body that includes the name of the function requested and parameters that the function requires. The structure of the message body for each function is defined in one of the Rhythmyx Web Services XML Schema Definition (xsd) files. These files are installed into the [Rhythmyx root]/WebServices directory.

Each schema definition file contains elements for one or more related functions. Each of these elements defines the structure of the body portion of the request message made to Rhythmyx and the response message sent from Rhythmyx. The response message may simply indicate that the function succeeded or failed or may return Content Item data for modification.

Convera Full-Text Search Engine

By default Rhythmyx provides a database search engine for searching Content Items. This search engine uses the query functionality embedded in the database system.

Rhythmyx also offers a full-text search engine that uses Convera's RetrievalWare (RW) server to include many advanced search features such as the ability to search the content in Content Item fields and options for searching for synonyms of search terms. The Convera server can convert over 250 different file formats into searchable text. This search engine is available by special license and requires additional installation steps.

Typically, the full-text search engine is installed on the same machine as the Rhythmyx master server. In a system that runs multiple content servers, a search configuration is entered on each of these servers to specify the location of the full-text search engine. The full-text search engine can also run on a different machine than the Rhythmyx server either because running both servers on the same machine degrades the performance of one or both servers, or because the full-text search indices consume too much disk space. If the engine runs on a different machine, the Administrator must update the search configurations to specify the new location of the full-text search engine.

For more information about deploying Rhythmyx engines, see *Physical Architecture, Deployment, and Scaling* (on page 39).

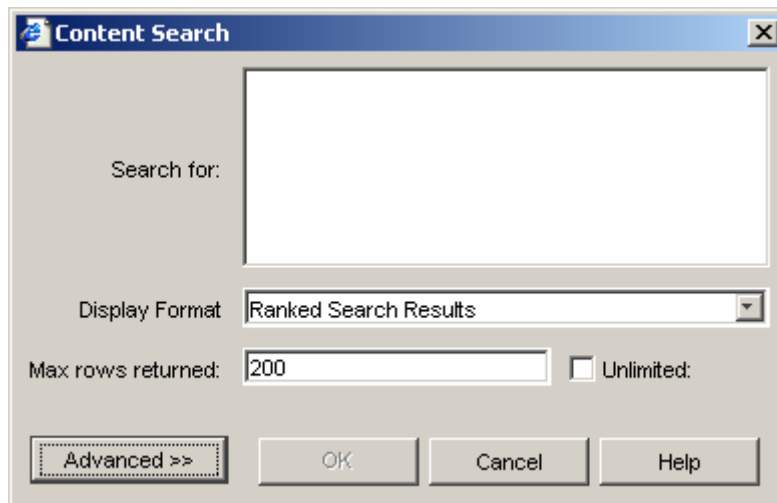


Figure 40: Full-text Search dialog

CHAPTER 6

Rhythmyx Modules

As an Enterprise Content Management system, Rhythmyx offers several distinct types of applications, such as Web Content Management (WCM), Document Management (DM), and Digital Asset Management (DAM). Some of these capabilities are offered as Modules to enable customers to license Rhythmyx for use in one of these specific applications. However, while Modules provide functional feature groupings for licensing, they are transparent to both the logical architecture of the Rhythmyx server and the actual physical software components that make up the Rhythmyx system. Thus, the Modules included in any given Rhythmyx implementation will not impact deployment, scaling, or performance considerations.

Modules are generally offered through specific packaging and licensing options (contact Percussion Sales for further information). The following Modules are available in Rhythmyx:

Module	Description	Notes
Web Content Management Module	The Web Content Management Module streamlines the process of updating Web content, facilitates content reuse, and enables customers to target content to audience-specific channels.	
Document Management Module	The Document Management module allows Rhythmyx users to centralize and organize documents, with the added functionality of Rhythmyx's Workflow, security, and content reuse capabilities.	
Digital Asset Management Capabilities	Included as part of both the Web Content Management Module and the Document Management Module, Rhythmyx's Digital Asset Management Capabilities allow organizations to include digital assets such as images, graphics and other media files in documents and Web content.	
Portal Connectors	Portal Connectors enable delivery of content through a native Portal repository, as well as contribution of content into Rhythmyx through a set of Portlets that run in the Portal. They also enable the installation of Rhythmyx servlets into the Portal's associated J2EE container.	Available for IBM, BEA, and Oracle.

Module	Description	Notes
Rhythmyx Express Portal	The Rhythmyx Express Portal is a cost-effective, workgroup portal solution based on J2EE standards. It can be implemented quickly and easily while providing an organization with the core portal capability to be productive. It includes out-of-the-box personalized content delivery, project collaboration, and content capture capabilities.	

Connectors

Rhythmyx provides several Connectors that enable users to connect Rhythmyx with other software packages to create Rhythmyx Content Items or work in the Rhythmyx Content Explorer interface. Some Connectors enable automated content upload, such as the Enterprise Content Connector. Some Connectors provide options for client contribution, such as the Microsoft Word Connector. Other Connectors provide integration to other enterprise Portals, such as the BEA WebLogic and IBM WebSphere Portal Connectors

Some of the Rhythmyx Connectors are covered in this section. However, Connectors may be added at any time, by Percussion and Percussion partners. Please check with Percussion Software for the latest list and information on other Connectors.

Enterprise Content Connector

The Rhythmyx Enterprise Content Connector is a stand alone tool that uploads content in HTML files, image files, and other file types from an external site into a Rhythmyx server or a filesystem.

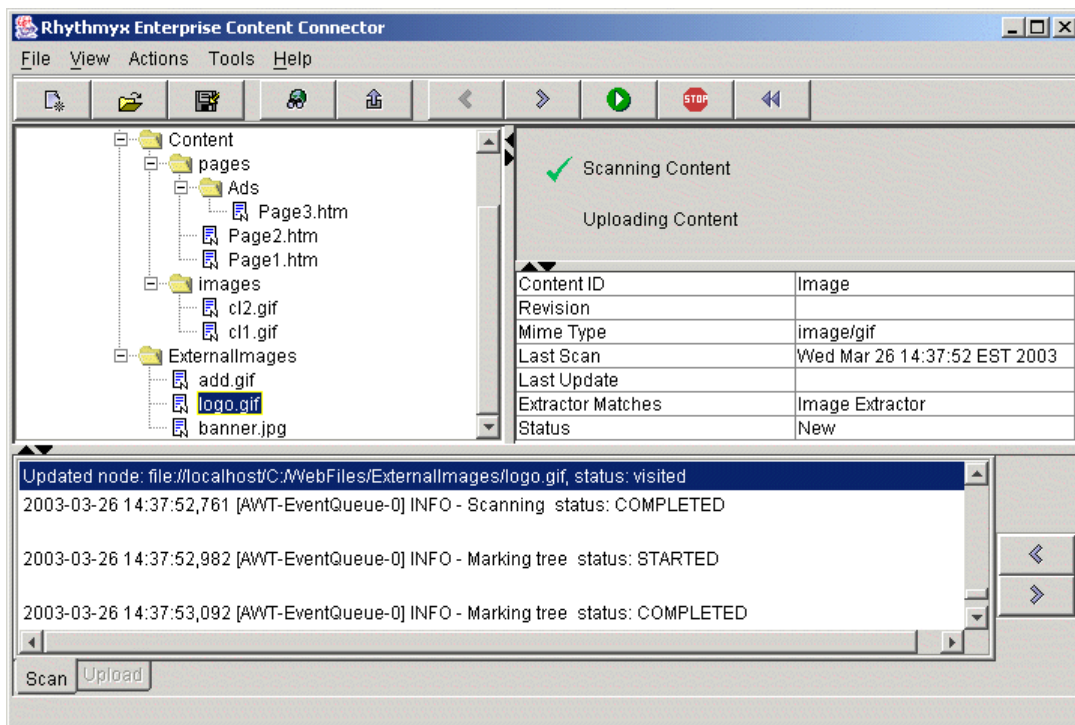


Figure 41: Enterprise Content Connector Interface

Implementers use the Enterprise Content Connector to automate periodic uploads of content stored in another location (for example, to perform syndication feeds). During scheduled uploads, the Enterprise Content Connector compares current information to information that it previously uploaded and incrementally uploads new or modified content.

Implementers can run the Enterprise Content Connector through the Content Connector Interface, or on a command line. For scheduled or incremental uploads, users can configure a batch file to run the Enterprise Content Connector at regular intervals with a scheduler program.

Word Connector

Rhythmyx includes a Word Connector that allows content contributors who work primarily in Microsoft Word to continue to create and edit content in their familiar Word environment and easily save it to Rhythmyx as Rhythmyx Content Items.

The Word Connector uses Word-based Content Editors to let users create Content Items in Word and save them to Rhythmyx. A Word-based Content Editor opens a Word document with a Word template that associates Word styles with Rhythmyx Content Editor body and metadata fields.

The Connector also enables various Rhythmyx features to appear in Word. For example, the Save to Rhythmyx control lets a user save a document with an alternate save command to upload it to Rhythmyx as a Rhythmyx Content Item. Rhythmyx saves the Word.doc file in a column in the backend table for the Word-based Content Editor. The Inline Link button lets users add links to Rhythmyx Content Items within the body of the Word document.

After a Word-based Content Item is created and saved, a user can either open it in Rhythmyx and access Microsoft Word from a control within the Content Editor or open it directly in Microsoft Word. The user edits the content in Word and saves it back to Rhythmyx.

Rhythmyx uses a Word Macro to manage Word menu extensions, and an ActiveX control to manage the browser-to-Word launching behaviors.

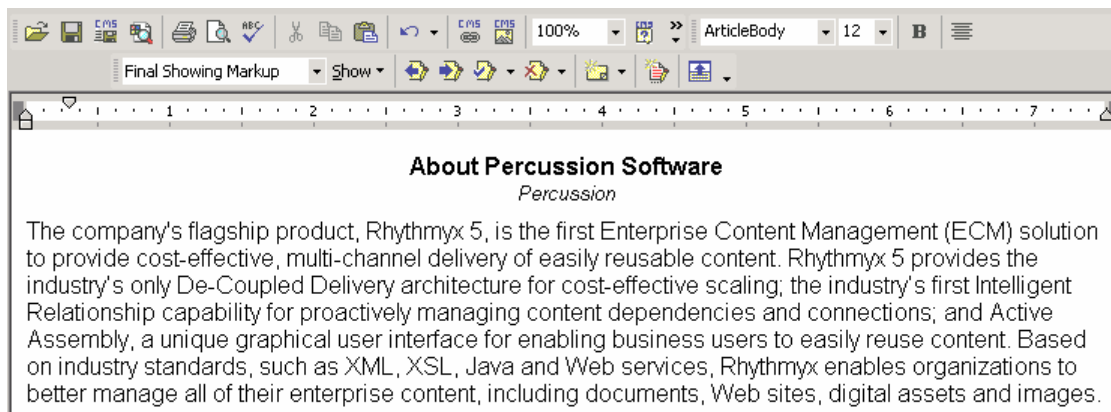


Figure 42: A Microsoft Word document using a Word Connector template. Icons for saving to Rhythmyx and including Rhythmyx links and graphics appear in the Toolbar.

BEA WebLogic Portal Connector

For customers who have implemented or are planning to implement a BEA WebLogic Portal environment for internal use, Rhythmyx's BEA WebLogic Portal Connector lets users access Rhythmyx through the WebLogic Portal, allowing them to perform all of their work functions through the same environment. With a minimum amount of configuration for Rhythmyx implementers, the Portal Connector enables content contributors to access Rhythmyx Content Explorer through the Portal, and Web Administrators to publish content directly to the Portal. The Connector provides a Rhythmyx Portal and Rhythmyx Portlets through which users can access Content Explorer and various Content Explorer options such as Inbox and Search.

The Portlets use Rhythmyx Web Services to make requests and receive responses from Rhythmyx. Web Services are installed with other applications on the WebLogic server.

A custom publisher plug-in lets Web administrators use a Rhythmyx publisher to deliver Rhythmyx data to the BEA Portal repository. The administrator may install the Publisher service with other applications on the WebLogic server to use its capabilities as a J2EE servlet container.

The Portal Connector also includes a sample delivery Portlet and sample Rhythmyx BEA Publishing components that implementers can use as models for their own configuration of Rhythmyx and BEA WebLogic. The sample delivery Portlet displays the Content Items published to the Portal repository. The sample publishing components include a Rhythmyx Publisher, Edition, Site, Content Lists, and Location Schemes for publishing to the BEA Portal.

IBM Websphere Portal Connector

The IBM WebSphere Portal Connector lets users who perform their work functions through an IBM WebSphere Portal environment access Rhythmyx through the WebSphere Portal, allowing them to perform all of their work functions through the same environment. With a minimum amount of configuration for Rhythmyx implementers, the Portal Connector enables content contributors to access Rhythmyx Content Explorer through the Portal, and Web Administrators to publish content directly to the Portal. The Connector provides a Rhythmyx Portal and Rhythmyx Portlets through which users can access Content Explorer and various Content Explorer options such as Inbox and Search.

The IBM WebSphere Portal Connector supports Rhythmyx content creation in the WebSphere Portal and provides the components and logic necessary for publishing to the Portal. The Connector installs Rhythmyx Portlets into WebSphere's sample Portal. Users can access Content Explorer and various Content Explorer options such as Inbox and Search through the Rhythmyx Portlets.

The Portlets use Rhythmyx Web Services to make requests and receive responses from Rhythmyx. Web Services are installed with other applications on the WebSphere server.

The WebSphere Portal Connector includes Rhythmyx's database publisher plug-in, which lets Web administrators publish Rhythmyx data to the IBM WebSphere Portal repository. The administrator may install the Rhythmyx Publisher service with other applications on the WebSphere server to use its capabilities as a J2EE servlet container.

The Portal Connector also includes a sample delivery Portlet and sample Rhythmyx WebSphere Publishing components that implementers can use as models for custom configurations. The sample delivery Portlet displays the Content Items published to the Portal repository. The sample publishing components include a Rhythmyx Publisher, Edition, Site, Content Lists, Context, and Location Schemes for publishing to the IBM WebSphere Portal.

Oracle Portal Connector

Rhythmyx's Oracle Portal Connector lets users access Rhythmyx through the Oracle Portal, giving them the option of performing all of their work functions through the same environment. By providing a Rhythmyx Portal and Rhythmyx Portlets, the Portal Connector enables content contributors to access Rhythmyx Content Explorer and its functions, including content creation, through the Portal. The Portal Connector also allows Web Administrators to publish content directly to the Portal.

The Portlets provided use Rhythmyx Web Services to make requests and receive responses from Rhythmyx. Web Services are installed with other applications on the Oracle server. A sample delivery Portlet that implementers can use as a model for their own configuration of Rhythmyx and Oracle is included. The sample delivery Portlet displays the Content Items published to the Portal repository.

Rhythmyx Express Portal

Rhythmyx Express Portal provides a Portal environment designed for workgroups that is quick and easy to implement, and provides users with seamless interaction with Rhythmyx. The Express Portal includes a Rhythmyx Content Explorer Portlet and other Portlets that display Rhythmyx Content Items, and allows users to work with them as they do in Rhythmyx. Rhythmyx Express Portal also includes content delivery, project collaboration, and content capture capabilities that make it easier to implement and use than an enterprise portal.

Copyright © 2004 Percussion, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws.
 This material is the confidential and trade secret information of Percussion, Inc.
 This product includes software developed by the Apache Software Foundation, <http://www.apache.org>
 \$Revision: 1.2.2.1 \$ from \$Date: 2003/11/12 05:11:50 \$ (GMT) [View Release Notes](#)

Figure 43: Rhythmyx Express Portal

Rhythmyx FastForward for Web Content Management

Part of the Web Content Management (WCM) Module, Rhythmyx FastForward for WCM is a complete reference implementation which includes a set of pre-built components and applications that allow a company to quickly bring their Web site content under content management. Managers therefore can focus their customization budgets on projects that offer a measurable return, rather than on those that simply enable basic Rhythmyx capabilities. FastForward's pre-built components include:

- A set of commonly used Content Types and Variants to help administrators quickly plan the content and output required by the system, and to save implementers development time.
- Content Types and Variants used to create navigational Content Items, such as site maps, breadcrumb lists, and navigation menus.



About Enterprise Investments
Press Release
Investment Advice
Insurance Advice
Estate Planning
Retirement
Tax
Mortgages and Home Finance
Home Purchase
Home Equity
Products and Services
Mortgages
Funds
Insurance Products

Figure 44: Output of a Navigation Content Item

- Global Templates. A Global Template is a Variant that defines common features for pages on a Web Site or in a document.

- Site Folder Publishing components. Site Folder publishing enables users to publish all of the content in a Content Explorer Site Folder tree to the same folder structure on a Site.

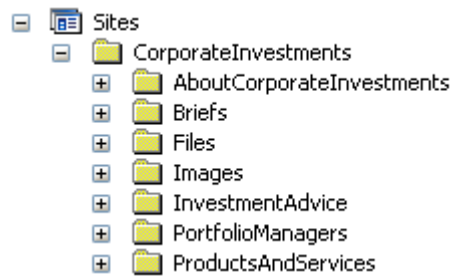


Figure 45: FastForward's Internet Site Folder Trees

- Two Workflows that reflect the business processes used by most customers.
- Communities that allow customers to assign users according to Sites and access levels.

CHAPTER 8

Physical Architecture, Deployment, and Scaling

The physical architecture of each Rhythmyx system depends on a variety of factors including the degree of security used and the location of the Web Server. The number of servers and the optional components included in each system depend on the system load, backup requirements, and business processes. This section offers general information and guidelines for planning the deployment and scaling of a Rhythmyx system.

Rhythmyx Physical Architecture

All Rhythmyx systems have the following physical architecture. Content contributors who are either local or remote access the Rhythmyx Server via HTTP (over Port 9992 by default). Rhythmyx must have access to a backend database and also to the Publisher Server, which may or may not reside on the same network as Rhythmyx. The Rhythmyx Server accesses the Publisher Server via HTTP (over Port 9992 by default) and the Rhythmyx Publisher accesses the Rhythmyx Server via SOAP (over Port 9980 by default). Content contributors can also access the Rhythmyx Server via HTTPS, and the Rhythmyx Server and Publisher can also communicate via HTTPS.

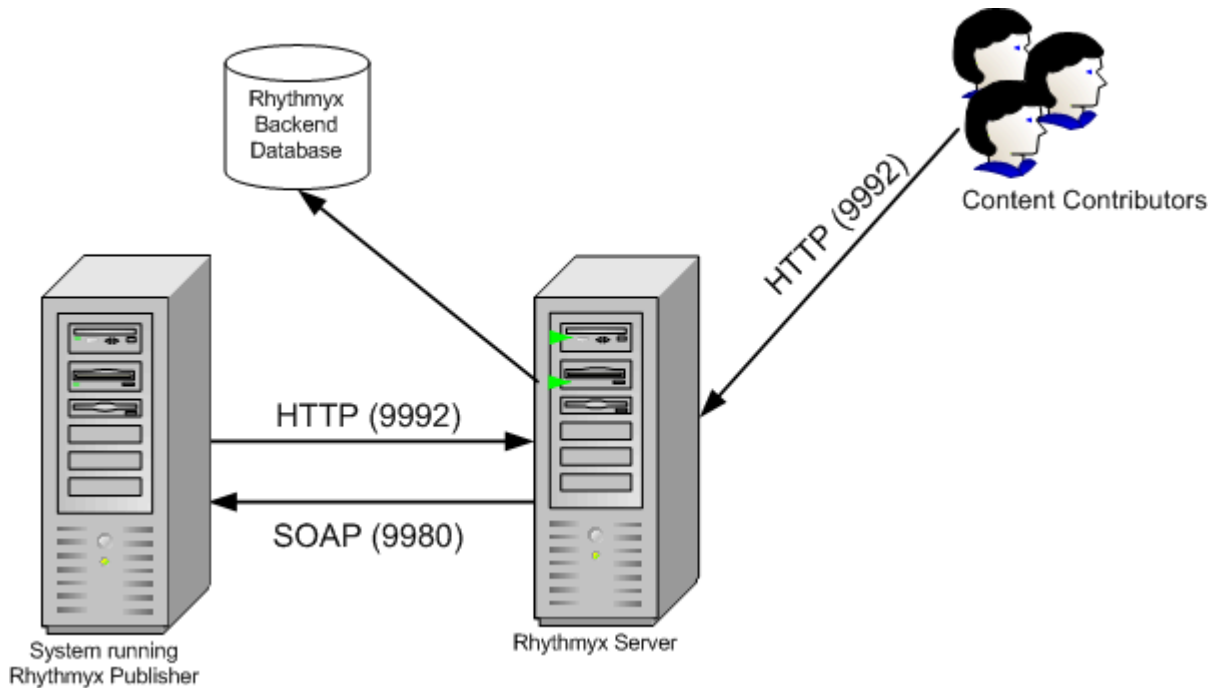


Figure 46: General architecture of Rhythmyx

The Rhythmyx Publisher may or may not be on the same machine as the Rhythmyx Server. In most cases, the Publisher either publishes locally to a file system or via FTP to a Web Server on a remote network. If the Publisher is on the same machine as the Web Server or the Web Server has access to the file system, the Publisher should use file system publishing. If the Publisher is on a different machine than the Web Server and does not have access to the file system, the Publisher should use FTP publishing.

Below are two diagrams showing these implementations. In the first diagram, the Web Server, Rhythmyx Publisher and Web Site are located on the same machine, so the Rhythmyx Publisher publishes directly to file system. In the second, the Rhythmyx Publisher is located on a different machine from the Web Server (and Web Site), so it publishes via FTP.

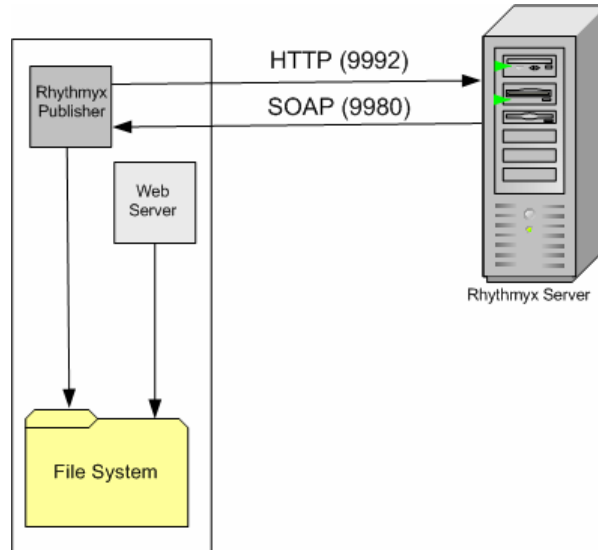


Figure 47: Architecture of Rhythmyx Publisher using file system publishing

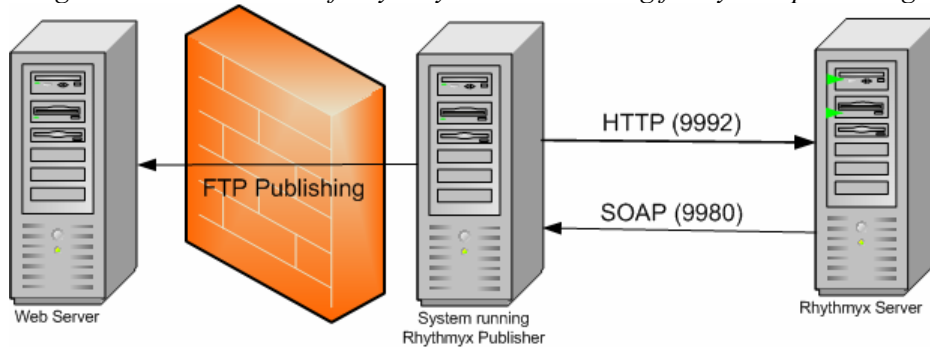


Figure 48: Architecture of Rhythmyx Publisher using FTP publishing

To ensure authenticity, a System Administrator may SSL-enable the Rhythmyx Server and Rhythmyx Publisher in any of these configurations. To increase security when publishing over a firewall, the Administrator may configure an SSH tunnel that runs from the Publisher to the Web Server. For information about configuring Rhythmyx Publishing with these options and others, see the document *Implementing Publishing in Rhythmyx*.

Deployment Scenarios

The optimal deployment scenario for a Rhythmyx system depends on the Rhythmyx components included in the system, the number of users working on the system, the level of integration of users' work, and other factors. This section reviews the elements affecting the way a system should be deployed, and suggests deployment models depending on the characteristics of the system.

Rhythmyx System Components

As stated earlier, the Rhythmyx system is made up of the same fundamental components regardless of the Modules or licensing options purchased. Rhythmyx components include Servers, servlets, end user clients, implementer clients and administrative clients. Client components deployment is straightforward and not covered in this section. Server components can provide for greater scaling, performance, and availability options, as well as more robust development and testing. This section will focus on the server components and the deployment options they provide for the system as a whole.

The available components include:

Server Components	Description	Notes
Rhythmyx Server (also referred to as: Production Server or System Master)	The fundamental processing engine of the CMS. Runs as a stand alone Server.	Requires a Repository. The search engine sub-component of the Rhythmyx Server may be installed as a separate server.
Repository	Defines a specific database schema for storing and managing system metadata for all content managed by Rhythmyx. May be installed in a variety of standard RDBMS systems, either local or remote to the Rhythmyx Server. The same repository may be shared by multiple Servers.	External data and content from outside this Repository may also be used by the CMS on a virtual basis.
Servlets and/or Application Server	A set of input/output oriented capabilities. These may be deployed in the native Rhythmyx Application Server, or as a set of servlets in a number of J2EE containers.	

Server Components	Description	Notes
Test Server	A Rhythmyx Server, restricted by license to be used for QA and for User Acceptance Testing of the CMS implementation.	Because of de-coupled delivery, a Test Server is not required to test content (such as Web Sites). The test server is only needed for testing CMS implementation design (such as forms, content types, workflow definitions, etc.)
Development Server	A Rhythmyx Server, restricted by license to be used for development of the CMS implementation.	
Publishing Hub	A Publishing Hub is a Rhythmyx Server licensed only for use in publishing activities initiated by the Publishing engine.	Organizations may include additional publishing hubs to publish higher volumes of pages with greater efficiency.
Rhythmyx Express Portal	A distinct Portal Server offered by Percussion. May be installed locally or remotely with the Rhythmyx Server. See <i>Rhythmyx Express Portal</i> (on page 53).	Includes Rhythmyx Portlets for contribution and support for publishing from Rhythmyx into the Rhythmyx Express Portal.
Publisher	The Publisher allows both formatted content and data to be published to any external site or application, using a variety of different storage mechanisms including files, database schemas, hybrid file-dbms systems and repositories with proprietary APIs.	Each Publisher enables greater security and administrative control over publishing content flow.
Enterprise Content Connector	The Rhythmyx Enterprise Content Connector uploads content from external sites into Rhythmyx Servers or file systems. Customers may use the Enterprise Content Connector to initially upload the contents of an entire Web site or to regularly upload content stored in a data asset management system.	

Multi-tiered Environment

In the Rhythmyx environment that Percussion recommends, implementers create new Rhythmyx elements and features on a development server, perform user testing on a testing server, and install tested, new functionality to a production server. In technical terms, this is called a multi-tiered environment and each server level represents a tier.

NOTE: During delivery, Rhythmyx content and its associated metadata are rolled out to one or more content delivery applications. The multi-tiered CMS environment should not be confused with the multiple tiers that may exist in each of the delivery channels. In the CMS tiers, CMS functionality, such as forms, workflow, and content type definitions are implemented and rolled out to production users of the CMS. This section will cover only deployment of CMS functionality using the multi-tiered Rhythmyx CMS.

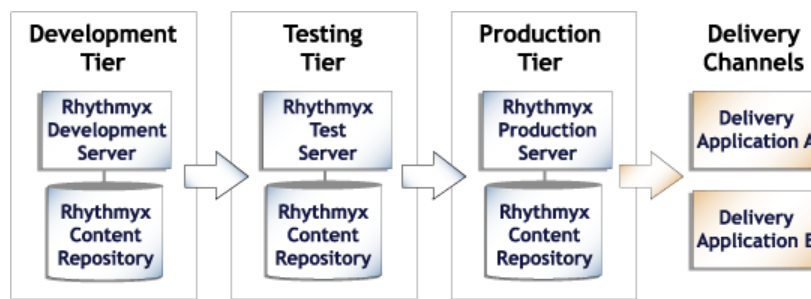


Figure 49: A multi-tiered environment includes development, testing and production tiers.

Development Tier

Developers create new functionality in the development tier. Unique tasks that developers perform on the development tier include:

- Developing Rhythmyx elements and applications such as Content Editors and Content Assemblers;
- Registering new Rhythmyx elements;
- Testing and fixing bugs in Rhythmyx elements;
- Publishing to development servers and local file systems;
- Installing Multi-Server Manager archives from production and integration servers to modify existing functionality and from other development servers to adapt other developers' functionality;
- Installing sample Content Items from production or integration tiers with the Multi-Server Manager for testing or bug fixing.

Testing Tier

Developers test new features and fix bugs on the integration tier. Other tasks include:

- User Acceptance Testing (UAT);
- Publishing to an integration testing Web server;

- Installing archives from the development tier for testing and integration and from the production tier for bug fixing.

Production Tier

Tasks on the production tier include:

- Content creation and approval;
- Publishing to a staging Web server (to test a production configuration);
- Publishing to a production Web server;
- Installing archives of tested new functionality from the testing tier.

Configuration Options for Development, Testing, and Production Tiers

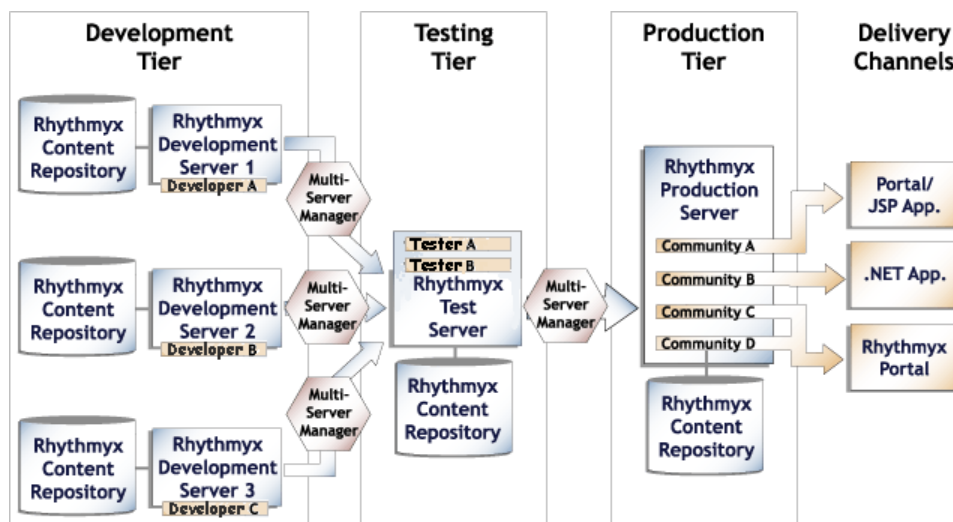
In the development tier, implementers should deploy separate Rhythmyx Servers with separate repositories if developers work on independent projects or are separated geographically. If developers work together on projects, implementers should deploy a single development server or a single development repository to allow them to view and integrate with one another's work. Put developers who are working on independent projects or who are geographically separated on different servers and repositories so they can avoid conflicting with other developers' work.

Depending on an organization's content-sharing requirements, implementers may deploy one shared Rhythmyx Server and repository or multiple, separate, Rhythmyx Servers and repositories on the testing tier. Testing on a single, shared Rhythmyx Server on this tier is recommended if users share a Rhythmyx Server that integrates all functionality in the production tier. Deploying multiple, separate instances of Rhythmyx on this tier can be beneficial if Rhythmyx Communities are separated onto different Rhythmyx instances with different features and different publishing destinations in the production tier. In many situations both Rhythmyx configurations are deployed in the integration tier.

Likewise, implementers can deploy a shared Rhythmyx Server or separate Rhythmyx Servers on the production tier. Configuration depends on whether or not Roles and Communities share content and applications and whether or not they always publish to the same or different Sites.

In many cases, tiers include both shared Rhythmyx Servers and separate Rhythmyx Servers that are used by individual developers or certain Communities or Roles.

The following graphics show two possible configuration scenarios for a multi-tiered environment; in each graphic, at least one tier includes both a separate and a shared Rhythmyx repository. In the first graphic, three developers work separately on three different Rhythmyx repositories and Servers, but deploy all of their new functionality to a single Rhythmyx instance for testing. Two testers work on the Rhythmyx testing server and deploy the tested components to a single production server and repository. Four different Communities share the production Server, although two of the Communities publish content to unique Sites.



In the second graphic, two developers work on individual Rhythmyx Servers and repositories and two developers share a third Server and repository. The functionality developed on each Server is deployed to a separate testing Server. Different users test the functionality on each testing Server and deploy the tested components to separate production Servers. Two Communities work on their own production Servers and repositories and two additional Communities share a third production Server and repository. Content created in each production Server is published to a different destination.

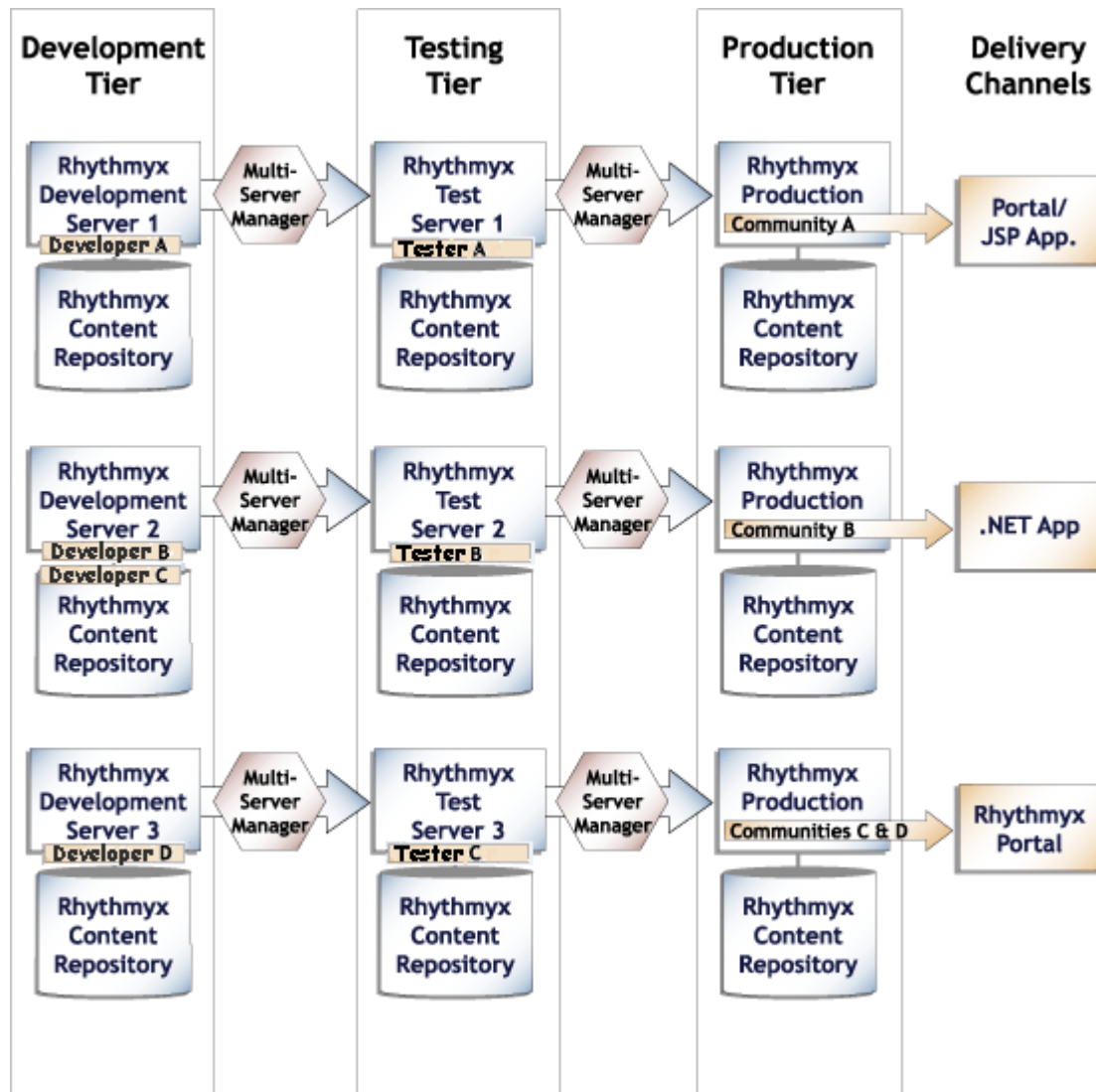


Figure 50: A decentralized Rhythmyx environment

Scaling the Rhythmyx Publishing Environment

After selecting a deployment approach, an Administrator must consider what number of Publishing Hubs will meet an organization's publishing requirements.

The number of Publishing Hubs required depends on the amount of content published to each target environment.

In the following graphic, the amount of content published requires two publishing hubs, each handling the entire volume published to one destination and sharing the task of publishing to another destination.

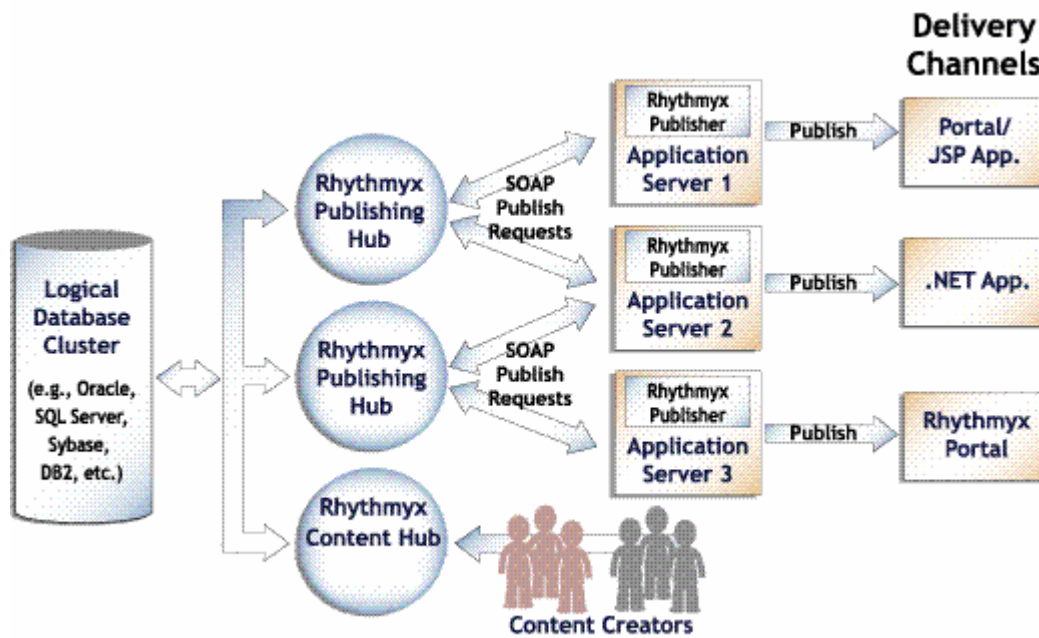


Figure 51: Scaling Rhythmyx Publishing Hubs

Percussion recommends scaling by increasing Publishing hubs because increased productivity in Rhythmyx causes greater CPU use rather than more network traffic. This architecture allows for the segmentation of the CPU cycles across multiple servers based on typical user activity and on the expected publishing requirements of multiple communities and multiple delivery locations.

Guidelines for Expanding a Rhythmyx System

A Rhythmyx Web Content Module or Document Management Module includes one Content Hub and one Publishing Hub. Depending on the factors discussed in this section, a company may initially choose to purchase additional publishing hubs, a test server, or a hot standby server, or may purchase them later, as its system develops and grows.

The following are guidelines for determining when to expand a Rhythmyx system:

Optional Component	Scaling Guidelines
Additional Publishing Hub	If the publishing runs in your incremental publishing cycle overlap each other (one publishing run begins before the previous one is finished) your system may begin to republish Content Items and waste resources. You may acquire an additional publishing hub to handle some of the publishing volume.
Database Publisher	If you store content in a database and deliver it dynamically, you should consider purchasing a Database Publisher when you purchase your Rhythmyx system. Otherwise, purchase a Database Publisher when you implement a database storage and delivery system for your Web site.
Test Server	<p>Access to development servers is restricted to users with Workstation licenses; therefore a test server is necessary if you want business users to test your changes before you move the changes to a production server.</p> <p>A test server is also useful if:</p> <ul style="list-style-type: none"> ▪ Your implementers need the ability to merge their development work on a common server. ▪ You want to make new functionality available to some users for testing, but not to others.
Hot Standby Server	If customer need or industry regulations require that you have a highly available and updated system, you may acquire hot standby servers to back up your Rhythmyx Server and other servers that you have added to your Rhythmyx system.

Tomcat Web Application Server

Rhythmyx runs in a Java 2 Enterprise Edition (J2EE) environment and can be installed with any J2EE Web application server. For those customers who do not wish to choose or administer a separate J2EE server, Rhythmyx installs the Apache Tomcat Web application server and provides management for that server as part of the installation. Tomcat offers several advantages over other Web application servers, including its open source status, its widespread use, and its technical capabilities. Tomcat functions as a Java servlet container that lets Rhythmyx run the Rhythmyx Publisher and Web Services and several of the Rhythmyx standalone applications as Java Servlets. In addition, Tomcat includes a Java Server Pages (JSP) engine which enables Rhythmyx users to deliver dynamic Web Pages with efficiency and flexibility.

Although Tomcat offers a number of benefits, some systems require features offered by other Web application servers. The Rhythmyx Publisher and Rhythmyx Web Services may be installed into any other J2EE Web application server, for example, BEA WebLogic or IBM WebSphere. In all cases, since the fundamental APIs for Rhythmyx are Web Services based, all application platforms are covered equally, including .Net, Java, and legacy Web environments.

Database

Rhythmyx stores unformatted content and metadata in third-party database applications. Most of the database features are managed through Rhythmyx while backup is handled separately by the database. A large number of functions may be accomplished through Rhythmyx interfaces, such as the Server Administrator, the Workbench, and the Multi-Server Manager. For example:

- entering backend server timeout and connection limits, setting up database users and passwords, and configuring backend table security providers.
- deploying table schemas and data from one backend database to another.
- creating backend tables that store local Content Editor data.

Each Rhythmyx Server may connect to its own backend database (or databases) or Rhythmyx Servers may share databases depending on the Servers' functions. See *Configuration Options for Development, Testing, and Production Tiers* (on page [60](#)).

Rhythmyx supports the use of the following RDBMS's for its backend repository:

Oracle 8i

Oracle 9i

Oracle 10g

Microsoft SQL Server 2000

Sybase 12.5+

IBM DB2 UDB 7.2

CHAPTER 9

Data Protection

Data protection in Rhythmyx includes a range of security functions that limit access to Rhythmyx Content Items, interfaces, and applications to users with the proper credentials. Since Rhythmyx content is stored in external database repositories, Rhythmyx does not include its own backup or archiving features; however Percussion recommends that customers carefully follow the procedures provided by their database systems.

Security

Rhythmyx provides various mechanisms for preventing users without the proper permissions from gaining access to Content Items and other Rhythmyx components.

At the most basic level, users must have a username and password to log in to Rhythmyx. The System Administrator first assigns these in a security provider, and then adds the user to Rhythmyx through the Rhythmyx Server Administrator.

Security providers are systems that maintain directories of usernames and credentials for the purpose of authenticating users who attempt to access a server or an individual application. By default, Rhythmyx uses a backend table security provider that validates usernames and passwords. Rhythmyx systems that access Rhythmyx through a Web application server, such as BEA WebLogic or IBM Websphere, may use a Web Server security provider which relies on an existing Web server or Web application server authentication.

Rhythmyx also uses Access Control Lists (ACLs) to define users' access levels to servers, applications, Content Explorer Folders and Content Items. The Rhythmyx Administrator defines a set of Roles with different privileges in the Server ACL and then assigns users to one or more Roles.

Although Communities and Locales can limit the content and user interface components that are available to users, they are not security mechanisms. Communities and Locales simplify what users see by filtering out information that is not relevant to them; however, they do not provide the security available through the server and ACL settings.

To prevent the loss of data that might occur if multiple users attempted to modify and save the same Content Item simultaneously, Rhythmyx includes a check in and check out feature that prevents more than one user from having edit access to a Content Item at the same time. A Content Item must be checked in for a user to have edit access to it. The user must check it out to edit it; checking it out locks it to all other users.

For more information, see the section “About Security” in the online help accessible through the Rhythmyx Server Administrator.

Backup

Backing up Rhythmyx includes implementing procedures for archiving data and backing up the server and database.

Archiving

All Rhythmyx content is stored in the database; therefore administrators can use standard database archiving procedures to meet content archiving requirements.

NOTE: When Content Items are removed from a Publish State in a Workflow, they are often transitioned to an Archive State. No intrinsic connection exists between Content Items that are in an Archive State and Content Items archived through database archiving.

Purging

Purging permanently destroys all records of a Content Item in the system. A company's retention rules may require that purge actions only be available for Content Items that have been stored through database archiving.

Backup Recommendations

Rhythmyx Server Backup

To back up the Rhythmyx Server, the Network Administrator can use any type of back up software or hardware to make a copy of the Rhythmyx Tree (the entire contents of the Rhythmyx installation root). Depending on the amount of server use and the significance of the data entered, the Network Administrator can choose appropriate backup software and hardware and determine an optimal back up schedule.

In most back up scenarios, the Rhythmyx development server is backed up more frequently than the production server because applications are modified more frequently in development. For example, an Administrator might schedule full weekly backups and partial nightly backups for an active development environment, but perform a backup of the production environment only when a change is introduced.

Database Backup

The Network Administrator determines the procedure and schedule for database backup based on the methods available for the specific database that the Rhythmyx system uses. Rhythmyx works with any database backup and recovery scheme. Most Rhythmyx implementations use a single database/schema, and the backup procedure depends on its physical storage configuration.

As a general rule development databases should have the same backup schedule as the corresponding development Server tree. In most cases, the production repository backup schedule depends on the frequency of content contribution to the system.

CHAPTER 10

System Requirements

Server Side

The Rhythmyx Server must be installed on one of the following supported Operating Systems. The size requirements for both the development and production tiers are noted:

- Sun Solaris 7, 8, 9

Development	Production
UltraSPARC 10 or higher	V120 or Higher recommended
RAM - 512 MB	RAM – 2 GB
Disk Space - 700 MB	Disk Space – 700 MB

- Microsoft Windows 2000 (Professional, Server, Advanced Server and Datacenter Server), Windows 2003 Server platform.

Development	Production
Processor - 1 Ghz+	Processor - 2 Ghz+
RAM - 512 MB+	RAM – 1+ GB
Disk Space - 700 MB	Disk Space – 700 MB

- Linux (Certified on Red Hat Linux AS 3.0+; Other distributions supported; Linux OS certified on Intel-based hardware.)

Development	Production
Processor - 1 Ghz	Processor - 2 Ghz
RAM - 512 MB	RAM - 1+ GB
Disk Space - 700 MB	Disk Space - 700 MB

Client Side

One of the following browsers must be installed to support content contribution, Workflow, Publishing, and System Administration:

- Internet Explorer V5.5+ (Windows)
- Mozilla Firefox (Windows OS; open source Esker plug-in required on desktop installations to support rich text editing control)
- Netscape Communicator V7.0+ (Windows OS; open source Esker plug-in required on desktop installations to support rich text editing control)
- Safari 1.21+ (Apple OS/X 10.3+)

In order to use the Rhythmyx Connector for Word, your system must have one of the following versions of Microsoft Office Client installed:

- Word 2003, 2002/XP, 2000
- Functionality limited in Word 97

The following Desktop Client Components are recommended for optimal use of all features:

- 1 Ghz or higher desktop machine recommended; 256mb ram or higher recommended.
- RAM - MIN 256 MB, 512 MB+ recommended
- Disk Space - MIN 200 MB to run installed product (500 MB during install)
- Sun Java browser plug-in v1.4 required; 1.4.2 or higher recommended; compatible with all supported browser platforms for users who access Content Explorer utilities.
- ActiveX control support required for browser-based rich text editing controls.

In order for implementers to use the Rhythmyx Workbench, one of the following supported environments must be installed:

- Windows Server 2003
- Windows XP Professional/2003
- Windows 2000

Publisher

In order to use the Rhythmyx Publisher, your system must meet the following requirements:

- Installation with a J2EE Web Application Server required. By default, the Tomcat J2EE Web Application Server is installed with Rhythmyx.
- Hardware Requirements:
 - RAM - MIN 128 MB, 512MB Recommended
 - Disk Space - MIN 30 MB to run installed product

Rhythmyx Repository

Rhythmyx stores its data in an external relational database. Your system must include one of the supported Databases:

- Oracle 8i
- Oracle 9i
- MS SQL Server 2000
- Sybase 12.5+
- DB2 UDB 7.2

Optional Rhythmyx Full Text Search Engine components:

Supported Operating Systems:

- Windows 2000 Server and higher
- Linux (Red Hat AS 3.0 certified; other distributions supported)
- Solaris 7 or higher

Hardware Requirements

- Additional 512MB ram over Rhythmyx Server, 1GB+ recommended depending on size of content repository
- 256MB disk space minimum. 1GB+ recommended depending on size of content repository

Portal Connectors

- Portal Connector for BEA WebLogic Portal
Supports BEA WebLogic Portal 8.1
- Portal Connector for IBM WebSphere Portal
Supports WebSphere Portal Version 4.2
- Portal Connector for Oracle Portal

Index

A

Active Assembly • 15
 Archiving • 97
 Assembly Engine • 14, 42

B

Backup • 97
 Backup Recommendations • 97
 BEA WebLogic Portal Connector • 74

C

Client Side • 100
 Clients and Interfaces • 9, 51
 Communities • 22, 23
 Configuration Options for Development, Testing, and Production Tiers • 88, 93
 Connectors • 9, 71
 Content Editors • 11, 14
 Content Engine • 11, 14, 26
 Content Explorer • 17, 61, 64
 Content Items • 11
 Content Management in Rhythmyx • 7
 Content Types • 11
 Convera Full-Text Search Engine • 9, 67

D

Data Protection • 9, 58, 95
 Database • 93
 Deployment Scenarios • 84
 Development Tier • 86
 Directory Services • 57
 Document Introduction • 5

E

Editions • 20
 Enterprise Content Connector • 72
 Extensions • 60

F

Folders • 18

G

Guidelines for Expanding a Rhythmyx System • 91

I

IBM Websphere Portal Connector • 75

L

Locales • 23
 Logging • 56

M

Monitor • 60
 Multi-Server Manager • 62
 Multi-tiered Environment • 86

O

Optional Rhythmyx Full Text Search Engine components • 101
 Oracle Portal Connector • 75

P

Physical Architecture, Deployment, and Scaling • 9, 67, 81
 Portal Connectors • 101
 Production Tier • 87
 Publisher • 100
 Publishing • 20
 Publishing Engine • 46
 Purging • 97

R

Relationship Engine • 16, 19, 35
 Relationships • 15, 19
 Rhythmyx and Item-based Content Management • 8
 Rhythmyx Concepts • 9, 11
 Rhythmyx Express Portal • 9, 76, 85
 Rhythmyx FastForward for Web Content Management • 9, 79
 Rhythmyx Logical Architecture and Processing • 9, 25
 Rhythmyx Modules • 9, 69
 Rhythmyx Physical Architecture • 82
 Rhythmyx Repository • 101
 Rhythmyx System Components • 84
 Roles • 22

S

- Scaling the Rhythmyx Publishing Environment • 90
- Search • 59
- Security • 58, 96
- Server Administrator • 55
- Server Side • 99
- Sessions • 23
- Settings • 56
- Sites • 11, 20
- Status • 55
- System Requirements • 9, 99

T

- Templates • 12
- Testing Tier • 86
- The Basics of Content Management • 6
- Tomcat Web Application Server • 92

U

- Using the Concepts Guide • 9

V

- Variants • 12, 13
- Views • 19

W

- Web Services API • 66
- Word Connector • 73
- Workbench • 52
- Workflow • 21
- Workflow Engine • 22, 39