

---

# **Rhythmyx Implementation Guide**

Version 6.5.2

## Copyright and Licensing Statement

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

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

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

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

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

Copyright © 1999-2007 Percussion Software.  
All rights reserved

## Licenses and Source Code

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

Netscape Public License

Apache Software License

IBM Public License

Lesser GNU Public License

## Other Copyrights

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

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

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

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

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

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

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

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

This product includes software developed by CDS Networks, Inc.

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

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

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

AuthorIT™ is a trademark of Optical Systems Corporation Ltd.

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

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

Schema documentation was created using XMLSpy™.

**Percussion Software**

600 Unicorn Park Drive

Woburn, MA 01801 U.S.A.

781.438.9900

Internet E-Mail: [technical\\_support@percussion.com](mailto:technical_support@percussion.com)

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



# Contents

---

## About the Rhythmyx Implementation Guide 7

Rhythmyx Implementation Roadmap .....	8
Implementation in the Rhythmyx Implementation Roadmap .....	10

---

## Accessing Rhythmyx Client Interfaces 11

Starting the Rhythmyx Workbench .....	12
Starting the Rhythmyx Server Administrator .....	14

---

## Setting Up the Development Infrastructure 15

Creating User Logins .....	17
Adding Users to the USERLOGIN Table Manually .....	17
Adding Users to the USERLOGIN Table Using a Script .....	19
Roles .....	20
Creating a Role .....	20
Adding Users to a Role .....	23
Communities .....	29
Creating a New Community .....	29
Workflows .....	31
Implementing the Simple Workflow .....	31
Implementing the Standard Workflow .....	47

---

## Setting up the Publishing Site and Basic Navigation 61

Creating the Site Root Folder .....	63
Registering the Publishing Site with Rhythmyx .....	65
Creating Site Subfolders .....	67
Managed Navigation for the Site .....	68
Adding a NavTree to the Site Hierarchy .....	68
Defining Access to Folders using Access Control Lists (ACLs) .....	70

---

## Creating Shared Fields 75

shared Field Set .....	77
sharedimage Field Set .....	78
The Rhythmyx Workbench Field and Field Set Editor .....	79
Creating Shared Field Sets and Configuring Fields .....	81
Implementing the "shared" Field Set .....	81
Implementing a List Control .....	90
Implementing the sharedimage Field Set .....	93
Field Visibility, Validation, and Transform Rules .....	99
Adding a Field Visibility Rule .....	100
Adding a Field Validation Rule .....	103

---

**Creating Slots and Templates** 107

---

Creating Slots.....	114
Creating a Standard Slot.....	115
Creating Templates.....	120
Preparing HTML for Use in Templates.....	121
Implementing Snippet Templates.....	122
Bindings.....	136
Implementing a Binary Template.....	140
Complex Snippets.....	143
Implementing Page Templates.....	149
Implementing Global Templates.....	163
Implementing a Page Template Without a Global Template.....	178
Dispatch Templates.....	181
Creating an Automated Slot.....	187
Creating a Simple Automated Slot.....	187
Automated Slots with Variable Parameters.....	192
Troubleshooting Templates.....	194
Property Not Found Error.....	194
Macro Rendered as Plain Text.....	195
Invalid Argument.....	196
Problem Assembling Output: Value is Badly Formed.....	197
Parameter Not Defined.....	198
Lexical Error.....	199
Velocity Code in Output.....	200
Illegal Argument Exception: Target Template May Not be Null.....	201
Problems Assembling Binary Outputs.....	201
Could Not Find Method <Name> for Object [null].....	203
Java.lang.RuntimeException: Could not find method <name> for object <bindingfunction>.....	205
Problem Parsing Expression.....	206
Java.lang.NullPointerException.....	207

---

**Creating Content Types** 209

---

Summary of Content Types.....	211
Generic Content Type.....	211
Image Content Type.....	211
Event Content Type.....	212
Basic Content Type Creation.....	214
Creating the Generic Content Type Object.....	215
Including Shared and System Fields.....	219
The Generic Content Editor.....	225
Viewing Generic Content Items.....	226
Image Content Type Creation.....	229
Creating the Image Content Type Object.....	230
Including a Local Field.....	231
Entering Content Editor Properties.....	233
The Image Content Editor.....	236
Viewing Image Content Items.....	238
WebImageFX.....	238
Creating a Content Type with a Child Field Set.....	240
Overriding a Shared Field.....	241
Including a Child Field Set.....	242

The Event Content Editor .....	246
Viewing Event Content Items.....	249
Item Transformation, Validation, and Pre- and Post-Processing .....	250
Adding Pre-processing Extensions .....	251
Implementing Text Extraction .....	254
Creating a Text Extraction Content Type.....	254

---

## Managed Navigation 261

How Managed Navigation Works .....	263
Maintaining Managed Navigation Content Items .....	267
Navigation Communities.....	267
Assigning a Landing Page to a Navon.....	267
Creating a NavImage.....	268
Splitting Navigation Sections .....	269
Merging Navigation Items.....	272
Reordering a Submenu .....	273
Creating and Using Keywords.....	273
Managed Navigation Slot .....	277
Customizing Navigation Look and Feel .....	278
Creating Managed Navigation Templates .....	278
Customizing Navigation CSS.....	286
Navon Properties .....	290

---

## Configuring Publishing 291

Publishing Specifications.....	294
Content Explorer's Publishing Tab .....	296
Defining Content Lists.....	297
Defining the Full Binary Content List.....	297
Defining the Full Non-Binary Content List.....	301
Defining the Incremental Content List .....	303
Defining the Unpublish Content List.....	305
Defining Contexts and Location Schemes .....	308
Creating the Publish Context and its Generic Location Scheme .....	309
Creating the Assembly Context and its Generic Location Scheme .....	315
Checking for Errors in the Location Scheme.....	319
Creating Editions .....	328
Full Publish Edition.....	330
Incremental Publish Edition .....	334
Testing your Content Lists .....	337
Testing Publishing of your Editions .....	339
Publishing the Full Publish Edition .....	339
Reviewing the Full Publish Log and Site .....	340
Publishing the Incremental Publish Edition.....	343
Reviewing the Incremental Publish Log and Site.....	344
Setting Up Publishing to a Local Web Server .....	346
Creating a New Site Registration to Deliver to a Web Server.....	346
Creating Context Variables for Delivery to a Web Server .....	348
Creating a New Edition to Publish to a Web Server.....	349
Setting Up FTP Publishing .....	351
Defining a Site Registration for FTP Publishing .....	353
Defining an FTP Content List Registration .....	355
Defining an Edition to Use the FTP Content List.....	356

---

Running the FTP Edition.....	357
<b>Database Publishing in Rhythmyx</b> .....	<b>359</b>
Special Components of Database Publishing.....	360
Database Publishing Specifications .....	361
Checking your Database Publishing Plugin Parameter.....	363
Steps for Database Publishing .....	364
Creating a Datasource for Your Target Database .....	365
Creating the Source Code for Your Database Template .....	368
Modifying the Table Definition File for Sequential Columns in Oracle .....	371
Registering the Database Publishing Site .....	373
Creating the Database Publishing Template .....	375
Registering the Database Publishing Context.....	383
Creating the Database Publishing Content List .....	385
Creating the Database Publishing Edition .....	387
Publishing to the Database.....	389
Reviewing the Database Publication .....	390
Unpublishing from the Database .....	393
Debugging Database Publishing .....	394
<b>Specialized Implementations</b> .....	<b>403</b>
<b>Next Steps</b> .....	<b>405</b>
<b>Appendices</b> .....	<b>407</b>
<b>Binding Variables</b> .....	<b>409</b>
System Variables .....	410
System Functions.....	414
\$rx.asmhelper .....	415
\$rx.codec .....	418
\$rx.cond .....	419
\$rx.db.....	420
\$rx.doc .....	420
\$rx.ext.....	421
\$rx.guid.....	422
\$rx.i18n.....	422
\$rx.keyword.....	422
\$rx.link .....	424
\$rx.location.....	425
\$rx.nav .....	427
\$rx.session .....	428
\$rx.string.....	428



Navon Properties .....	430
Database Publishing Variables .....	431
Velocity Tool Extensions.....	432
Assembly Items and Assembly Nodes.....	433

---

**Accessing Object Properties** **435**

---

**Naming Conventions** **437**

---

File and Application Naming Conventions.....	438
Design Object Naming Conventions .....	439
Project Prefix .....	439
Content Types.....	439
Templates .....	440
Slots.....	441
Communities .....	441
Workflows.....	442
Sites .....	442
Editions and Content Lists.....	442
Context Variables .....	442
Publishers .....	443

---

**FastForward Implementation Plan** **445**

---

Shared Field Sets .....	446
shared Field Set .....	446
sharedimage Field Set.....	447
Shared Field Sets .....	448
sharedBinary Field Set Specification.....	451
Content Types.....	452
rffAutoIndex Content Type Specification .....	452
rffBrief Content Type Specification .....	453
rffCalendar Content Type Specification .....	454
rffContacts Content Type Specification .....	455
rffEvent Content Type specification.....	457
rffExternalLink Content Type Specification .....	458
rffFile Content Type Specification .....	459
rffGeneric Content Type.....	460
rffGenericWord Content Type Specification.....	462
rffHome .....	463
rffImage Content Type specification .....	465
rffPressRelease .....	467
FastForward Workflows .....	469
Implementation Plan for Simple Workflow.....	470
Implementation Plan for Standard Workflow.....	473
FastForward Publishing Configurations .....	478
FastForward Sites .....	478
FastForward Content Lists.....	478
FastForward Item Filters .....	482

Content Editor System Definition	485
----------------------------------	-----

---

Index	489
-------	-----

---

## CHAPTER 1

# About the Rhythmyx Implementation Guide

The Rhythmyx Implementation Guide provides instructions for implementers developing the Rhythmyx design objects specified in the development plan by illustrating the implementation of selected components of the Enterprise Investments Site included in the FastForward reference implementation.

*Setting Up the Development Infrastructure* (see page 15), shows how to set up Roles, Communities, and Workflows, as well as how to add users to a default Security Provider that can be used in your development environment. For a detailed discussion of Security Providers for the production environment, see "Implementing Security in the Production Environment" in *Setting Up the Rhythmyx Production Environment*.

*Setting Up a Publishing Site and Basic Navigation* (see page 61), describes the process of defining a Publishing Site in Content Explorer, including the Folder hierarchy for your Site. We also show how to add basic Managed Navigation components to the Publishing Site.

*Creating Shared Fields* (on page 75), describes the process of creating Fieldsets that can be shared by multiple Content Editors. This chapter illustrates the implementation of several different fields to demonstrate the rich variety of options available for implementing a Content Type field.

*Creating Slots and Templates* (see page 107), shows the implementation of both Standard and Automated Slots, Global Templates, and a variety of Local Templates.

*Creating Content Types* (see page 209) illustrates how to add fields specific to a Content Editor as well as how to include Shared Fields but the emphasis of this chapter is a description of the editor-level options that can be implemented (as opposed to the field-level options illustrated in detail in *Creating Shared Fields* (on page 75)).

*Managed Navigation* (on page 261) describes how to implement the infrastructure required to publish Managed Navigation successfully, with a special emphasis on Managed Navigation Templates.

*Configuring Publishing* (see page 291) describes how to set up the system infrastructure to publish content to a Web server and how to configure publishing to a database.

---

# Rhythmyx Implementation Roadmap

The Rhythmyx implementation roadmap follows. You may find that performing some of these steps in a different order better serves the function of your system. You will also find yourself returning to steps that you have already completed because it has become clear that you must revise some of the components that you have designed.

Steps in the implementation roadmap:

- 1** Model and design your Web Site and the components that will make up your Rhythmyx CMS. Create a development plan that implementers can follow when designing these components. Most of the remaining steps instruct you to create the components designed and outlined during this process.
- 2** Configure the *Roles* (see page 20), *Communities* (see page 29), *Workflows* (on page 31), and *users* (see "Creating User Logins" on page 17) sketched out during modelling and design. As you continue the implementation process, you will see changes that you want to make.
- 3** *Set up the basic framework for your Site Folders and navigation hierarchy* (see "Setting up the Publishing Site and Basic Navigation" on page 61). The Site Folder structure may not be established during modelling and design; you may begin to determine it at this time, and will note changes that you want to make as your implementation proceeds.
- 4** *Create your shared fields* (see "Creating Shared Fields" on page 75).
- 5** *Create your Slots* (see "Creating Slots" on page 114).
- 6** *Create your Global template* (see "Implementing Global Templates" on page 163).
- 7** *Create your Content Types* (see page 209), either by modifying existing FastForward Content Types or by creating new ones.
- 8** *Create your local and shared Templates* (see page 120).
- 9** *Completing the set up of your Site Folders and navigation hierarchy* (see "Managed Navigation" on page 261).
- 10** Modify the configuration of your *Roles* (see page 20), *Communities* (see page 29), *Workflows* (on page 31), and *users* (see "Creating User Logins" on page 17) according to any necessary changes that you have noted during implementation.
- 11** Configure site folder publishing.
- 12** Deploy your Rhythmyx components to your integration environment, and, after testing, to your production environment.

The implementation roadmap will be represented by the following graphic at the section or chapter that begins each step. The road map will indicate which step you have reached in the process.

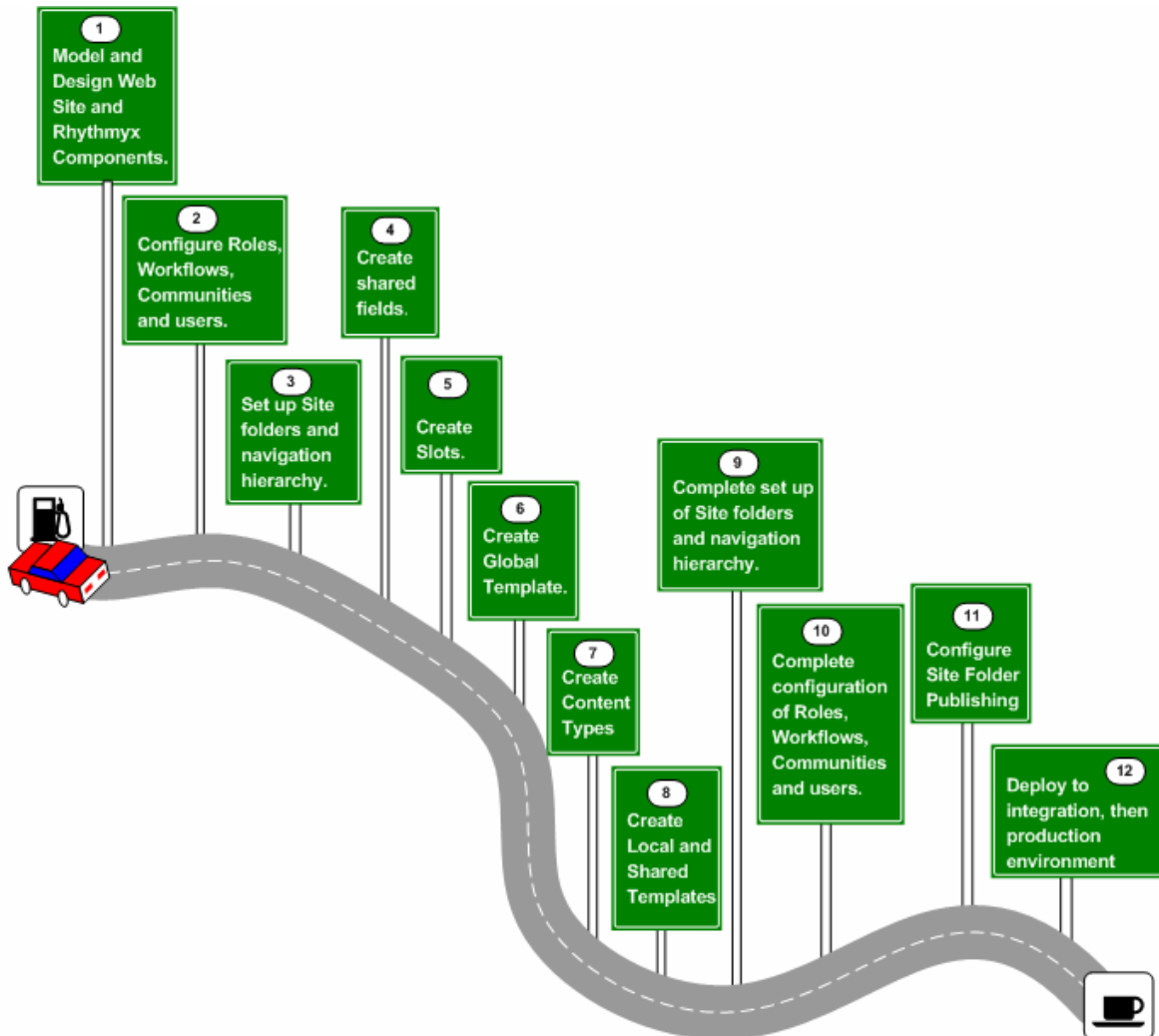


Figure 1: Rhythmyx Implementation Roadmap

---

# Implementation in the Rhythmyx Implementation Roadmap

This document is part of the Rhythmyx development library, including:

- *Getting Started with Rhythmyx*
- *Modeling and Design of a Rhythmyx Content Management System*
- *Setting Up the Rhythmyx Production Environment*

A variety of documents is also available addressing specialized implementation issues.

Implementation should not occur until you have become familiar with Rhythmyx and completed modeling and design of your implementation. The result of modeling and design should be a development plan specifying the Rhythmyx objects in your implementation and how they interact.

Before beginning implementation, you should complete the following tasks:

- Read the *Rhythmyx Concepts Guide*.  
This document introduces and explains the basic concepts of Rhythmyx and of Content Management using Rhythmyx. You should read at least the portions of the *Rhythmyx Concepts Guide* recommended for implementers.
- Read *Getting Started with Rhythmyx*.  
This document guides you through a basic installation of Rhythmyx with the FastForward implementation and includes some basic tutorial exercises to help you learn more about Rhythmyx and how it works.
- Attend training on Rhythmyx.  
Percussion Software provides training on Rhythmyx frequently throughout the year. Training will provide more opportunities to become familiar with Rhythmyx and the implementation process.
- Read *Modeling and Design of a Rhythmyx Content Management System*.  
This document outlines an example modeling and design process.
- Complete a development plan.  
This document provides the specification for the Rhythmyx design objects to implement for your system.

## CHAPTER 2

# Accessing Rhythmyx Client Interfaces

During implementation, the following Rhythmyx client interfaces are used:

- Rhythmyx Workbench
- Rhythmyx Server Administrator

These clients are part of the Rhythmyx Developer Tools. Rhythmyx Developer Tools are certified only on Microsoft Windows operating systems. Use of these clients on other operating systems is not supported.

## Starting the Rhythmyx Workbench

To start the Rhythmyx Workbench:

- In the Percussion Rhythmyx Program Group on your desktop, choose *Rhythmyx Workbench*;  
or
- Open Windows Explorer, browse to your Rhythmyx installation directory and double-click on *RhythmyxWorkbench.exe*.

When the Rhythmyx Workbench starts, it displays the Manage Connections dialog with a default connection configuration. You must enter the connection data to connect to your server:

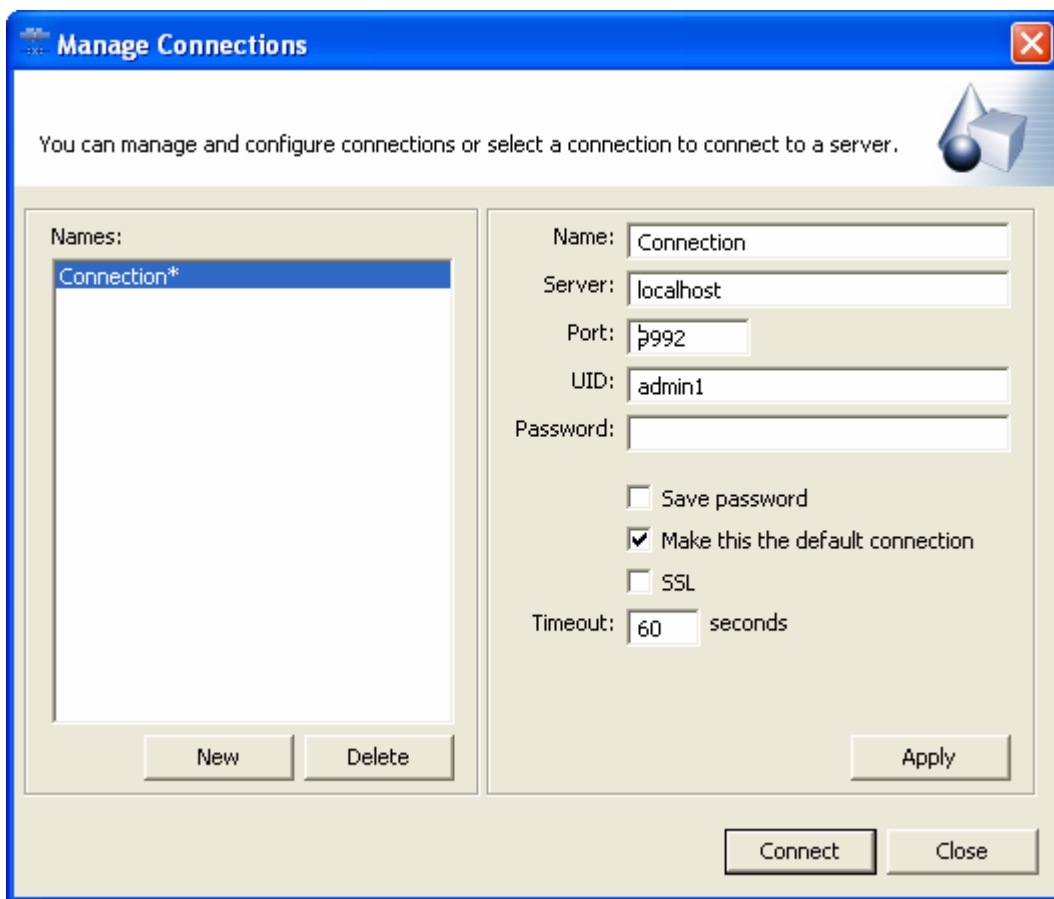


Figure 2: Connections dialog

- 1 The **Name** field specifies the name of the connection configuration. You can save named configurations so you can re-use them quickly. The default name of the default connection configuration is *Connection*. You can change this value to any alphanumeric string.
- 2 The **Server** field specifies the name or IP address where the Rhythmyx server for the connection resides. The default value is *localhost*. If you are connecting to a Rhythmyx server on a remote machine, change this value to the name or IP address of that machine.



- 3 The **Port** field specifies the port of the Rhythmyx server for the connection. The default value is *9992*, the default Rhythmyx port. Change this value to the correct port.
- 4 The **UID** field specifies the user ID used the connection uses to log in to the Rhythmyx server. The default value is *admin1*. Change this value to your Rhythmyx user name.
- 5 The **Password** field specifies the password used to log in to the Rhythmyx server with the specified ID. Enter the password for the ID specified in the UID field.
- 6 The **Save Password** checkbox specifies whether the password entered with the connection configuration will be saved with the configuration. If you check this box, the password will be saved. If you also check the **Make this the default connection** checkbox, the Rhythmyx Workbench will automatically attempt to log in Rhythmyx server specified in the connection configuration using the specified ID and password. If you do not check this box, you must enter the password and click the [**Connect**] button to connect to the Rhythmyx server.
- 7 The **Make this the default connection** checkbox specifies that the configuration is the default connection configuration. When you initially start the Rhythmyx Workbench, the Manage Connections dialog displays the default connection configuration. If you have also checked the Save Password box, the Workbench also automatically attempts to log in to the Rhythmyx server specified in the configuration using specified ID and password.
- 8 The **SSL** checkbox specifies that the Workbench communicates with the server over the Secure Socket Layer using secure HTTP (HTTPS). SSL must be enabled on the server if you enable this option. If SSL is not enabled on the server, the connection will fail. For details about enabling SSL on the Rhythmyx server, see "Enabling SSL on the Rhythmyx Server" in *Setting Up the Rhythmyx Production Environment*.
- 9 The **Timeout** field specifies how long the Workbench will attempt to establish a connection to the specified server before stopping and reporting a connection error. You can change this value if you would like a different timeout period.
- 10 Click the [**Apply**] button to save the connection configuration.
- 11 Click the [**Connect**] button to connect to the Rhythmyx server.

To create a new connection configuration, click the [**New**] button, enter the connection configuration data, and click the [**Apply**] button to save the configuration.

Connection configurations are listed by name in the Name field. Select the Configuration you want to use to connect to the Rhythmyx server.

For details about specific Rhythmyx Workbench functionality, see the Rhythmyx Workbench online Help.

---

## Starting the Rhythmyx Server Administrator

To start the Rhythmyx Server Administrator:

- In the Percussion Rhythmyx Program Group on your desktop, choose *Rhythmyx Server Administrator*; or
- Open Windows Explorer, browse to your Rhythmyx installation directory and double-click on *RhythmyxServerAdminsrator.exe*.

When the Rhythmyx Server Administrator starts, it displays the Login dialog with default data.



Figure 3: Server Administrator Log In Dialog

- 1 The value in the **Server** field defaults to the name of the machine on which the Rhythmyx Server Administrator is installed. Change this value to the name or IP address of the machine where the Rhythmyx server resides.
- 2 The value in the **Port** field defaults to 9992, the default Rhythmyx port. Change this value to the port of the Rhythmyx server to which you want to connect.
- 3 The value in the **User name** field defaults to *admin1*. Change this value to the User ID you want to use to log in to the Rhythmyx server. This user must have administrative rights on the Rhythmyx server or login will fail.
- 4 Enter the Password of the ID specified in the User name field.
- 5 If you want to connect to the Rhythmyx server over the Secure Socket Layer using secure HTTP (HTTPS), check the **Use SSL** box. SSL must be enabled on the server if you enable this option. If SSL is not enabled on the server, the connection will fail. For details about enabling SSL on the Rhythmyx server, see "Enabling SSL on the Rhythmyx Server" in *Setting Up the Rhythmyx Production Environment*.
- 6 Click the **Login** button to connect to the Rhythmyx server.

## CHAPTER 3

# Setting Up the Development Infrastructure



The most logical way to start your implementation is by creating the user logins, Roles, Communities, and Workflows required as part of both your development environment and your Rhythmyx implementation. These elements provide a foundation for the rest of your development process. The development infrastructure consists of:

- User logins, which allow you to log in to Rhythmyx with the proper access;
- Roles, which allow you to group users that require the same access to parts of the Rhythmyx implementation, including Sites, Communities, and Workflows.
- Communities, which allow you to filter access to specific content for different users. As part of the process of creating other design elements of your system, you will assign the design elements to Communities, so you should have the Communities available before creating the design elements.
- Workflows, which define the business processes for creating, maintaining, and publishing content. You must specify the Workflows available to Content Editors when designing Content Editors later; creating Workflows now streamlines your development process.

You do not have to create all the users, Roles, Communities, and Workflows you might need right now, but you need to create a minimum set of these elements that allows you to begin your work. Rhythmyx includes a number of pre-defined elements as part of the FastForward Sites Enterprise Investments (EI) and Corporate Investments (CI). You should have already determined in your modeling and design work which of those elements you can use as they are and which elements you need to modify or create from scratch. Normally, later in the implementation process, you would revisit this step and complete the configuration of your Roles, Workflows, Communities and users. Therefore, on the *implementation roadmap graphic* (see page 8) shown in this document, Step 10 is "Complete configuration of Roles, Workflows, Communities, and users".

---

Note: Throughout this guide, we illustrate procedures by showing how selected elements of the Enterprise Investments (EI) Site were created. The EI elements already exist on your server if you installed FastForward, so don't attempt to save your work if you enter the data to create these examples. The examples are only intended to provide a model to follow as you enter the data for your own Site.

For descriptions of the type of data required in each field on the dialogs, see the online help.

---

The following sections describe how to:

- 1 ***Create user logins for your Rhythmyx system*** (see "Creating User Logins" on page 17).
- 2 ***Create the Roles you need to get started with your implementation*** (see "Creating a Role" on page 20).
- 3 ***Add users to Roles*** (see "Adding Users to a Role" on page 23).
- 4 ***Create a Community for your Site*** (see "Creating a New Community" on page 29).
- 5 ***Create a Workflow, defining the process for approval and publication of content in your Site*** (see "Workflows" on page 31).

---

## Creating User Logins

Rhythmyx is installed with default user login data used in demonstrations and training. When developing your own implementation, however, best practice is to create your own logins. You should not use the default logins and passwords installed with Rhythmyx in any of your working environments. For details about the potential security problems posed by this default login data, see "Important Security Considerations" in *Installing Rhythmyx*.

Rhythmyx provides several Security Providers to list and authenticate users into Rhythmyx. The default Security Provider is the rxmaster Backend Table Security Provider. This Security Provider stores the user names and passwords in a database table. This Security Provider is adequate for development environments but should not be used in production environments because the passwords are stored unencrypted.

---

NOTE: For details about implementing other Security Providers, see the Online Help for the Rhythmyx Server Administrator.

---

The rxmaster Security Provider stores the authentication data in the table USERLOGIN. This table contains two columns: USERID and PASSWORD. Add authentication data for each developer and each system administrator that will interact with Rhythmyx during development.

### Adding Users to the USERLOGIN Table Manually

To add users to the USERLOGIN table manually, use the Enterprise Manager tool for your RDBMS. In this procedure, we will use MS SQL Server Enterprise Manager to add the following users:

USERNAME	PASSWORD
Lisa Kerr	lisakerr
Ed Wong	edwong
Rita Perez	ritaperez

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To add users to the USERLOGIN table:

- 1 Open Microsoft SQL Server Enterprise Manager.

- 2 Expand the rxmaster database on the development server and click on Tables to display a list of tables in the database. The tables are listed alphabetically, so USERLOGIN is near the bottom of the list.

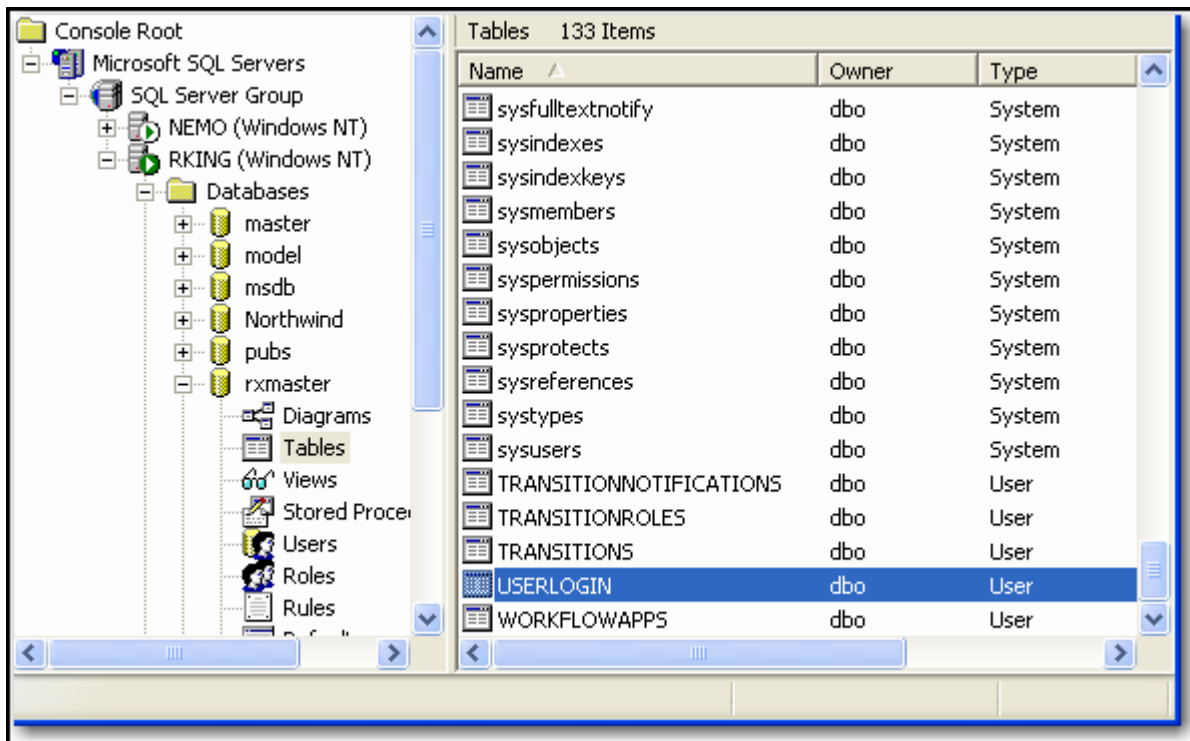
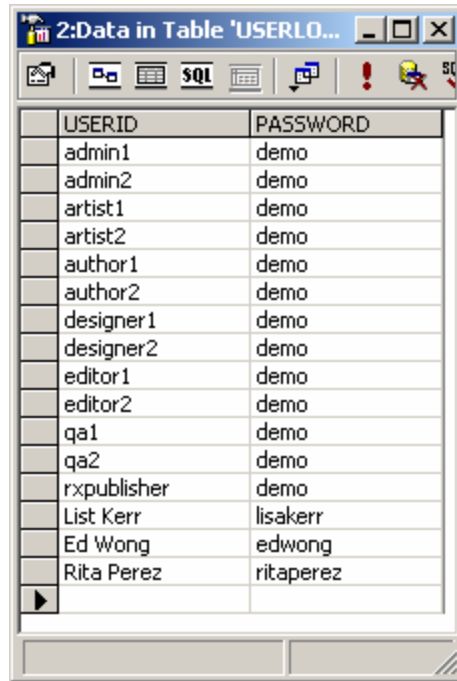


Figure 4: USERLOGIN Table

- 3 Right-click on the USERLOGIN table and from the popup menu choose *Open Table > Return all rows*.
- 4 Enterprise Manager displays the USERLOGIN table with its current data.
- 5 Click in the USERID column of the last row and enter *Lisa Kerr*; the click in the PASSWORD column of the same row and enter *lisakerr*. Press the return key on your keyboard to save the new entry.

- 6 Repeat Step 5 to enter the authentication data for Ed Wong and Rita Perez.



The screenshot shows a window titled '2:Data in Table 'USERLO...'. The window contains a table with two columns: 'USERID' and 'PASSWORD'. The data rows are as follows:

USERID	PASSWORD
admin1	demo
admin2	demo
artist1	demo
artist2	demo
author1	demo
author2	demo
designer1	demo
designer2	demo
editor1	demo
editor2	demo
qa1	demo
qa2	demo
rxpublisher	demo
List Kerr	lisakerr
Ed Wong	edwong
Rita Perez	ritaperez

Figure 5: Adding Users to the USERLOGIN Table

## Adding Users to the USERLOGIN Table Using a Script

To add users to the USERLOGIN table with a script, devise a script to insert the user names into the USERNAME column and the passwords into the PASSWORD column. For example, to add the following example authentication data:

USERNAME	PASSWORD
Lisa Kerr	lisakerr
Ed Wong	edwong
Rita Perez	ritaperez

The script would resemble the following code:

```
insert into USERLOGIN (USERNAME, PASSWORD) values Lisa Kerr, lisakerr
insert into USERLOGIN (USERNAME, PASSWORD) values Ed Wong, edwong
insert into USERLOGIN (USERNAME, PASSWORD) values Rita Perez, ritaperez
```

Then run the script in your RDBMS.

## Roles

A Role is a collection of users with the same access to certain elements in a Rhythmyx implementation, such as particular Sites, Communities, and Workflows. Grouping users into Roles helps administrators more easily manage users that have the same permissions. Instead of managing the permissions for each user, the administrator defines permissions for a Role, then assign users to the Role. Users inherit their access rights from the Roles to which they are assigned.

All Roles in Rhythmyx are created on the server using the Server Administrator. The same procedure is used to create all Roles regardless of how they are used.

Rhythmyx uses Roles in two ways:

- In Workflows, Roles specify which users can access content at specific points, or States, in the Workflow. Roles used in Workflow typically have names that describe the functions that the users in the Role typically perform in the Workflow, such as Author, Editor, or Administrator. After creating a Role in the server, you must add it to the Workflow before you can assign it to a State. For details about adding a Role to a Workflow, see *Adding a Role to a Workflow* (on page 35) for additional details.
- In Communities, Roles define the users that belong to the Community. All users that belong to a Community are Members of the Role assigned to that Community. Community Roles typically have names that indicate the Community with which they are associated, such as EI\_Members or CI\_Members. Once the user is assigned to the Community Role, they can access the design elements associated with that Community.

Note that in nearly every case, a Rhythmyx user will be a member of at least two Roles: a Community Role and one or more Workflow Roles.

## Creating a Role

To illustrate the process of creating a Role, we will create the EI\_Members Role. Later, we will assign this Role to the Enterprise Investments Community to grant users access to that Community.

We will define the property `sys_defaultcommunity` for this Role, with a value of *Enterprise Investments*. This property ensures that users will be logged in to the Enterprise Investments Community when they log in using this Role. Note that we will create the Enterprise Investments Community in a future procedure.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

- 1 Start the Rhythmyx Server Administrator. For instructions, see *Starting the Rhythmyx Server Administrator* (on page 14).



- 2 Click the Security tab along the top of the Server Administrator dialog and then click the Roles tab along the bottom. The Server Administrator displays the Roles tab.

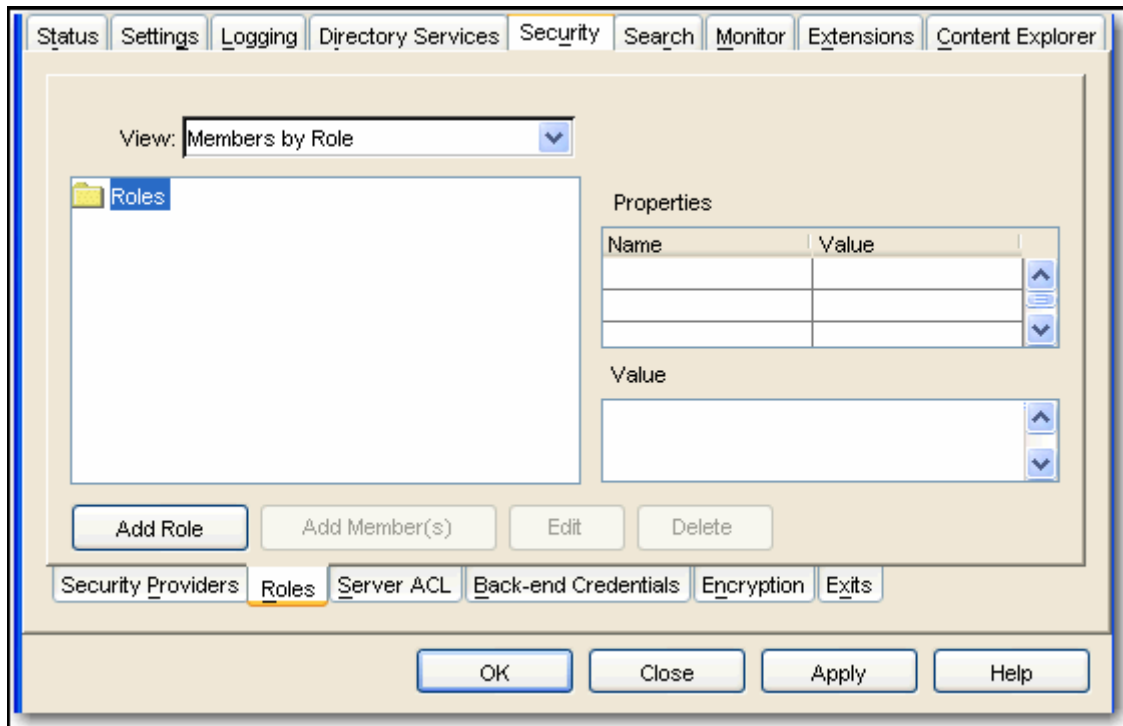


Figure 6: Server Administrator Roles Tab

- 3 Click the [Add Role] button to display the New Role dialog.

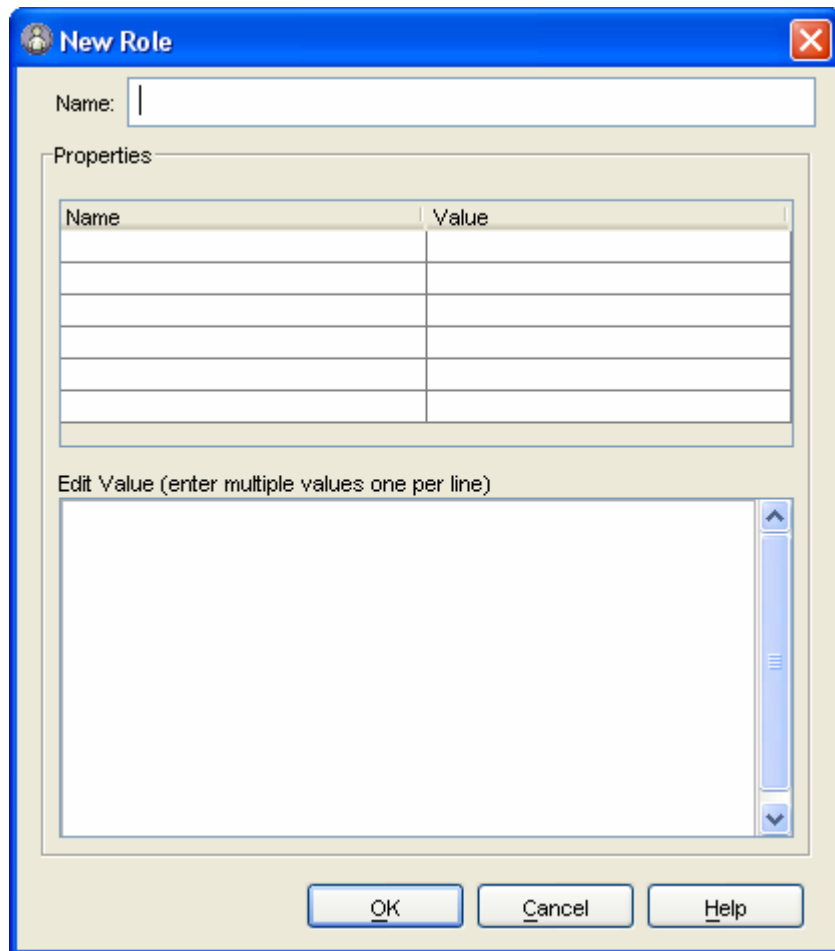


Figure 7: New Role Dialog

- 4 In the Name field, enter *EI\_Members*.

- 5 In the Properties box, click in the first row of the Name column and from the drop list, choose *sys\_defaultCommunity*. In the Value column of the same row, enter *Enterprise Investments*.

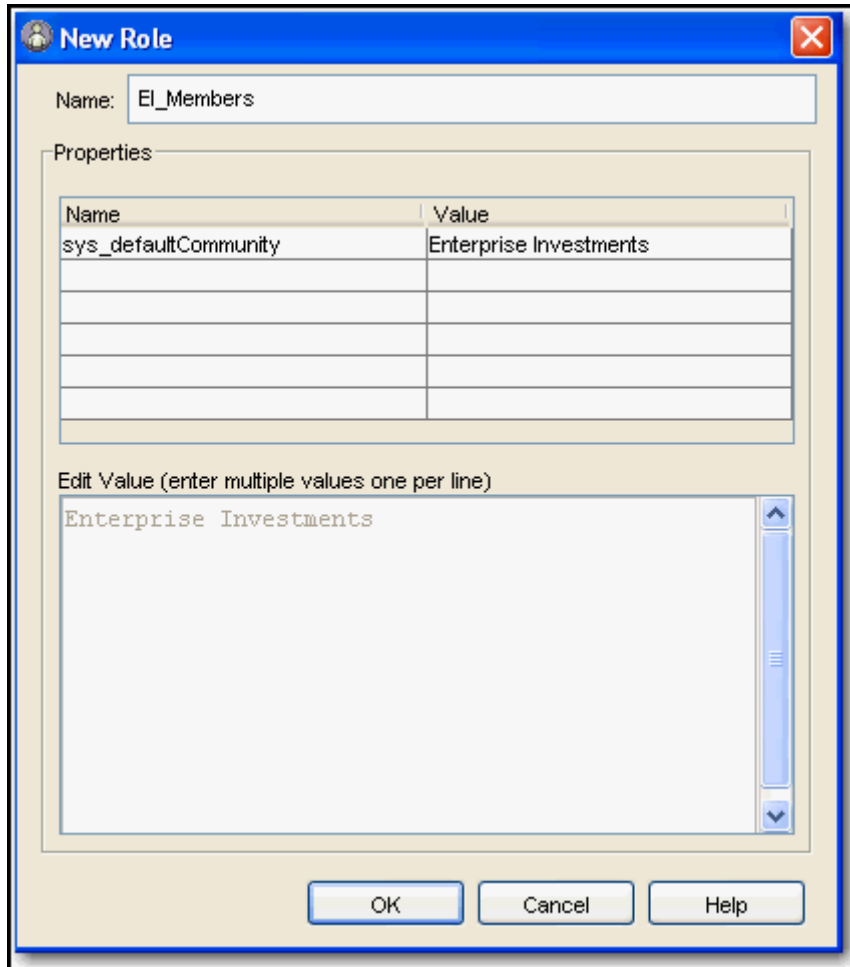


Figure 8: Creating the EI\_Members Role

- 6 Click the [OK] button.

Rhythmyx creates the EI\_Members Role and returns you to the Server Administrator.

## Adding Users to a Role

You must add users to a Role before they can log in to Rhythmyx using that Role. The users in a Role are referred to as Members of the Role. We need to add the users we created in *Creating User Logins* (see page 17) (Lisa Kerr, Ed Wong, and Rita Perez) to the EI\_Members Role we created in *Creating a Role* (see page 20).

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To add the Members to the Role:

- 1 Open the Rhythmyx Server Administrator, click the Security tab along the top, and then click the Roles tab along the bottom. The Server Administrator displays the Roles tab dialog.

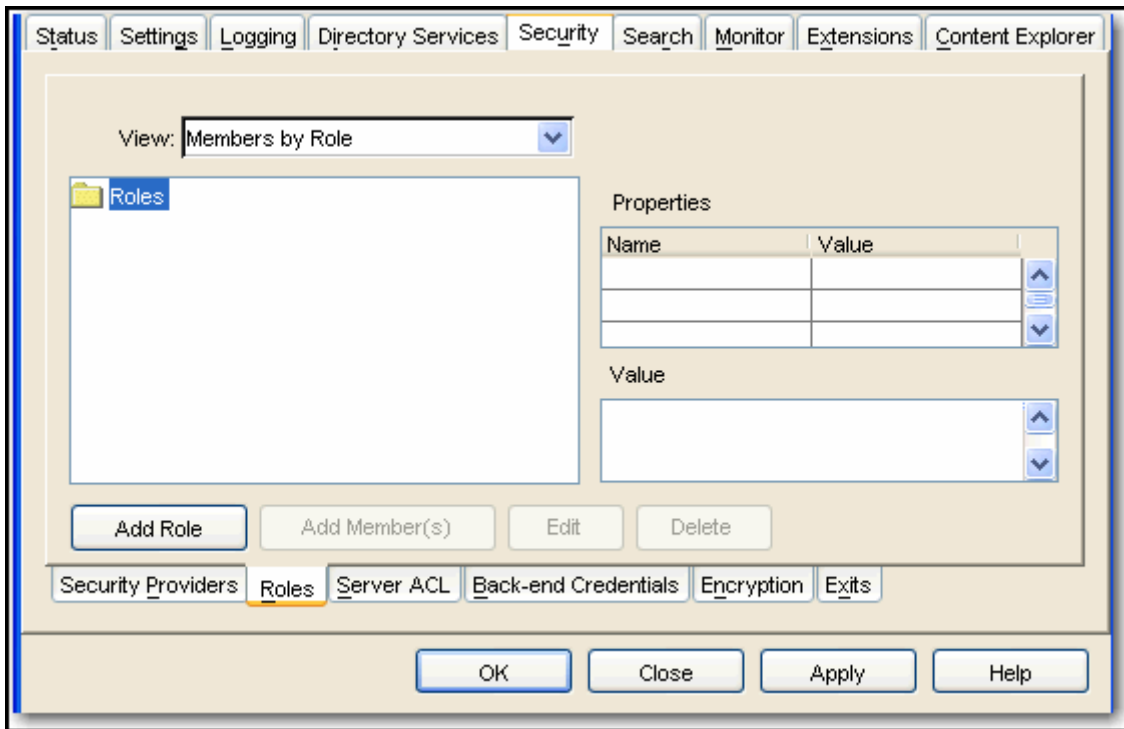


Figure 9: Server Administrator Roles Tab

- 2 Open the Roles folder (under the View: Members by Role drop list) and select the EI\_Members Role. The Server Administrator displays a list of users that are Members of the EI\_Members Role.

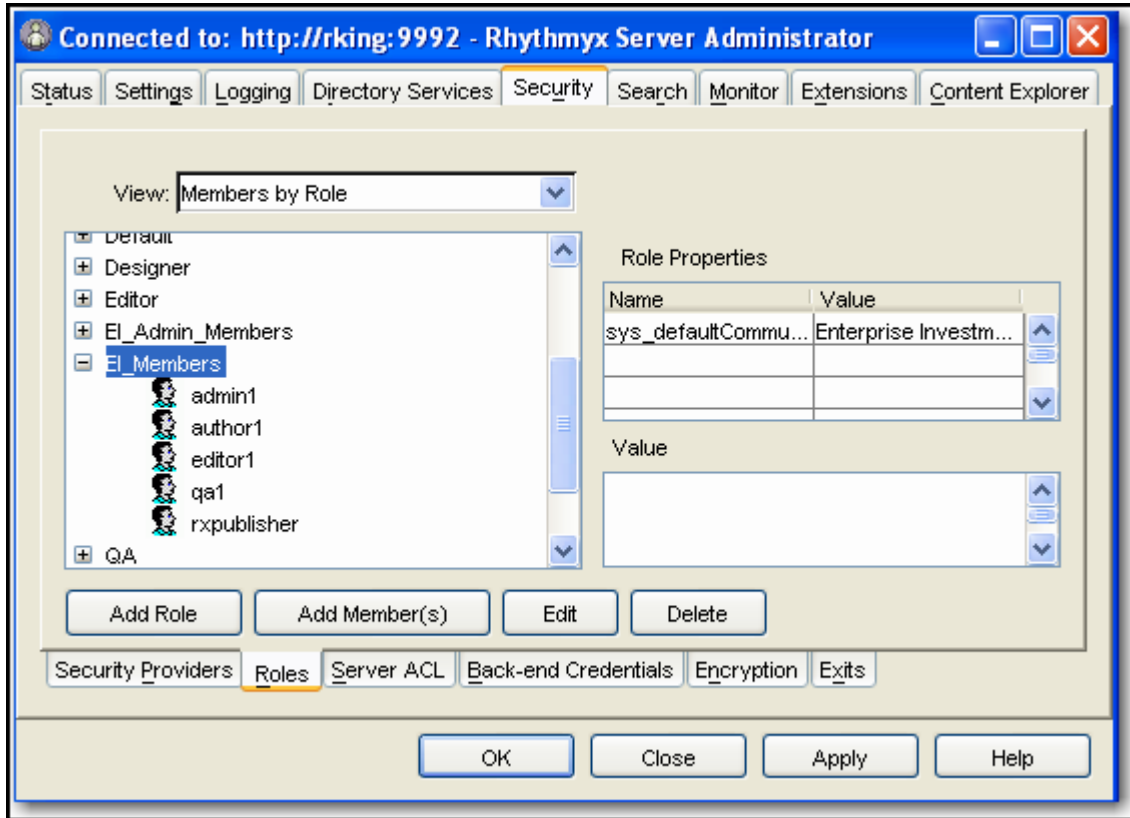


Figure 10: Viewing Members by Role

- 3 Click the Add Member(s) button. The Server Administrator displays the "Modify member list for: EI Members" dialog.

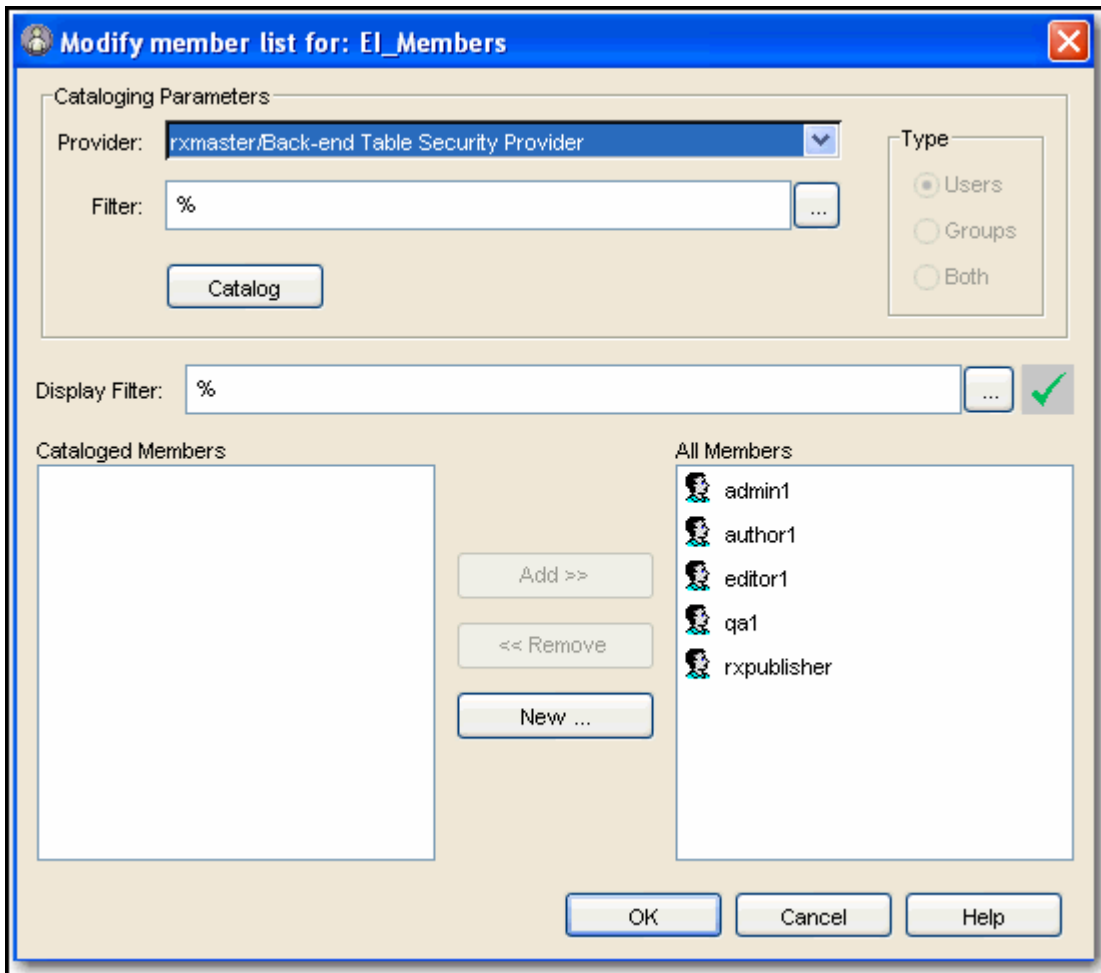


Figure 11: Modify Member List for: EI\_Members Dialog

You can retrieve a list of members available from each Security Provider. This function, known as cataloging, makes it easier to add new Members to the Role. Note that you can also add new Members manually, but cataloging is generally more efficient and less error prone, and ensures that the Members are associated with a Security Provider.

- 4 To catalog Members:
- a) In the Provider drop list, choose the default Security Provider, rxmaster/Back-End Table. **Members were added to this Security Provider when user logins were created** (see "Creating User Logins" on page 17).
  - b) Click the [Catalog] button.

In the Cataloged Members field the Server Administrator displays a list of users cataloged

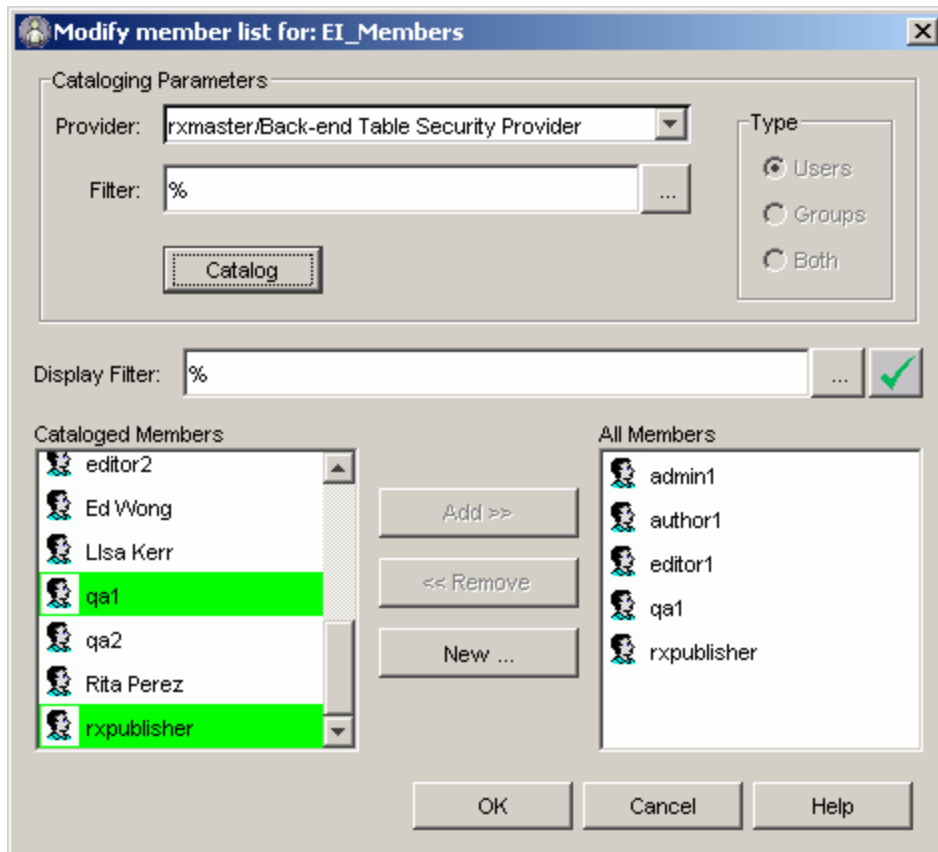


Figure 12: Modify Members dialog showing the users in the rxmaster/Back End Table Security Provider cataloged.

- To add Ed Wong, Lisa Kerr, and Rita Perez to the EI\_Members Role, select them in the Cataloged Members field (press the CTRL key while selecting Members in this field to multi-select). Then click the **[Add]** button.

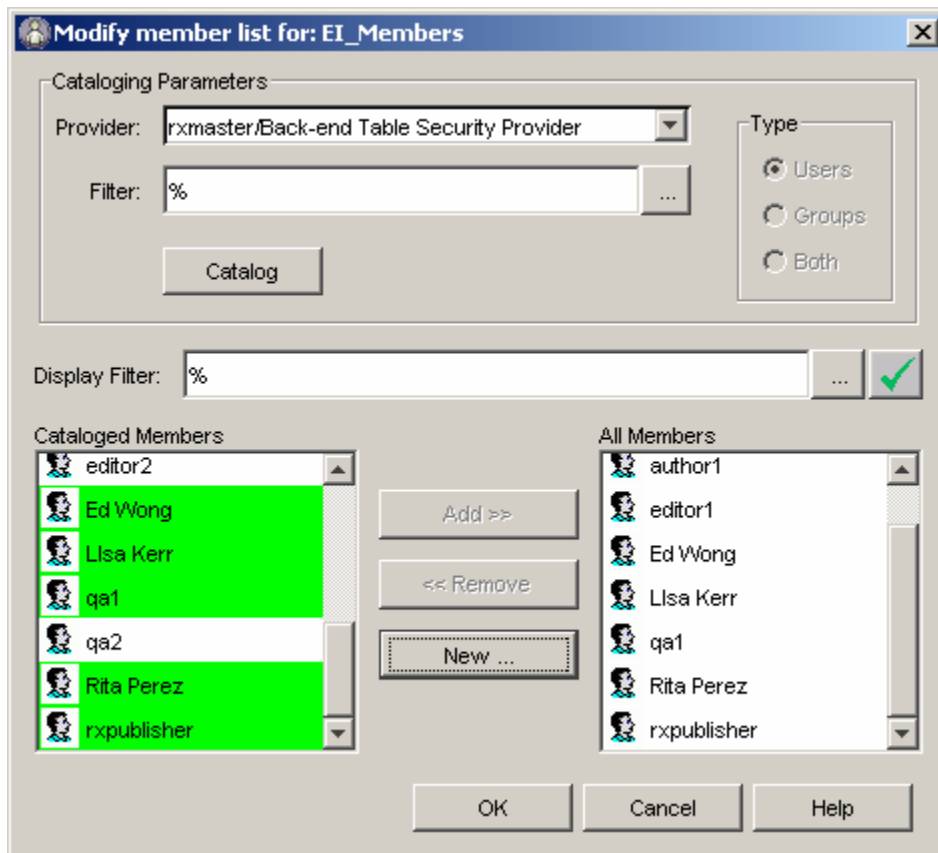


Figure 13: Modify Members dialog showing new Members added to the EI\_Members Role

- Click the **[OK]** button to save your changes.



## Communities

Communities streamline the content and design elements available to users by filtering the Content Types, Templates, Workflows, Sites, and other elements available to them. Rhythmyx displays only those elements associated with the user's Community. A user can belong to more than one Community, but can only log in to one Community at a time. (To see all the elements controlled by Communities, go to the Community Visibility view of the Rhythmyx Workbench.)

Typically, each Site defined in the system has at least one Community, but one Community may access several Sites, and a Site may use more than one Community to control access to Content Items. In FastForward, for instance, the EnterpriseInvestments and CorporateInvestments Sites each have two Communities: one Community can access and maintain the Managed Navigation Content Items in the Site (EI\_Admin, CI\_Admin) the other Community (EI\_Members and CI\_Members) can access and maintain all other Content Items in the Site.

Note that Community is not the sole factor that determines whether a user can access a specific Content Item. The user must also be in a Workflow Role that has access to the Content Item. If the user's Workflow Role is not assigned to the current State of the Content Item, they will not be able to see or access the Content Item.

## Creating a New Community

To illustrate the process of creating a Community, we will create the Enterprise Investments Community, with the description "*Community for users that create and manage Content Items for the Enterprise Investments Site*". We will assign **the EI\_Members Role that we created earlier** (see "Creating a Role" on page 20) as a Role assigned to this Community.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Enterprise Investments Community:

- 1 Log in to the Rhythmyx Workbench.
- 2 Click on the Security Design tab.
- 3 In the Menu bar, choose *File > New > Other*.
- 4 On the New Wizard dialog, choose Community and click the [Next] button.

5 The Rhythmyx Workbench displays the Community wizard.

**New community**

**Create a new Community**

Name is a required field.

Community name:

Description:

Available roles:

- Admin
- Artist
- Author
- CI\_Admin\_Members
- CI\_Members
- Default
- Designer
- Editor
- EI\_Admin\_Members
- EI\_Members
- QA
- Report\_Admin
- RxPublisher
- Web\_Admin

Assigned roles:

Finish Cancel

Figure 14: New Community Wizard

- 6 In the Community name field, enter *Enterprise Investments*.
- 7 In the Description field, enter *Community for users that create and manage Content Items for the Enterprise Investments Site*.
- 8 In the Available Roles field select *EI\_Members*. Click the [**>**] (add) button to move the EI\_Members Role to the Assigned Roles field.
- 9 Click the [**Finish**] button to complete the wizard.

As we create additional design elements later in the implementation, we will add them to the Community.

## Workflows

A Workflow defines the business process Content Items go through to be created, maintained, published and archived. Various individuals and business departments play different roles in this process, such as designers, writers, editors, and content approvers. These individuals are assigned to Rhythmyx Roles. Content Items pass through a number of stages in the process, or States; States typically have names that describe the activity occurring in the State such as Draft, Review, Publish, or Archive. Roles are assigned to each State to provide users with the ability to access the Content Items in each State. The mechanism used to move a Content Item from one State to another is called a Transition. Some Transitions are manually executed by users, but others, known as Aging Transitions, are executed automatically by the system. Manual Transitions may be restricted to specific users, or may require a set number of users to approve the Transition before the Content Item can be Transitioned successfully.

The FastForward implementation includes two basic Workflows: Simple and Standard. You should be able to implement most of the Workflows you require by copying and modifying these Workflows. We will demonstrate the implementation of Workflows by illustrating the implementation of specific features of these Workflows. To demonstrate some Rhythmyx Workflow features, we will change the implementation plan for some features of the Standard Workflow.

### Implementing the Simple Workflow

To illustrate the process of implementing a Workflow from scratch, we will demonstrate the implementation of the following selected features of the Simple Workflow:

- Draft State; we will also discuss the value of the Publishable parameter for other States
- Author Role assigned to Draft as an Assignee, and Editor assigned to Draft as a Reader
- Approve Transition as a manual Transition
- Age to Public as an Aging Transition
- Content Archived Notification

Later, we will implement the Standard Workflow to illustrate copying a Workflow and other Workflow features.

## Creating a Workflow

The first step in creating a Workflow from scratch is creating the new Workflow itself.

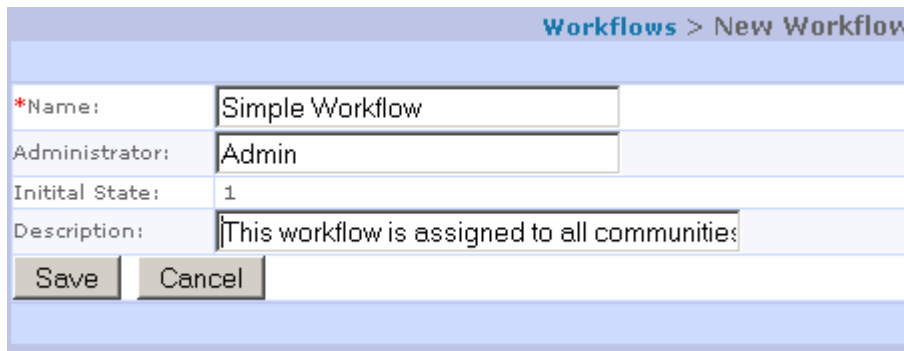
Each Workflow requires a *Workflow Administrator*<sup>1</sup>. We will assign the Admin Role as the Workflow Administrator of the Simple Workflow.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Simple Workflow:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the New Workflow link.  
Content Explorer displays the New Workflow page.
- 3 In the Name field, enter *Simple Workflow*.
- 4 In the Administrator field, enter *Admin*.
- 5 In the Description field, enter *This workflow is assigned to all communities*.

Note that the dialog includes a default value for Initial State. We will update this value after we create the Draft State.



Workflows > New Workflow	
*Name:	Simple Workflow
Administrator:	Admin
Initial State:	1
Description:	This workflow is assigned to all communities
Save Cancel	

Figure 15: Creating the Simple Workflow Object

- 6 Click the [Save] button to create the Workflow.

---

<sup>1</sup> The Workflow Administrator can edit and transition content in any state and has the ability to override the workflow process. The Workflow Administrator role gives assigned users these rights, and is otherwise like any other role.

Normally only the user who has checked out a content item may edit or check in those content items. If that user is unavailable, there is potential for a content item to get stuck in the workflow. Without administration, you would have to modify the content record in the database itself to return the content item to an accessible condition. The Workflow Administrator role assigns designated end users the power to check in content items without manually changing the database.

Any Role assigned as Workflow Administrator must include the internal Member `rxserver`. This is a special Member used in the Aging process. If the Role assigned as Workflow Administrator does not include the Member `rxserver`, Aging Transitions will not occur because Workflow authentication will fail.

## Associating a Workflow with a Community

When creating a Workflow, you cannot associate it with a Community because Workflows are not created in the Rhythmyx Workbench. Instead, you must manually associate the Workflow with the Community after creating it. Note that you can add this association at any time after initially creating the Workflow; for example you could wait until after creating the States, Transitions, and other Workflow elements rather than associating the Workflow with the Community immediately after creating the Workflow.

We must assign the Simple Workflow to the Enterprise Investments Community to make the Simple Workflow available to users in that Community.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To associate a the Simple Workflow with the Enterprise Investments Community:

- 1 Start the Rhythmyx Workbench. For instructions, see *Starting the Rhythmyx Workbench* (on page 12).
- 2 Display the Community Visibility View. To display the Community Visibility View:
  - 1 In the Menu bar of the Rhythmyx Workbench, choose *Window > Show View > Other*.

The Rhythmyx Workbench displays the Show View dialog

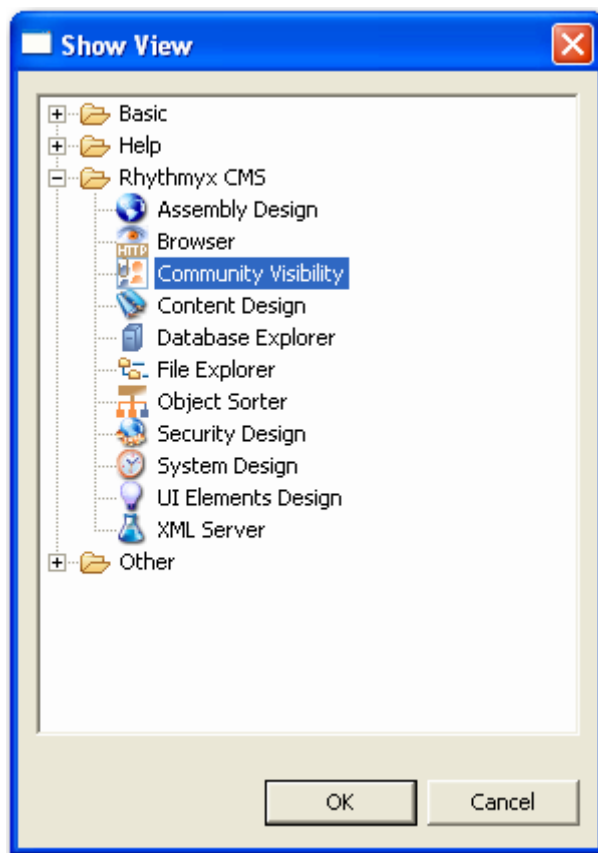


Figure 16: Show View dialog

- c) Select Community Visibility and click the [OK] button.

- d) Drag the Community Visibility View to the working area of the Rhythmyx Workbench.
- 2** In Community Visibility View, expand the Enterprise Investments Community, then expand the Visible Workflows node.
- 3** Display the System Design View.
- 4** On the System Design View, select the Simple Workflow and drag it to the Visible Workflows node of the Enterprise Investments Community in Community Design View.

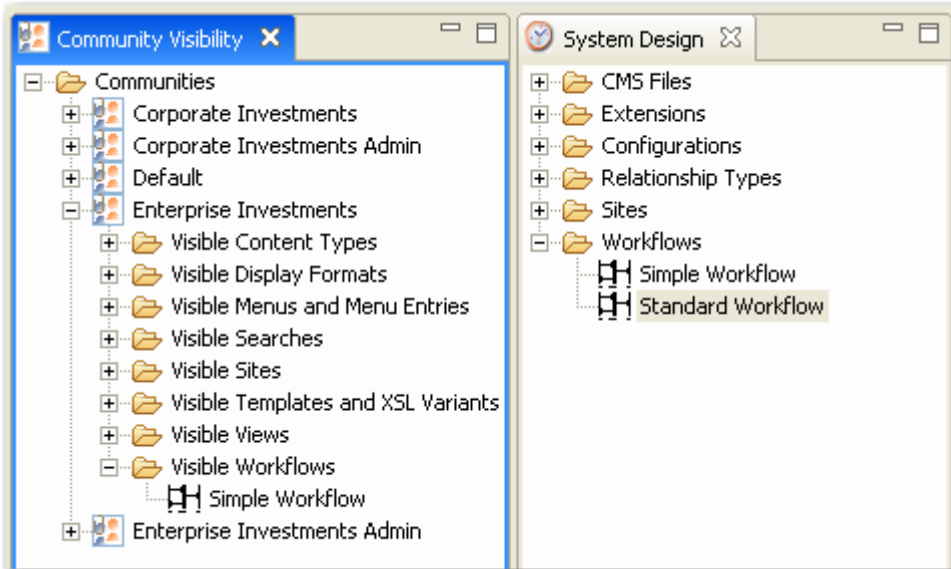


Figure 17: Standard Workflow added to the Enterprise Investments Community

## Adding a Role to a Workflow

If you want to associate any Roles with a Workflow, you must create them on the Rhythmyx server first. For details about creating a Role, see "*Creating a Role* (on page 20)". You must associate a Role with a Workflow before you can assign the Role to a State.

Note that you do not create new Roles in the Workflow. You can only assign a Role to a Workflow if the Role already exists on the Rhythmyx server.

Reviewing the *implementation plan for the Simple Workflow* (see page 470), we see that we need to assign the following Roles:

- Admin
- Author
- Editor
- QA
- RxPublisher (This is an internal Role used in Publishing; it must always be assigned to the Public State to ensure that the Publisher can access the Content Items to publish)
- Web Admin

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To illustrate the process, we will add the Author Role to the Simple Workflow:

- 5 Log in to Content Explorer and click the Workflow tab.
- 6 Click the [Simple Workflow](#) link in the Name (ID) column.  
Content Explorer displays the Simple Workflow page.
- 7 Click the [New Roles](#) link.  
Content Explore displays the New Role page.
- 8 In the Name drop list, choose *Author*. (The drop list contains all Roles in the system not already added to the Workflow.)

The screenshot shows a web form titled "New Role" within the "Simple Workflow" section. The form includes a required field for the role name, which is currently set to "Author". There is also a description field and two buttons: "Save" and "Cancel".

Figure 18: Adding a Role to a Workflow

- 9 Click Save. You have added the Role to the Workflow and can now assign the Role to to a State.

## Creating a Workflow State

States are the stages in a Workflow through which Content Items pass. Once you create a State, you can assign Roles to the State, which allows users to act on Content in that State.

Reviewing the implementation plan for the *Simple Workflow* (see page 470), we see that it includes the following States:

- Draft
- Pending
- Public
- Quick Edit
- Archive

One key piece of data in a State definition is the value in the Publishable field. This value determines whether Content Items in the State are eligible to be published. Rhythmyx is shipped with the following options for this field:

Publishable Value	Description	Example States in the Simple Workflow
Unpublish	Do not publish Content Content Items; if a Content Item has been published, remove it from the published output  Content Items in a State with this value can be seen as related Content Items in Previews.	Draft, Pending
Publish	Publish Content Items	Public
Ignore	Ignore the Publishable value; if the Content Item has not yet been published, do not publish it; if the Content Item has been published, continue to publish the last Revision that was published.	Quick Edit
Archive	Do not publish Content Content Items; if a Content Item has been published, remove it from the published output  Content Items in a State with this value cannot be seen as related Content Items in Previews.	Archive

Note that it is possible to add other values to extend the Publishable functionality. For details see "Extending Publishable States" in the *Rhythmyx Technical Reference Manual*.

To demonstrate the process of creating a State, we will create the Draft State. To create the Draft State:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow link name in the Name (ID) column.  
Content Explorer displays the Simple Workflow page.
- 3 Click the New State link.  
Content Explorer displays the New State page.
- 4 In the Name field, enter *Draft*.
- 5 In the Description field, enter *All Content Items start here*.



- 6 In the Sort Order field, enter 10.
- 7 In the Publishable drop list, choose *Unpublish*.

Figure 19: Creating a New Workflow State

- 8 Click the [Save] to save the State.

Content Explorer returns to the Simple Workflow page.

### Assigning a Role to a State

You must assign a Role to a State to allow users in that Role to access and act on Content Items in that State. Note that you must already have *added the Roles to the Workflow* (see page 35) before you can assign them to a State.

When you assign a Role to a State, you also determine whether users in the Role can act on Content Items in the State, or can only see them. This access is defined by the value in the Assignment field, which can take the following values:

- Assignee: users can see and act on Content Items in the State (in other words, they can open the Content Items to edit them, or they can Transition the Content Items).
- Reader: users have read-only access to Content Items in the State; the Content Items will be listed in Content Explorer, and the users can view both the properties and the data for the Content Items, but they cannot edit or Transition the Content Items.
- None: users cannot see or access Content Items in the State. Users may be assigned this level of access if they need to be notified about actions taken on a Content Item, but do not necessarily need to see the Content Items.

Typically, any user in a Role with Assignee access can act on a Content Item. In some cases, however, you may want to assign a Content Item to a specific user. In that situation, you must allow Ad-hoc assignment for the Role. Ad-hoc assignment allows the user Transitioning a Content Item to specify a particular user to act on the Content Item.

Finally, the Role assignment data also determines whether users in the Role will receive Notification e-mails sent when a Content Item Transitions to or from the State, and whether Content Items in the State will appear in the Inbox view of users in the Role.

Let us examine the Role assignments for the Draft State from the Simple Workflow implementation plan:

Roles	Assginement Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	No

Roles	Assginement Type	Ad Hoc	Show in Inbox	Receive Notifications?
Admin	Assignee	No	No	No
Author	Assignee	No	Yes	No
Editor	Reader	No	No	No

Members of the Author, Admin, and Web Admin Roles will be able to both see and act on Content Items in this State. Ad hoc assignment will be disabled, and none of the Members of these Roles will receive Notifications. Only Members of the Author Role will see Content Items in their Inbox View. Members of the Editor Role will be able to see Content Items in the Draft State, but will not be able to act on them.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To assign the Author Role to the Draft State:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow name in the Name (ID) column.
- 3 In the States section of the page, click Draft link.  
Content Explorer displays Edit State page for the Draft State.
- 4 In the Assigned Roles section of the page, click the New Assigned Role link.  
Content Explorer displays the New Assigned Role page.
- 5 In the Role drop list, choose *Author*.
- 6 In the Assignment drop list, choose *Assignee*.
- 7 In the Show in Inbox drop list, choose *Yes*.
- 8 In the Ad-hoc drop list leave *Disabled* selected, and in the Notify drop list leave *No* selected.

The screenshot shows a web form titled "New Assigned Role" with the following fields and values:

- Role: Author
- Assignment: Assignee
- Ad-hoc: Disabled
- Notify: No
- Show in In-Box: Yes

Buttons: Save, Cancel

Figure 20: Assigning a Role to a State

- 9 Click the [Save] button to assign the Role.

Assigning the Admin and Web Admin Roles to the Draft State follows essentially the same procedure, except for a change at Step 7. When assigning those Roles, the value specified for **Show in Inbox** is *No*.

When assigning the Editor Role, the value specified for the **Assignment** field is *Reader*.

### Assigning an Initial State to a Workflow

For each Workflow, you must define an Initial State. The Initial State is the State Content Items enter when they are first created.

Recall that when we created the Simple Workflow, this field had a default value. We could not assign an Initial State until we had created a State.

If you view the Simple Workflow page now, you will notice that the value in the Initial State field is *Draft*, which is the State we just created. The first State you create defaults as the value of the Initial State of the Workflow. For that reason, good practice is to ensure that the first State you create in your Workflow is the State specified in your implementation plan as the Initial State of the Workflow. If you follow this practice, that State will default as the Initial State of the Workflow, and you will not need to assign the Initial State manually.

### Defining Transitions for the Simple Workflow

Transitions are the mechanism Rhythmyx Workflow uses to move Content Items from one State to another. Each Transition defines:

- the State to which the Content Item will move;
- how the Content Item must be approved to move to a new State; and
- any automated processing that will occur during the Transition.

Rhythmyx uses two types of Transitions:

- Manual Transitions require a user to initiate the Transition.
- Automatic Transitions, known as Aging Transitions, are initiated automatically by the system.

To illustrate a manual Transition, we will implement the Approve Transition from the Draft State to the Pending State. To illustrate an Aging Transition, we will implement the Age to Public Transition from the Pending State to the Public State.

### Implementing a Basic Manual Transition

The Approve Transition from the Draft State is a very basic Transition:

Name	To State	Details	Notification
Approve	Pending	Manual Transition, 1 Approval	No

This Transition does not require any special approvals or automated processing.

Note that the procedure below assumes that you have already created the Pending State, which is the Target State for the Transition. If you do not create the Target State before you create the Transition, you need to come back to the Transition after you create the Target State and assign it.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To implement the Approve Transition:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow name in the Name (ID) column.
- 3 In the States section of the page, click Draft link.

Content Explorer displays Edit State page for the Draft State.

- 4 In the Transitions section, click the [New Transitions](#) link. (NOTE: Do not confuse this section with the Aging Transitions section. We will create an Aging Transition later.)

Content Explorer displays the New Transition page.

- 5 In the Label field, enter *Approve*. This is the label Content Explorer will display for the Transition.
- 6 In the Description field, enter *Approve Item to Pending*.
- 7 In the Trigger field, enter *Approve*. Rhythmyx uses the value in the Trigger field for internal processing. Each Transition defined for a State requires a different value in this field.
- 8 In the To State drop list, select *Pending*. The values in this drop list include all States currently defined for the Workflow.
- 9 The specification for this Transition requires only one approval, and it does not specify any specific Roles for the Transition, so we will leave the default values in the Approval Type drop list (*Specified Number*) and Approvals Required field (*1*).
- 10 The specification does not indicate that a comment is required, so we will leave the default value (*Optional*) for the Comments field. We have not specified a Default Transition for the Draft State, so we leave the value of the Default Transition drop list as *No*. Nor is automatic processing is specified, so we leave *None* as the value in the Workflow Action field. Finally, the specification does not restrict the Transition to specific Roles, so we leave *All Roles* as the Value in the Transition Roles drop list.

Workflows > Simple Workflow > Draft > New Transition	
ID:	
*Label:	Approve
Description:	Approve item toPublic
*Trigger:	Approve
From-State:	Draft (1)
To-State:	Pending
Approval Type:	Specified Number
*Approvals Required:	1 (Required if Approval Type is set to "Specified Number")
Comment:	Optional
Default Transition:	No
Workflow Action:	None
Transition Role:	- All roles -
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 21: Defining the Approve Transition

- 11 Click the [OK] button to save the Transition.

## Implementing an Aging Transition

An Aging Transition is a Transition that Rhythmyx initiates automatically. An Aging Transition may be initiated:

- once, based on reaching a specified date;
- once, after a specified period of time has passed; or
- repeatedly at specified intervals.

In this example, we will define the Age to Public Transition, which is initiated when a specified date occurs. Later, we will implement a repeating Transition and review other Aging Scenarios.

Let's review the specification for the Age to Public Transition:

Name	To State	Details	Notification
Age to Public	Public	Aging Transition, Triggered by Start Date of Content Item	No

This Transition automatically moves a Content Item from the Pending State to the Public State when the Start Date of the Content Item is reached. Note that the following procedure assumes that the Public State has already been created.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To implement the Age to Public Transition:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow name in the Name (ID) column.
- 3 In the States section of the page, click Pending link.  
Content Explorer displays Edit State page for the Pending State.
- 4 In the Aging Transitions section, click the New Aging Transitions link. (NOTE: Do not confuse this section with the Transitions section.)  
Content Explorer displays the New Aging Transition page.
- 5 In the Label field, enter *Age to Public*. Copy the value and paste it to the Description and Trigger fields.
- 6 In the To State drop list, choose *Public*.
- 7 We want to initiate the Transition on the Start Date of the Content Item, so from the Aging Type drop list, choose *System Field* and from the System Field drop list choose *Start Date*.

- 8 The specification for the Age To Public Transition does not call for any automatic processing, so leave *None* as the value for the **Workflow Action** drop list. Since we did not choose either the Repeated or Absolute Aging Type, we do not have to enter a value in the **Aging Interval** field.

Workflows > Simple Workflow > Pending > New Aging Transition	
ID:	
*Label:	Age to Public
Description:	Age to Public
*Trigger:	Age to Public
From-State:	Pending (2)
To-State:	Public
Workflow Action:	None
Aging Type:	System Field
Aging Interval(min):	(Required for Absolute and Repeated Aging Types)
System Field:	Start Date (Required for System Field Aging Type)
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 22: Defining the Age to Public Transition

- 9 Click the **[Save]** button to save the Transition.

## Implementing Notifications

Notifications are generic e-mail messages that the system can send automatically when a Content Item is Transitioned. Once created, Notifications must be associated with a Transition. The data for this association specifies which Roles will receive the Transition and additional recipients of the message.

The specification of the Public State of the Simple Workflow includes a Notification that is associated with two Transitions: Expired and Age to Archive. The specified subject of the e-mail message is "Content Archived", and the specified text is "A content item has transitioned into the archived state and will be removed from your web site." In both cases, the Members of the Roles assigned to the To State should receive the Notification message.

### Creating the "Content Archived" Notification

To create the Content Archived Notification:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow name in the Name (ID) column.  
Content Explorer displays the Simple Workflow page.
- 3 In the Notifications section, click the New Notifications link.  
Content Explorer displays the New Notification page.
- 4 In the Subject field, enter *Content Archived*.

- 5 In the **Description** field, enter *Notification for Transitions into the Archive State*.
- 6 In the **Body** field, enter *A content item has transitioned into the archived state and will be removed from your web site*.

The screenshot shows a web interface for configuring a workflow notification. The breadcrumb path is 'Workflows > Simple Workflow'. The form has the following fields:

- ID:** (empty)
- \*Subject:** Content Archived
- Description:** Notification for Transitions into the
- Body:** A content item has transitioned into the archiv will be removed from your web site.

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

Figure 23: Defining the Content Archived Notification

- 7 Click the [Save] button to save the Notification.

### Associating the Content Archived Notification

To assign the "Content Archived" Notification to the Expired Transition:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Simple Workflow name in the Name (ID) column.  
Content Explorer displays the Simple Workflow page.
- 3 In the States section, click on the Public link.  
Content Explorer displays the Public State Page.
- 4 In the Transitions section, click on the Expired link.  
Content Explorer displays the Expired Transition page.
- 5 In the Transition Notifications section, click the New Transition Notification link.  
Content Explorer displays the Transition Notification page.
- 6 In the Notification Subject drop list, choose *Content Archived*.
- 7 In the State Role Recipients Type drop list, leave the default option, *To State Role Recipients only*.

- 8 The specification does not call for additional recipients, so leave the Additional Recipients and CC fields blank.

The screenshot shows a web-based configuration form for a workflow notification. The breadcrumb trail at the top reads: **Workflows > Simple Workflow > Public > Expired > Tran**. The form contains the following fields:

- ID:** A text input field.
- Notification: Subject(ID):** A dropdown menu with the selected value "Content Archived(1)".
- State Role Recipients Type:** A dropdown menu with the selected value "To State Role Recipients only".
- Additional Recipient List:** A large text area with the instruction "(Comma separated list of email addresses)".
- CC List:** A large text area with the instruction "(Comma separated list of email addresses)".

At the bottom of the form are two buttons: "Save" and "Cancel".

*Figure 24: Assigning the Content Archived Notification*



- 9 Click the [Save] button to save the Notification assignment. Content Explorer returns you to the Expire Transition page.

Workflows > Simple Workflow > Public > Expire

ID:	8
*Label:	Expire
Description:	Expire to Archive
*Trigger:	Expire
From-State:	Public (3)
To-State:	Archive
Approval Type:	Specified Number
*Approvals Required:	1 (Required if Approval Type is set to "Specified Number")
Comment:	Optional
Default Transition:	Yes
Workflow Action:	sys_TouchParentItems
Transition Role:	- All roles -

Save Cancel

**Transition Roles**  
New Transition Role

Role (ID)
No entries found.

**Transition Notifications**  
New Transition Notification

Subject (Notification ID)	State Role Recipient Type	Additional Recipient List	CC List
✘ Content archived (2)	To State Role Recipients only		

Figure 25: Expire Transition with Content Archived Notification assigned

Repeat this procedure, with appropriate changes, to assign the Notification to the Age to Archive Transition.

### Specifying Workflow Properties for Notification

The final step in implementing Notifications is to define the Workflow properties for Notification.

Workflow properties are defined in the

file <Rhythmyxroot>/rxconfig/Workflow/rxworkflow.properties. You must specify values for the following properties:

- SMTP\_Host

This property defines the SMTP mail server used to e-mail Notifications. You can specify either the name of a server or its IP address. Specifying the name of the server is generally preferable. While the IP address of a server may change, it is unlikely that the name of the server will change.

- MAIL\_DOMAIN

This property is used to generate an originating e-mail address for the Notification if the user making the triggering Transition does not have a value for sys\_e-mail. In that case, Rhythmyx generates an e-mail address by concatenating the user name and the value of this property. If you do not specify a value for this property and you have any users that do not have a value for sys\_e-mail, Rhythmyx will return errors when generating Notifications.

- RXSERVER\_HOST\_NOTIFICATION

This property is used by the aging agent to generate links in Notification e-mails. Specify the name or IP address of the Rhythmyx server.

- RXSERVER\_PORT\_NOTIFICATION

This property is used by the aging agent to generate links in Notification e-mails. Specify the port of the Rhythmyx server.

You can optionally specify a value for the following property:

- RX\_SERVER\_IS\_SSLINK\_NOTIFICATION

If you enable SSL on your Rhythmyx server, specify "yes" for this property to ensure that the links in Notification e-mail messages use SSL to link to Content Items on your server. If this property specifies any other value, or if no value is specified for this property, links to Rhythmyx Content Items will not be generated for SSL. (NOTE: For details about setting up SSL, see "Enabling SSL on the Rhythmyx Server" in the document *Setting Up the Production Environment*.)

For example, assume we want to use the following values:

- SMTP host: outbound.enterpriseinvestments.com
- Mail domain: enterpriseinvestments.com
- Rhythmyx server name: Rhythmyx
- Rhythmyx installation directory: Rhythmyx
- Rhythmyx server port: 9992

For the purposes of this exercise, assume the SSL is not being used.

To specify Notification properties:

- 1 On the machine named Rhythmyx, start Notepad or another simple text editor.
- 2 Open the file `C:\Rhythmyx\rxconfig\Workflow\Workflow.properties`.
- 3 Specify the following values for these properties:

```
SMTP_Host=outbound.enterpriseinvestments.com
```

```
MAIL_DOMAIN=enterpriseinvestments.com
```

```
RXSERVER_HOST_NOTIFICATION=Rhythmyx
```

```
RXSERVER_PORT_NOTIFICATION=9992
```

---

NOTE: Not all properties in the file are illustrated here; only those properties associated with Notification are described.

---

- 4 Save the file `Workflow.properties`.

## Implementing the Standard Workflow

It is usually easier to implement a Workflow by copying an existing Workflow and modifying it rather than implementing a completely new Workflow. Copying a Workflow avoids the need to repeat several of the procedures when one Workflow closely matches another.

For example, the major difference between the Standard Workflow and the Simple Workflow is that the Standard Workflow has one additional State (Review). The Review State requires Transitions to other States, and also requires redirecting some existing Transitions.

In addition to demonstrating how to copy a Workflow, we will make some minor changes to the specification of the Standard Workflow to illustrate additional Rhythmyx Workflow features:

- enable Ad Hoc assignment and Notification for the Author Role in the Draft State;
- require user comments on the Rework Transition from the Review State to the Draft State and add a Notification, including user comments, to that Transition;
- add a repeating Reminder Transition to the Review State;
- change the Approve Transition from Review to Pending to require approvals from both the Editor and QA Roles; and
- implement an alternative escalating scenario for the Approve Transition.

## Copying a Workflow

The Simple and Standard Workflows included with FastForward provide a basic level of Workflow functionality based on Percussion Software experience implementing Workflows for a variety of customers. In most cases, you can implement your Workflows by copying one of the FastForward Workflows and modifying it to meet your needs. In this case, we will copy the Simple Workflow to create the Standard Workflow.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To copy the Simple Workflow and create the Standard Workflow:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the [Copy Workflow](#) Link.  
Content Explorer displays the Copy Workflow page.
- 3 In the Source Workflow drop list, choose *Simple Workflow*.

- 4 In the New Workflow field, enter *Standard Workflow*.

Figure 26: Copying the Simple Workflow to create the Standard Workflow

- 5 Click the [Create] button to create the Standard Workflow.
- 6 Content Explorer returns you to the Workflow page.

Workflows			
New Workflow Copy Workflow			
	Name (ID)	Description	Preview
	Simple Workflow (4)	This workflow is assigned to all communities	
	Standard Workflow (5)	This workflow requires two approvals before content is published. It is the default for most content types and is assigned to all communities.	

Figure 27: Workflows after creating the copy

## Enabling Ad Hoc Assignment and Notification for a Role Assignment

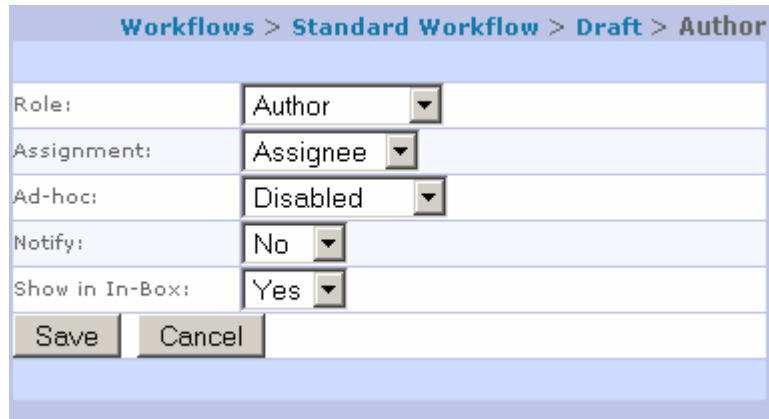
Suppose that when returning a Content Item to the Draft State, we want the reviewer to be able to assign it specifically to the user that last worked on it rather than assigning it to the Author Role generally. We need to enable Ad Hoc assignment to allow this behavior. Later we will see how to create a Notification that allows the assignee to see the comments added when a Content Item is sent back. To allow Members of the Author Role to receive the Notifications, we must enable Notification for the Role.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To enable Ad Hoc assignment to the Author Role in the Draft State:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Standard Workflow name in the Name (ID) column.
- 3 In the States section of the page, click Draft link.  
Content Explorer displays Edit State page for the Draft State.
- 4 In the Assigned Roles section of the page, click the Author link.

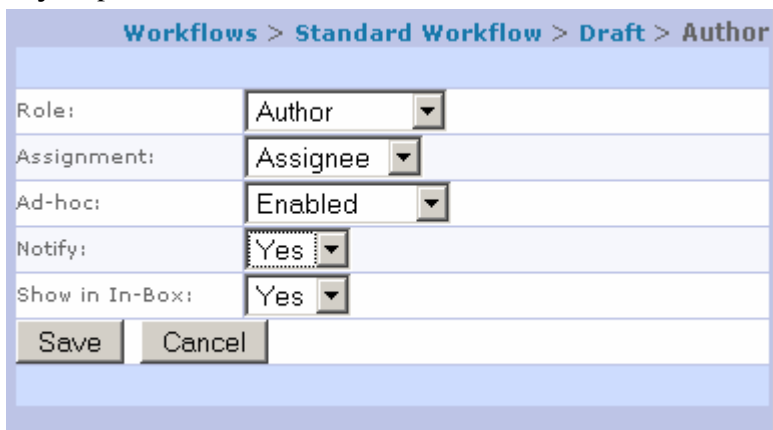
- 5 Content Explorer displays the Author Role Assignment page.



Workflows > Standard Workflow > Draft > Author	
Role:	Author
Assignment:	Assignee
Ad-hoc:	Disabled
Notify:	No
Show in In-Box:	Yes
Save Cancel	

Figure 28: Assigning the Author Role to the Draft State with Ad Hoc Assignment enabled

- 6 In the Ad Hoc drop list, choose *Enabled*.
- 7 In the Notify drop list, choose *Yes*.



Workflows > Standard Workflow > Draft > Author	
Role:	Author
Assignment:	Assignee
Ad-hoc:	Enabled
Notify:	Yes
Show in In-Box:	Yes
Save Cancel	

Figure 29: Assigning the Author Role to the Draft State with Notification enabled

- 8 Click the **[Save]** button to save your changes.  
Content Explorer returns you to the Draft page.

Now, suppose a user wants to return a Content Item to Ed Wong, who was the last user that worked on it. When the user initiates the Transition, Content Explorer displays the following dialog:



Figure 30: Content Explorer Transition dialog for Workflow Comments and Ad Hoc assignee

The user could enter *Ed Wong* directly into the Ad hoc Assignees field, but to ensure that they got the right name, they click the **[Search]** button and enter *Ed* in the Name Filter field:

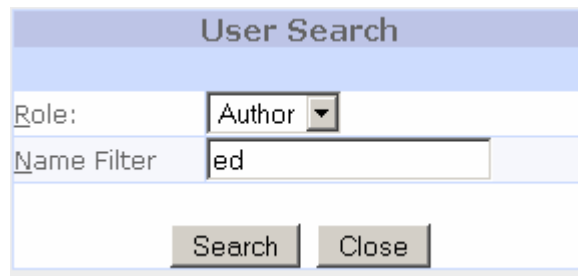


Figure 31: Searching for users with names beginning "ed"

which returns the following results:

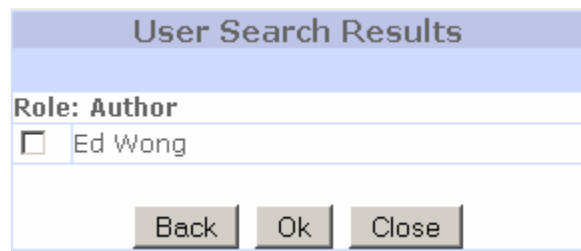


Figure 32: Search results returned with Ed Wong

When the user checks the box next to Ed Wong's name and clicks the [OK] button, Ed Wong is added as the assignee. The user can also enter a comment in the **Workflow Comments** field. Later, we will see how to implement a Notification that includes this comment.

## Adding a New State to a Copied Workflow

Let us review the specification for the Review State from the implementation plan for the Standard Workflow:

### State: Review

Publishable Value: Unpublish

Sort Order: 20

Assigned Roles

Roles	Assignment Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Author	Reader	No	No	No
QA	Reader	No	No	No
Editor	Assignee	No	Yes	Yes

Transitions

Name	To State	Details	Notification
Approve	Pending	Manual Transition, 1 Approval	No
Rework	Draft	Manual Transition, 1 Approval	No

Creating this State uses the same procedure that we used to create the Draft State for the *Simple Workflow* (see page 36). We also need to update the Sort Order for the other States in the Workflow to match the Standard Workflow specification.

## Updating Transitions in a Copied Workflow

Since we have added a new State to our Workflow, we will naturally need to add some new Transitions. We will also need to replace another transition.

Let us review the Transition specification for the Review State:

Name	To State	Details	Notification
Approve	Pending	Manual Transition, 1 Approval	No
Rework	Draft	Manual Transition, 1 Approval	No

We will need to add these Transitions, using the procedure described in *Implementing a Basic Manual Transition* (see page 39).

If we Preview the Standard Workflow as initially created, we notice that it includes an Approve Transition from Draft to Pending:

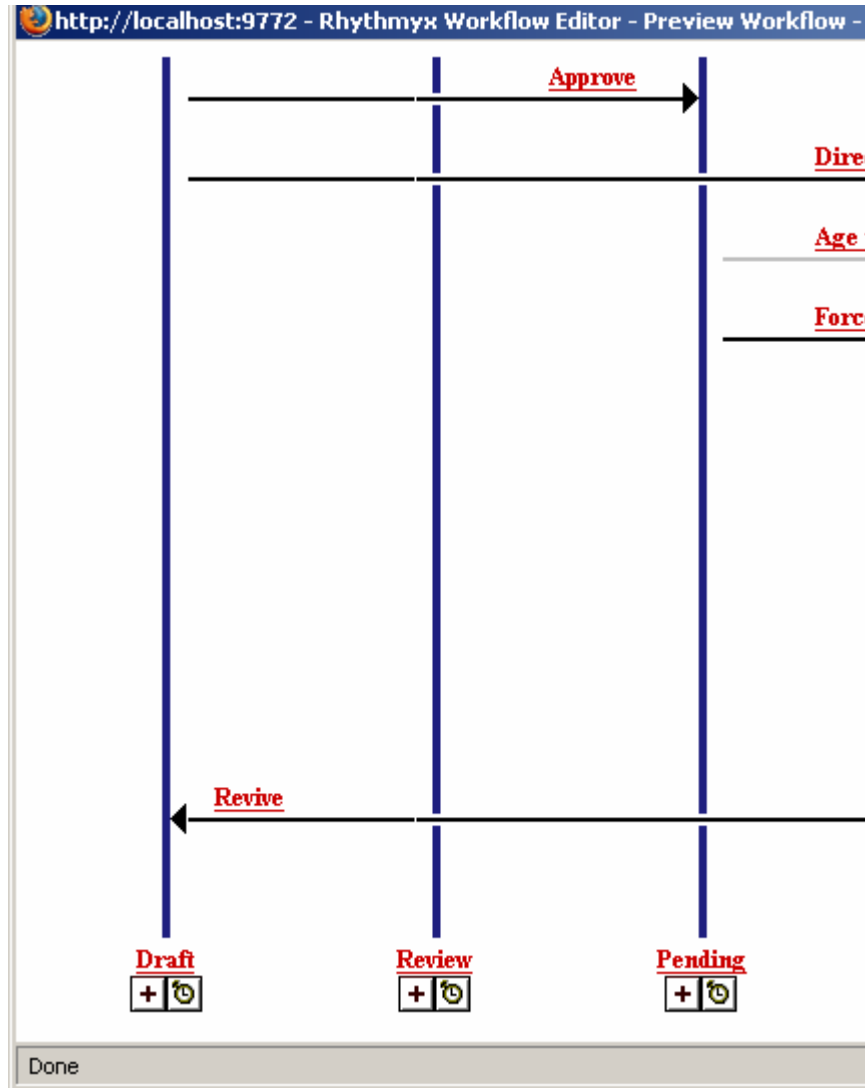


Figure 33: Preview of Standard Workflow as initially copied

The specification for the Draft State, however, calls for a Submit Transition to the Review State:

Name	To State	Details	Notification
Submit	Review	Manual Transition, 1 Approval	Content Into Review, to State Role Recipients; "A content item has transitioned into the review state."
Direct to Public	Public	Manual Transition, 1 Approval, Admin only	No



While we could edit the name of the Approve State, we cannot edit the value in the From State. We will have to delete the Approve Transition from the Draft State to the Pending State and create the Submit Transition from the Draft State to the Review State.

After we make all changes, we should see the following Preview of the Standard Workflow:

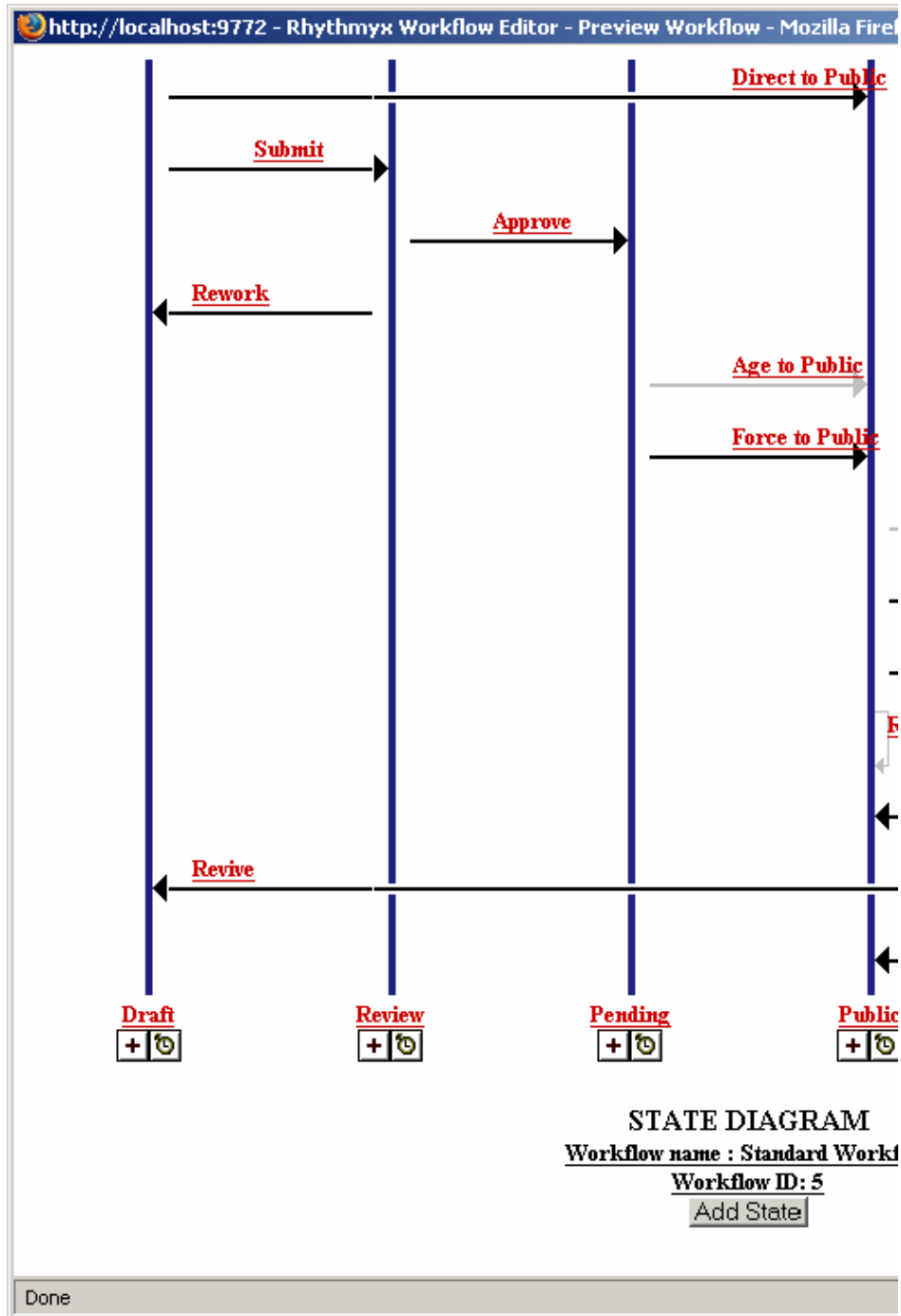


Figure 34: Preview of Standard Workflow after updating Transitions

### Requiring Comments on a Transition and Including User Comments on Notifications

Earlier, we decided to modify the specification of the Rework Transition from Review to Draft in two ways:

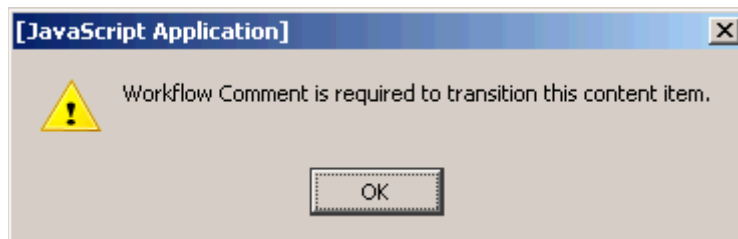
- require user comments on the Transition; and
- add a Notification that included the user Comments.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To modify the Rework Transition:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Standard Workflow name in the Name (ID) column.  
Content Explorer displays the Standard Workflow page.
- 3 In the States section of the page, click Review link.  
Content Explorer displays Edit State page for the Review State.
- 4 In the Transitions section, click the Rework link.  
Content Explorer displays the Edit Transition page for the Review Transition.
- 5 In the Comment drop list, choose *Required*.
- 6 Click the [Save] button to save the Transition.  
Content Explorer returns to the Review State page.

Now, if a user tries to use the Rework Transition but does not include comments, Content Explorer will display the following error message:



*Figure 35: Workflow Comment Required warning*

To include the user comments, we must define a new Notification. Let us assume that we will add a Notification with the Subject "Content Item Requires Additional Work" and the text: "A Content Item has been returned to Draft State for additional work with the following comment", followed by the user comment on a new line.

To implement this Notification, we would use the procedure illustrated in *Implementing Notifications* (see page 42), but we would add the macro `$wfcomment` on a new line:

The screenshot shows a web form for creating a new workflow notification. The breadcrumb trail is 'Workflows > Standard Workflow > New'. The form has the following fields:

- ID:** (empty)
- \*Subject:** Content Item Requires Additional
- Description:** Content Item Requires Additional
- Body:** Content Item has been returned to Draft State for additional work with the following comment  
\$wfcomment

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

Figure 36: Notification with `$wfcomment` macro

### Implementing a Repeating Transition

A common Workflow requirement is periodic reminders to members of a Role to which a Content Item is assigned that the Content Item requires action. To accomplish this, use a Repeated Aging Transition. A Repeated Transition occurs each time the interval specified in the Transition passes. A Notification is associated with the Transition, reminding the user that an action is necessary.

To illustrate a Repeating Transition, we will implement a Reminder Transition that will occur daily, sending the Reminder Notification to the recipients.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To implement the Reminder Transition:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Standard Workflow name in the Name (ID) column.
- 3 In the States section of the page, click Review link.  
Content Explorer displays Edit State page for the Review State.
- 4 In the Aging Transitions section, click the New Aging Transitions link.  
Content Explorer displays the New Aging Transition page.
- 5 In the Label field, enter *Reminder Transition*. Copy the value and paste it to the Description and Trigger fields.
- 6 In the To State drop list, choose *Review*. This choice creates a "circular Transition", a Transition that returns to the State from which it was initiated.
- 7 From the Aging Type drop list, choose *Repeated*. In the Aging Interval field, enter *1440* (the number of minutes in a day)

- 8 We do not want any automatic processing, so we leave *None* as the value for the **Workflow Action** drop list. Since we did not choose the System Field Aging Type, we can ignore the **System Field** field.

Workflows > Standard Workflow > Review > New Aging Transition	
ID:	
*Label:	Reminder Transition
Description:	Reminder Transition
*Trigger:	Reminder Transition
From-State:	Review (2)
To-State:	Review
Workflow Action:	None
Aging Type:	Repeated
Aging Interval(min):	1440 (Required for Absolute and Repeated Aging Types)
System Field:	(Required for System Field Aging Type)
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 37: Implementing a Repeating Transition

- 9 Click the **[Save]** button to save the Transition.

Content Explorer returns to the Review State page. Use the procedure described in "**Implementing Notifications** (see page 42)" to add the Reminder Notification to the Transition.

### Implementing a Transition Requiring Approvals from Specific Roles

In some cases, you might want to implement a Transition that requires approvals from users in several specific Roles. For example, let's change the Approve Transition from the Review State to the Pending State to require approvals from both the Editor Role and the QA Role.

Note: The following procedure assumes that the Assignment Type of the QA Role has been changed from *Reader* to *Assignee*.

To implement this behavior:

- 1 Log in to Content Explorer and click the Workflow tab.
- 2 Click the Standard Workflow name in the Name (ID) column.  
Content Explorer displays the Standard Workflow page.
- 3 In the States section of the page, click Review link.  
Content Explorer displays Edit State page for the Review State.
- 4 In the Transitions section, click the Approve link.  
Content Explorer displays the Edit Transition page for the Approve Transition.
- 5 In the Approval Type drop list, choose *Each Role*.

- 6 In the Transition Roles section of the page, click the [New Transition Roles](#) link. Content Explorer displays the Transition Role page.
- 7 In the ROLE drop list, choose *Editor* and click the [Save] button.

Figure 38: Specifying Editor as a required approver for a Transition

Content Explorer returns to the Transition Role page.

- 8 Repeat Steps 6 and 7 to add the QA Role.

When you finish, the Approve Transition page should resemble the following screenshot:

Transition Roles	
New Transition Role	
Role (ID)	
✗ Editor (3)	
✗ QA (4)	

Figure 39: Review Transition with required approvers

## Implementing an Escalating Aging Scenario

Another common Workflow scenario is escalating notices. In this scenario, when a Content Item initially Transitions into a State, the Members of the Assigned Roles are notified that the Content Item needs action, and are typically given a specific period of time to take action. After the specified period of time, an Aging Transition automatically Transitions the Content Item to another State. In between, one or more additional Transitions may occur reminding the users that they need to act on the Content Item.

This scenario uses Absolute Aging Transitions. Like Repeated Aging Transitions, Absolute Aging Transitions occur after a specified period of time has passed. Unlike Repeated Aging Transitions, Absolute Aging Transitions occur only once.

For example, suppose we want to give Members of the Editor Role three days to act on Content Items before we automatically move those items to Pending, with daily reminders that a Content Item needs attention. This scenario would involve four Transitions:

- 1 The existing Submit Transition from Draft to Review. The only modification required here would be to add a new Notification, which would inform the Editors that a Content Item required attention within three days or it would automatically move to Pending. Don't forget to enable Notification for the Editor Assignment to the Review State.
- 2 An Absolute Aging Transition to the Review State with an Aging Interval of 1440 minutes, and a Notification informing the Editors that they had two days to act on the Content Item before it automatically moved to Pending.
- 3 An Absolute Aging Transition to the Review State with an Aging Interval of 2880 minutes, and a Notification informing the Editors that they had one day to act on the Content Item before it automatically moved to pending.
- 4 An Absolute Aging Transition to the Pending State with an Aging Interval of 4320 minutes.

## Associating a Copied Workflow with Content Types

A copied Workflow is not automatically associated with the Content Types that the original Workflow is associated with.

If your system already includes Content Types when you copy a Workflow, and you want to associate the copied Workflow with existing Content Types, you must associate them manually.

The following procedure assumes that you have copied a Workflow that was associated with a group of Content Types and shows you the simplest way to associate the new Workflow with a *group* of Content Types. You can associate a Workflow with a single Content Type in by dragging it onto the Allowed Workflows folder for the Content Type in Content Design view, or by opening the Content Type's editor.

To associated a Workflow with a group of existing Content Types:

- 1 Open the System Design view and the Content Design view in separate windows.
- 2 In System Design view, expand the new Workflow to display the Allowed Content Types folder.
- 3 In Content Design view, multi-select the Content Types that you want the Workflow associated with.
- 4 From Content Design view drag the Content Types to the Allowed Content Types folder under the Workflow in System Design view.

- 5 If you expand Allowed Content Types under the Workflow, the Content Types are now listed.

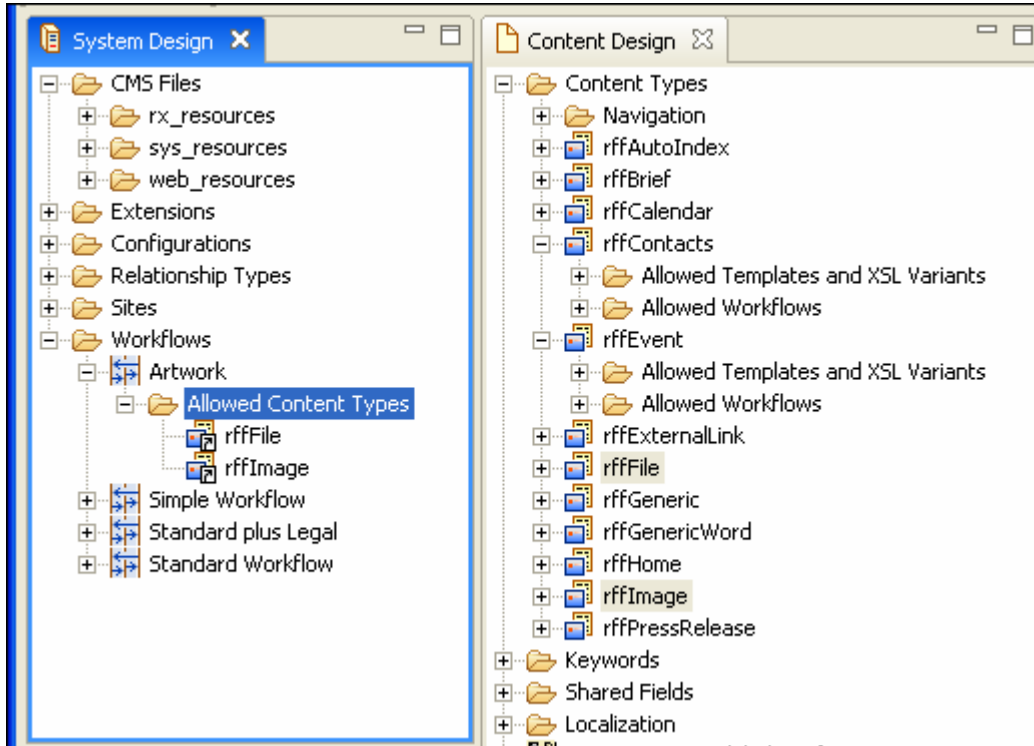


Figure 40: A Workflow and its associated Content Types

If you use Multi-Server Manager to deploy the Workflow to another server, you must separately package the Content Types or they will not have the associations to the Workflow on the target server.





## CHAPTER 4

# Setting up the Publishing Site and Basic Navigation



The Enterprise Investments Site is comprised of two pieces:

- Site Folder and Subfolders - organize the directory structure of the published Site and the Content Items that will be published to that Site.
- Site registration - identifies the location where output will be published, and the data used to connect to and pass published content to that location. Note: The Enterprise Investments registration identifies a directory; however a Site registration may specify either a directory or an FTP location

Rhythmyx can maintain multiple Sites. Each Site in Rhythmyx represents a complete directory structure in the output location. Note that in some cases, what appears to a visitor browsing the Web site as one site may be managed in Rhythmyx as two or more Site Folders.

As an implementer you will need to perform the following basic Site setup tasks:

- Create the Site Root Folder and assign user access
- Define a folder structure parallel to the intended output directory structure

- Register your Site
- Add a Managed Navigation NavTree Content Item to the Site

Optionally, you might want to define Access Control Lists (ACLs) for specific Folders in the Site to specify user access to those Folders.

---

## Creating the Site Root Folder

The first step in Rhythmyx Publishing is the creation of the Site Root folder in the Content Explorer. The Site Root folder is the hub of all the Site information to be published and is the first level of information for the Site. A Site can have only one Site Root Folder.

Creating the Site Root Folder is a simple procedure but the processes that follow require careful attention to be sure the publishing environment remains consistent.

In the following exercise we will create the Site Root folder for the Enterprise Investments web site. You already have the Enterprise Investments Site on your system as part of Rhythmyx so this is for demonstration purposes only. You would use the information in your implementation plan as substitute for the data used in the following exercise.

To create the Enterprise Investments Site Root folder:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Right-click on the Sites node and select New Folder from the popup menu. Content Explorer displays the Create Folder dialog.

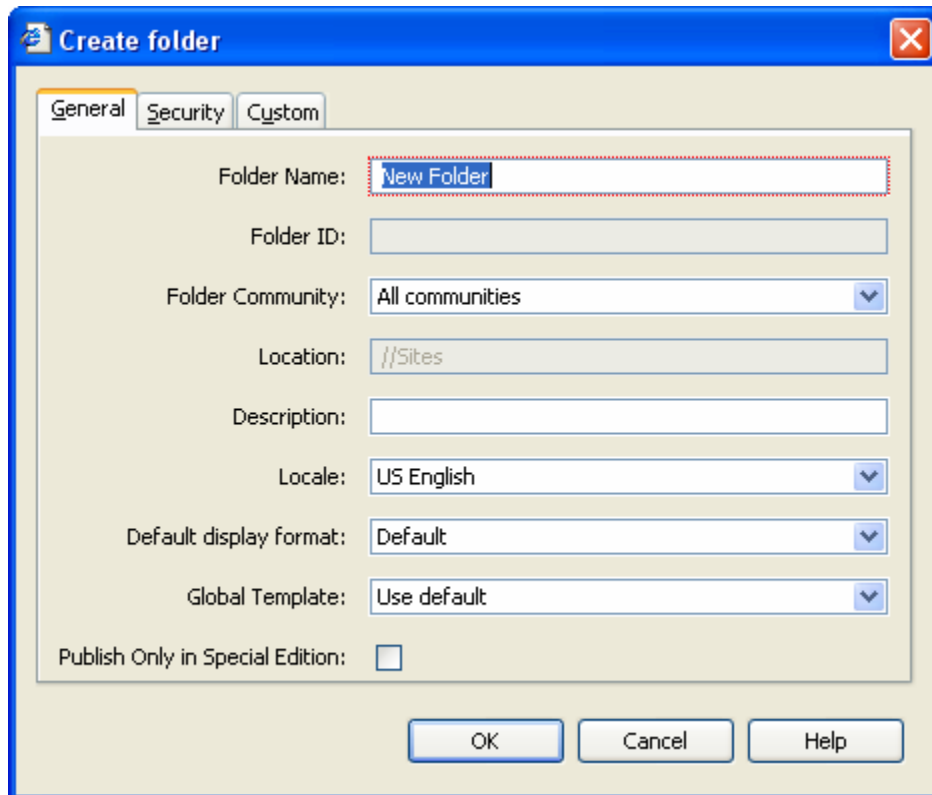


Figure 41: Create folder dialog

- 3 In the Folder Name field enter EnterpriseInvestments (no spaces). This will be the name that your Site folder will have under the Sites node in Content Explorer.

- 4 The **Folder ID** field is assigned a value by Rhythmyx when you have finished entering all of the Site Root folder attributes and save.
- 5 For **Folder Community** accept the default value All Communities. By selecting All Communities you ensure that users in all Communities can view the Enterprise Investments site. You can restrict access based on Community through this dropdown list. By selecting a community from the list, only that Community is able to view the Enterprise Investments site.
- 6 The **Location** field is unavailable because the field is automatically populated when you save the folder. It will use the Folder Name contents that you entered to name the Location.
- 7 The **Description** field is optional. If you enter a Description in here, you can change it at any time; this field is always editable.
- 8 The next field is the **Default Display Format** field. Display Formats are used to specify the columns to show in the user interface and the order in which to organize them. This field contains a dropdown list of available Display Formats for the Site. For this exercise select Default.

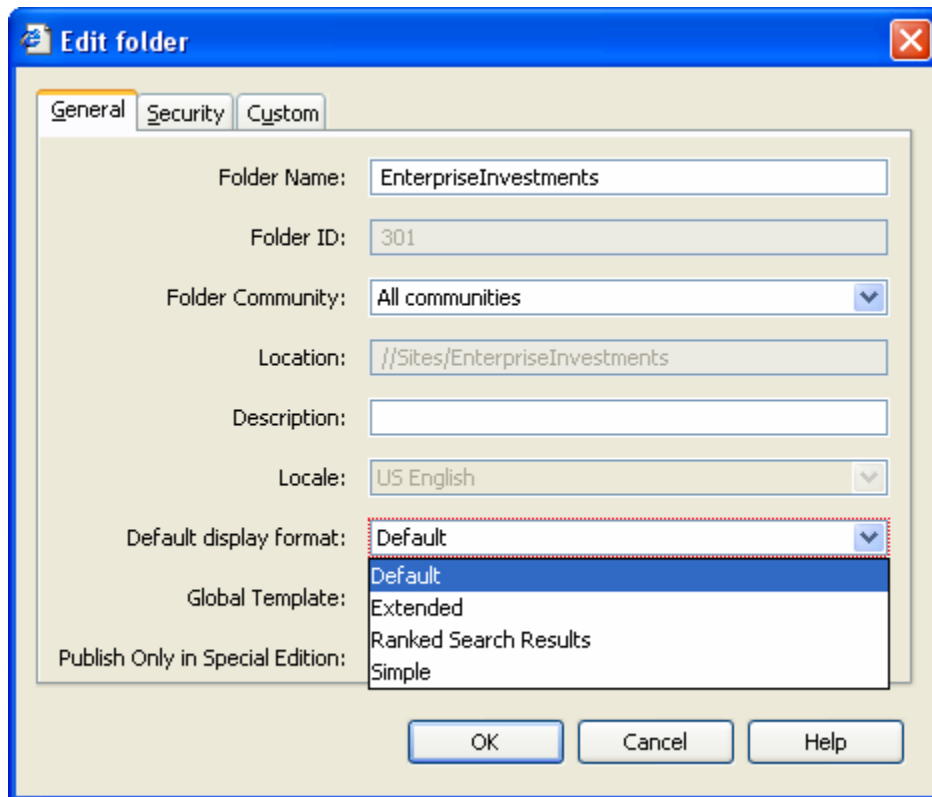


Figure 42: Edit folder dialog

- 9 The next field is the **Global Template** field. Global Templating facilitates the reuse of page templates. For this example, select Use Default.
- 10 Leave the **Publish Only in Special Edition** field unchecked. For a detailed discussion on this topic refer to the chapter *Configuring Publishing* (see page 291) in this book.

# Registering the Publishing Site with Rhythmyx

Register the Enterprise Investments Site to allow Rhythmyx to determine the correct location to publish the Site's content.

To register the Site with Rhythmyx:

- 1 Log in to Rhythmyx Content Explorer and click the Publishing tab.
- 2 In the left navigation panel, click the *Site by Name* link.
- 3 Click the *New Site* link.  
The Edit Site Properties screen opens.

Edit Site Properties	
*Site Name	<input type="text"/>
Description	<input type="text"/>
Site Address (URL)	<input type="text"/>
Home Page (URL)	<input type="text"/>
Publishing Root Location	<input type="text"/>
*Publisher	--Choose-- <input type="button" value="v"/>
Status	Active <input type="button" value="v"/>
Folder Root	<input type="text"/>
Global Template	<input type="button" value="v"/>
Nav Theme	<input type="button" value="v"/>
Allowed Namespaces	<input type="text"/>
<b>FTP Information:</b>	
IP Address	<input type="text"/>
Port Number	<input type="text"/>
User ID	<input type="text"/>
Password	<input type="text"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 43: Edit site properties page

- 4 The Site Name field is a mandatory field, it denotes the name of the Site. Enter Enterprise Investments in this field.
- 5 The Description field provides a description of your Site; it is optional. Enter the following description: Represents the Enterprise Investments web site.

- 6 The **Site Address** field defines the URL entered to browse the Site output. This can be a fully qualified path if the output target is filesystem or a virtual directory if the output target is ftp. Enter the following into the field: `http://127.0.0.1:9992/EI_Home`.
- 7 The optional **Home Page (URL)** field refers to the Site Explorer home page for the Site. Only Sites with Home Page URLs appear in Sites View in the Content Explorer and are available in the Site Explorer. Enter the following in the **Home Page (URL)** field:  
`http://localhost:9992/Rhythmyx/rxs_Home_cas`
- 8 Note: Currently Site Explorer only supports virtual sites (non-published Sites in your Rhythmyx directory); the Home Page (URL) must be the address of a virtual site.
- 9 The **Publishing Root Location** field is optional. It specifies the directory location where the files will be saved. For this example enter the following: `../EI_Home.war`. By specifying `..` you indicate the default location in the application server root. This location is `<Rhythmyx root>\AppServer\server\rx\deploy\`. If you want to publish to a different location, enter the address in place of `..`.
- 10 The next field is the **Publisher** field. This represents the Publisher that publishes to the Enterprise Investments site. Choose Localhost Publisher Default Port from the drop down list.
- 11 The next field is the **Status** field. This field indicates whether the status of the Site is active or inactive. Choose Active from the list.
- 12 The **Folder Root** field is optional. It is used with Site Folder Publishing and refers to the Content Explorer location of the Root Folder in the following format - `//Sites/<root folder name>` Enter the following text in the Folder Root field: `//Sites/EnterpriseInvestments`.
- 13 The **Global Template** field specifies the Global Template to use when rendering the Enterprise Investments Site. At this time you are not going to select a Global Template; leave the field empty.
- 14 The **Nav Theme** field is optional. It is used with Managed Navigation and is a theme indicator for stylesheet coding. You will not select a Nav Theme at this time, leave the field empty. For more information on Global Templating see *Creating Slots and Templates* (see page 107).
- 15 The final four fields in the Edit Site Properties page fall under the heading FTP Information. Leave all of these fields with their default information for this exercise. For detailed information on setting up an FTP Publisher, see *Configuring Publishing* (see page 291).

The fields in the FTP Information section are described below.

**IP Address** – the address where files are located for an FTP output. For this example, enter 127.0.0.1 in the IP Address field

**Port Number** - refers to the FTP port number

**User ID** - the User ID used to access the database

**Password** - the password used to access the database

- 16 Click [**Save**].  
The Enterprise Investments site is now registered in Rhythmyx.

## Creating Site Subfolders

Site subfolders are merely folders within folders but they contain all of the information and files for your Site. Folders can only be added by a user with administrative rights in the system. Subfolders can also contain additional subfolders within them. This establishes the hierarchy within the Site. The Site folder structure in the graphic below details the Enterprise Investments Site.

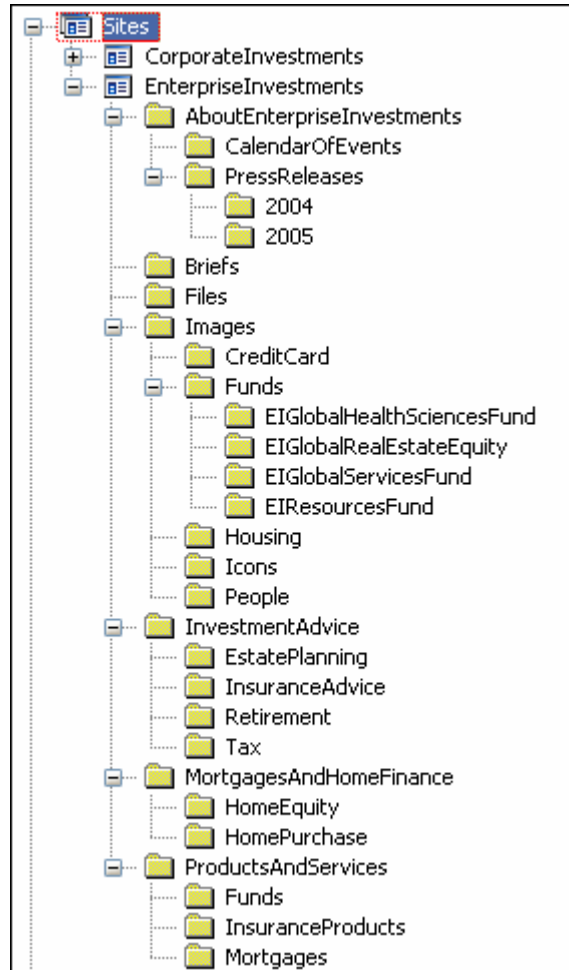


Figure 44: Site hierarchy in Content Explorer

With the creation of subfolders complete, you can start to set up the navigation for the Site. To review the steps for creating a new folder refer to Getting Started with Rhythmyx.

## Managed Navigation for the Site

Rhythmyx includes a Managed Navigation system, which makes it easy to add navigation elements to a web page. Managed Navigation is added once you have created the Site Root Folder and any Site Subfolders

Once implemented, Managed Navigation is fairly simple to use. It is based on three navigation Content Types, which are described below.

**NavTree** - Similar to a Navon, the NavTree Item resides at the root of a Site. A NavTree initiates the propagation of Navons to every subfolder in the Site. The NavTree Item is generally linked to the Site's Home Page Item.

**Navon** - Items used to create navigation menus including breadcrumbs, bottom, side, and top navigation, and site maps. Each Navon should be linked to a Content Item not used for Navigation (such as a Generic Page or a Category Content Item).

**NavImage** - Images used by Navons to replace text links for navigation elements. NavImages are also used by several Content Types to provide Image Links.

See the chapter *Managed Navigation* (on page 261) for more information on implementing and configuring Managed Navigation.

## Adding a NavTree to the Site Hierarchy

Adding a NavTree to the root of the Site hierarchy is the first step in setting up Managed Navigation.

The NavTree is responsible for propagating Navons through the Site's subfolders. Once you create the NavTree, Rhythmyx automatically propagates Navons for each subfolder you create in your Site. Note that a Site folder cannot already have a NavTree Content Item. If a Site root folder already contains a NavTree Content Item, Rhythmyx will return an error when you try to add the new NavTree Content Item to the Repository.

To create the NavTree:

- 1 Log into Content Explorer under the Admin Community for your Site. (You must have Admin access to create a NavTree.)
- 2 On the Content Tab, locate the Site root folder where you want to create the NavTree. Note that this folder cannot already have a NavTree Content Item.



- 3 Right click the Site root folder and choose New Item > NavTree from the popup menu. Content Explorer displays the Edit Content dialog.

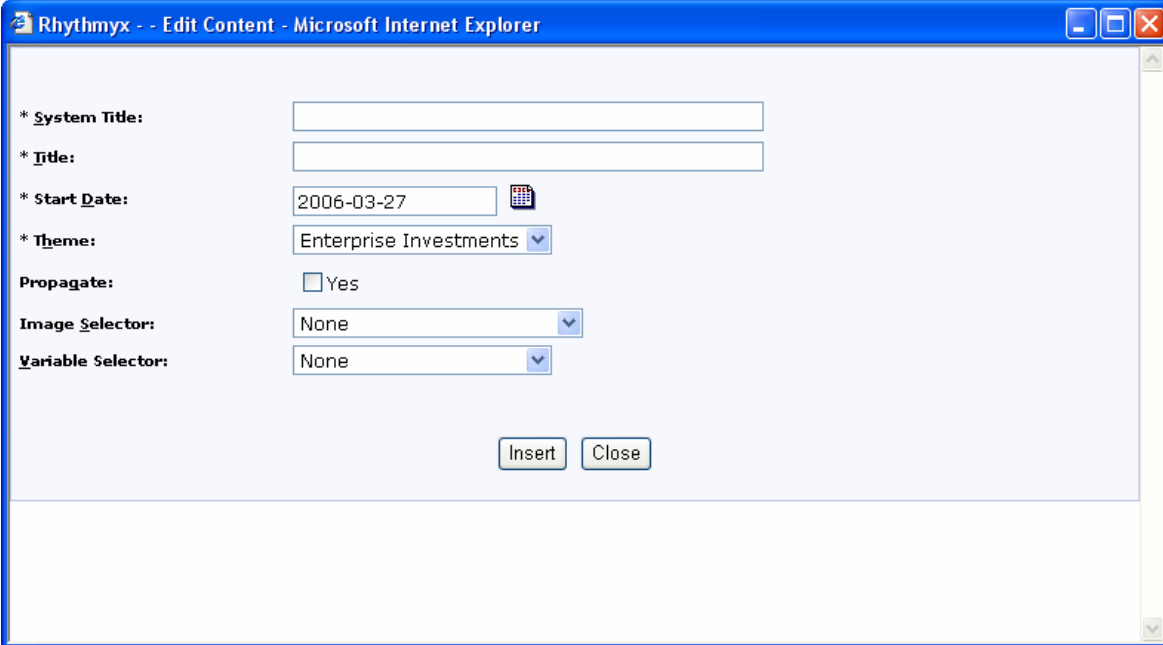


Figure 45: Content Editor

- 4 In the System Title field enter Enterprise Investments Internet Root. This field represents the title of the Content Item within the Repository.
- 5 In the Title field enter Enterprise Investments Home. This field represents the name of the NavTree that the end user will see in Content Explorer.
- 6 Leave the Start Date field with the default selection. This field represents the date the NavTree was created.
- 7 In the Theme field dropdown list select Enterprise Investments. This field represents the NavTheme that the Site will use. For now you will ignore this field.
- 8 Check "Yes" in the Propagate field. This will add Navons to any existing Site Subfolders. Rhythmyx will always add Navons to new Site Subfolders you add after adding a NavTree to a Site Root Folder.
- 9 Leave the Image Selector field with None selected. For now you will ignore this field.
- 10 Leave the Variable Selector field with None selected. For now you will ignore this field.
- 11 Click the [Insert] button.  
This will add the NavTree to the Repository.
- 12 Close the Content Editor.

---

## Defining Access to Folders using Access Control Lists (ACLs)

When a folder is created all users in Rhythmyx have access to the folder by default, defined by the “Everyone” category. The Security tab is where you can set permissions for particular Roles, Users and Communities.

To specify Site access for particular users or Roles, use the Security tab in the Edit Folder dialog. You can also define Access Control Lists (ACLs) for specific Folders within the Site to specify user access to those specific Folders.

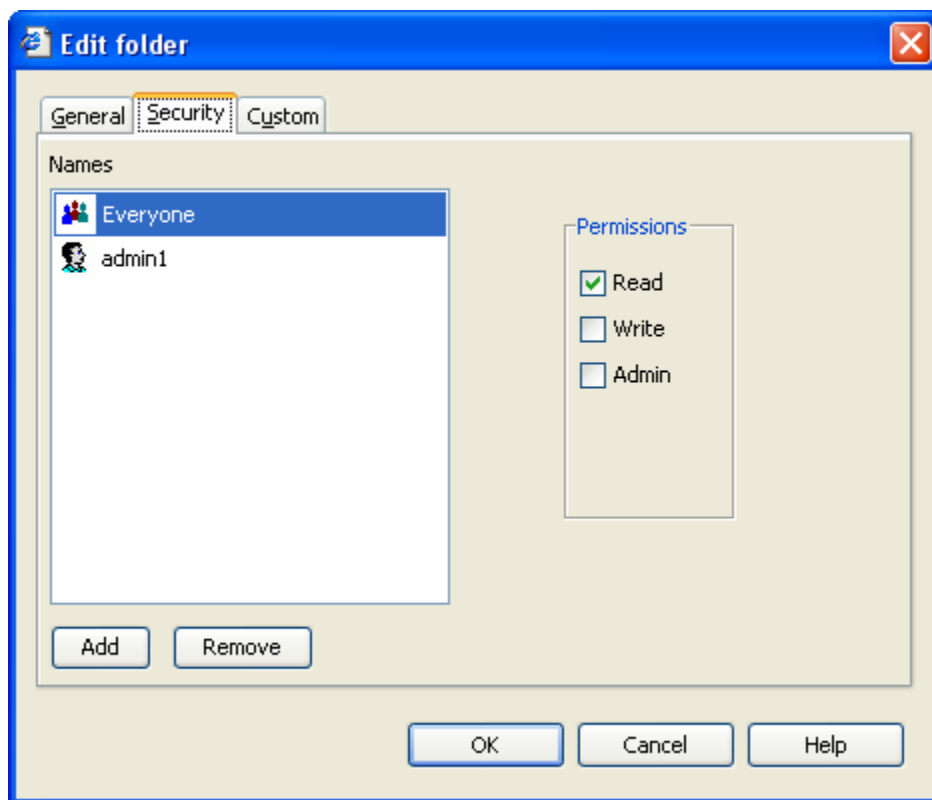


Figure 46: Edit folder

In the following procedure you are going to add EI\_Admin\_Members and EI\_Members to the list of roles that have access to the Site. We are also going to grant the Editor Role Admin permissions for the Briefs folder.

Note: the example we use in the following exercise is already included as part of your FastForward Rhythmyx installation. Use your implementation plan or statement of work to determine the data to use.

To specify access to a specific folder:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Expand the Sites node in the Content tab, then right-click the Briefs folder.

- 3 In the popup, select Properties. The Edit Folder dialog opens.
- 4 Click the Security tab of the Edit Folder dialog. This tab is where you can add or remove Users and/or Groups from the Enterprise Investments site. Under the Names heading you will see all users and groups that are currently assigned folder rights.

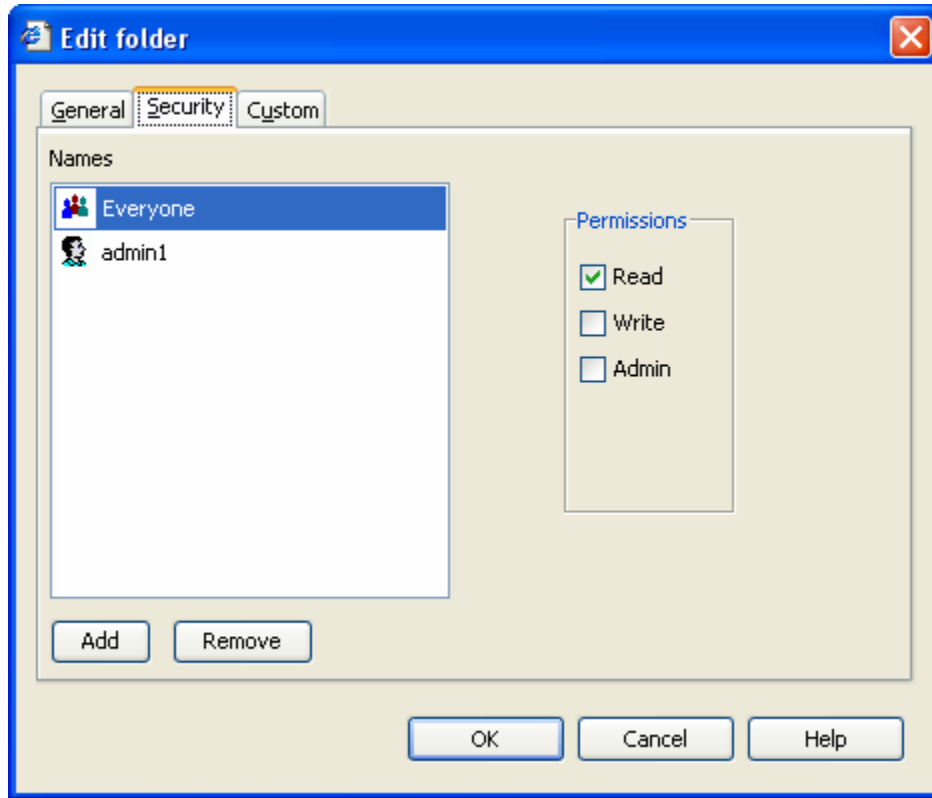


Figure 47: Edit folder dialog

By default, the Everyone role is issued Read permissions. This allows users not previously defined to have rights to see the contents of the new Folder. The user who created the Folder is issued Read, Write, and Admin rights to the Folder. In this example that user is admin1. This allows the creator to see the contents of the Folder, add Items to the Folder, and change the Folder's properties. In addition, the Admin permission allows the user to delete the folder.

The fields in the Permissions section are described below.

**Read** – allows the user to view the folder and its contents. Does not allow users to move, copy, or add contents to the folder. Lets users copy but not move contents in the folder to another folder. Lets users view folder properties.

**Write** – allows the user to view, copy, and move the folder, but not delete it, and to view, copy, move, or add contents. Lets users view folder properties..

**Admin** – grants the user all Write permissions and enables them to delete folders, sub-folders and to edit all folder and sub-folder properties.

Permissions can be added for additional users or Roles. Click the [Add] button and select the additional user or Role you want to add to the Folder ACL using the Folder ACL List Entry Editor.

- 5 Add the EI\_Admin\_Members role and the EI\_Members role to the list of Current ACL Entries on the right. To do this press the Ctrl + Shift keys to select both EI\_Admin\_Members and EI\_Members from the Cataloged Entries list. Once both are selected, click the [Add] button.
- 6 Now you need to assign permissions to the newly added roles. Select EI\_Admin\_Members and notice that the Read checkbox is selected. The Read checkbox is selected by default for all roles you add to the Current ACL Entries list. Select the Write and Admin checkboxes for EI\_Admin\_Members.
- 7 Click [Add] on the Edit Folder dialog. The Folder ACL List Entry Editor opens.

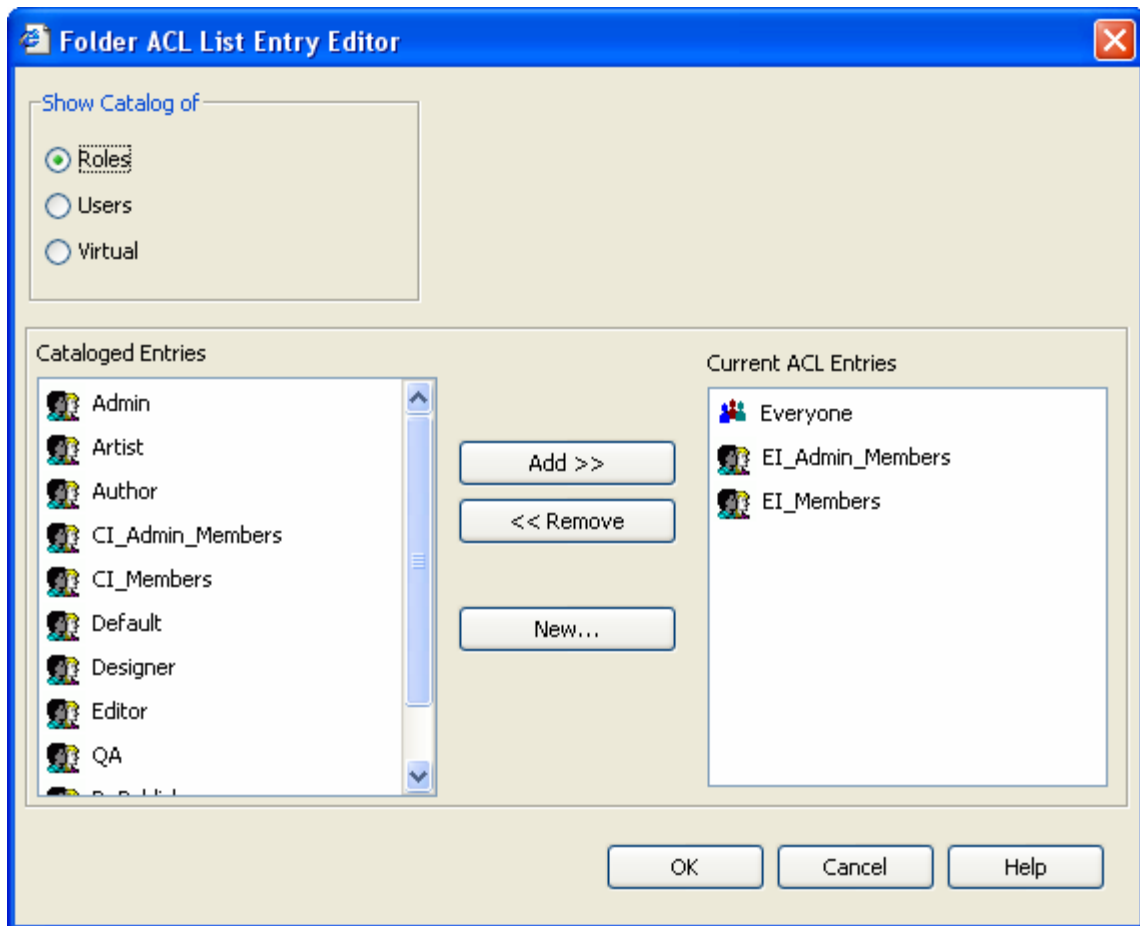


Figure 48: Folder ACL

- 8 In the "Show catalog of" section, select Roles. The options indicate how you would like to assign access:
  - Roles – lists all system-defined Roles in the Cataloged Entries section of the dialog
  - Users - lists all system-defined users in the Cataloged Entries section of the dialog
  - Virtual – displays the groups Everyone and Folder Community. Everyone includes all users logged into the system. Folder Community includes all members of the folder creator's Community.

- 9 Depending on which selection you make, the list of system-defined Roles, Users, or Virtual appears in the Cataloged Entries column.
- 10 Select *Editor* from the Cataloged Entries column and click [Add].
- 11 Click [OK].  
The Edit Folder dialog now displays the newly added Editor Role.

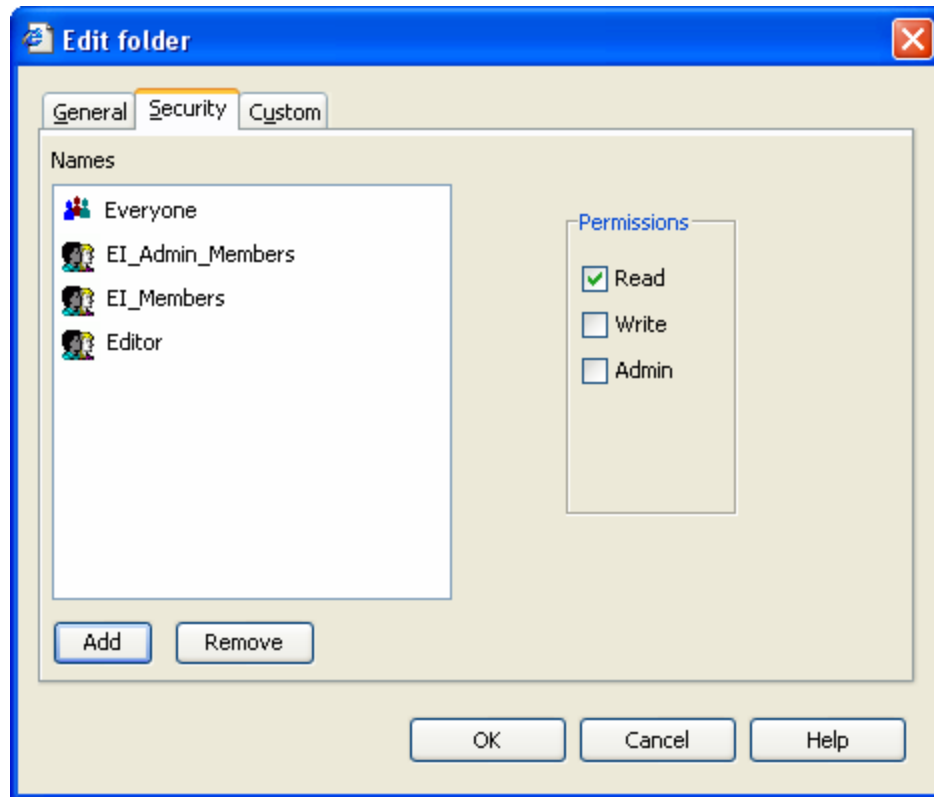


Figure 49: Edit folder dialog with Editor Role added to ACL

- 12 Select Editor from the Names column and select the Admin checkbox.  
Note that the Editor will have Admin permission for the Briefs folder only.
- 13 Click [OK] to save the new permission setting.

Folder permissions can be assigned on all folder levels. A user could have Write permission on a specific subfolder but not the folder as a whole. For example, the user could have Write access to the CalendarOfEvents folder but not the Press Releases folder. However, if the user is granted Write permission on the AboutEnterpriseInvestments folder, that permission will carry over to any subfolders contained therein.

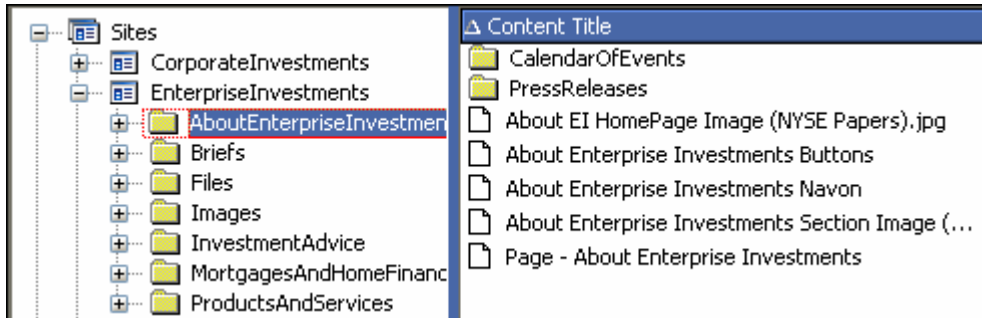
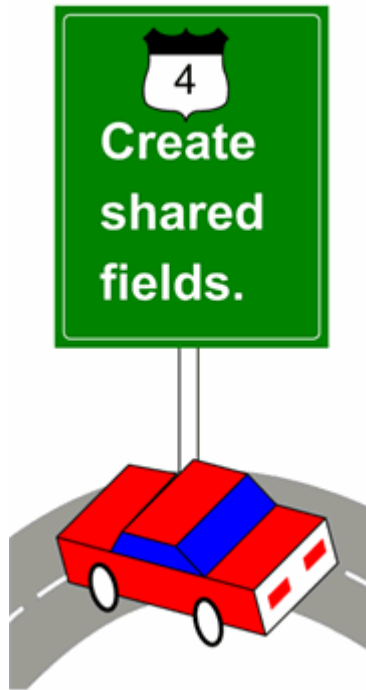


Figure 50: Site hierarchy in Content Explorer

# Creating Shared Fields



Rhythmyx Content Types include three types of fields:

- System - System fields are supplied by Rhythmyx and can appear in all Content Types. An example of a system field is the Community associated with a Content Item, `sys_communityid`.
- Shared - Shared fields are user-defined. You can include them in multiple Content Types. Shared fields are always defined as field sets in shared field objects. In general, several shared field sets are included in a single shared field set object.
- Local - Local fields are user-defined for a specific Content Type. You must open a Content Type object to view its local fields.

This section emphasizes shared fields rather than system or local fields because you do not create system fields, and local field creation is discussed in the chapter *Creating Content Types* (see page 209). Since the details of creating local fields are identical to the details of creating shared fields, when the Creating Content Types chapter instructs you to enter properties for a field, it refers back to this chapter.

When you create a Content Type, you have the option of including fields from shared field sets. Therefore, shared field sets allow you to create common fields once, but use them in multiple Content Types. Shared fields also enable you to create modified versions of system fields (you should not modify system fields directly since they are overwritten on upgrade).

Store your shared fields in separate sets that group them by their function to make it easier for other implementers to determine which shared fields they need when they create a new Content Type. FastForward includes three shared field sets. Each shared field set includes fields that tend to be used repeatedly in Content Types for a similar function. For example, the *shared* Field Set includes general fields, such as *displaytitle* and *body*, while the *sharedimage* field set includes fields for uploading images, such as *img1\_filename* and *img1\_type*.

The topics in this chapter walk you through the procedure of using the Workbench's Field and Field Sets Editor to create FastForward's *shared* (see page 77) and *sharedimage* (see page 78) field sets. We have chosen to demonstrate these two field sets because most Rhythmyx systems include general fields that are used repeatedly and include one or more Content Types that upload images. The FastForward *shared* and *sharedimage* field sets model the types of fields you might want to include in a general shared field set and in a shared field set for uploading images.

Since the process of creating a shared field set involves creating a group of fields, the topics will also focus on the creation of certain fields, and why certain values are chosen for the properties in these fields. FastForward also includes two other shared field sets: *sharedbinary* and *sharedcategory*. For instructional purposes, we are not demonstrating how to create these two field sets in this chapter, but you may read more about them in the *Rhythmyx Technical Reference*.

---

Note that you already have the FastForward *shared* and *sharedimage* field sets on your system as part of Rhythmyx so we are using them in this chapter for demonstration purposes only. You would use the information in your implementation plan as a substitute for the data used in the instructions in this chapter (or copy our steps but use different names for the field sets).

---



---

## shared Field Set

The *shared* field set includes fields that are used by many Content Types. Most Content Types include a body field to store their main body content, and a field that stores a summary of the content. In the *shared* field set, these fields are *body* and *callout*. Most Content Types also require a title that is displayed to users. In the *shared* field set, this field is *displaytitle*. Some Content Types require a file name that is used when the Content Item is published. In the *shared* field set, this field is *filename*. By default, this field is hidden from users because it is used for processing and not displayed in Content Items.

In the sample CMS that we are creating, all Content Items should be able to store search words and phrases that are not part of any displayed fields in the Content Item. In the *shared* field set the fields *keywords* and *description* store words and phrases for searching on.

The *webdavowner* field stores the user who has a lock on the Content Item when content is uploaded through Rhythmyx's WebDAV feature. By default, this field, which is used for processing only, is hidden from users. It is included in the *shared* field set so that if implementers choose to enable WebDAV in their system, this required field is readily available in Content Types. However, this document will not cover WebDAV. See the document *Implementing WebDAV in Rhythmyx* for information about WebDAV.

The ***shared Field Set specification*** (see page 446) outlines the *shared* field set. The sections of the chapter that follow will explain the purpose of the information included and will refer to this table.

In this chapter, to demonstrate ways of adding default values, we give some fields in the *shared* field set default values that are not included in FastForward. We give the *displaytitle* field the default value *sys\_title* and the body field the default value "Enter body here". In addition, we add a *doc\_type* field to the *shared* field set. The *doc\_type* field is not included in FastForward's *shared* field set, but here we include it to demonstrate how to implement a list control. The specification for our *doc\_type* field is:

Name	Label	Description	Control	Data Type/ Storage Size	Default Value
doc_type	Type	The type of document.	sys_DropDownList	text	50

---

## sharedimage Field Set

The *sharedimage* field set includes fields that are used by Content Types that upload image files. Two versions of the same fields are included, one for uploading full size images (the full size image fields are prefixed with *img1*) and one for uploading a thumbnail graphic of the same image (the thumbnail image fields are prefixed with *img2*). Since many systems do not require the thumbnail image, the *img2* fields are hidden by default.

Both versions include a field for uploading the image. The upload fields are named *img1* and *img2*. *img1* and *img2* use the *sys\_file* control, which is necessary for uploading a binary file into a field in Rhythmyx. As the topics that follow explain, the *sys\_file* control uploads the image file and lets the *img1* and *img2* fields store its binary data.

The *img1\_filename*, *img1\_size*, *img1\_type*, *img1\_ext* and *img2\_filename*, *img2\_size*, *img2\_type*, and *img2\_ext* fields in the *sharedimage* field set store metadata that is automatically extracted when the *sys\_file* control uploads the *img1* or *img2* file. Other metadata is also extracted, and you can include fields for storing these values in your shared image upload field set if you want to include them in Content Types in your system.

In FastForward most of the metadata fields are extracted by the *sys\_fileInfo* and *sys\_imageInfoExtractor* extensions. The *sys\_fileInfo* java extension (java plugin) is automatically included as a dependency of the content type when the *sys\_file* control is used. *sys\_imageInfoExtractor* may be included as a pre-processing extension if you want to store height and width metadata as well. These extensions require certain names for fields in which they store data. See the topics *sys\_fileInfo* and *sys\_imageInfoExtractor* in the *Rhythmyx Technical Reference* for more information about this extension. The topic ***The Image Content Editor*** (see page 236) in the chapter *Creating Content Types* for more information about using *sys\_fileInfo* and *sys\_imageInfoExtractor* with a Content Type.

The ***sharedimage Field Set specification*** (see page 447) outlines the *sharedimage* field set that you will use and is included for your reference. The sections of the chapter that follow will explain the purpose of the information included and will refer back to this table

# The Rhythmyx Workbench Field and Field Set Editor

All fields, shared, local, and system, are viewed in the Rhythmyx Workbench's Fields and Field Sets Editor. Shared and Local fields are also created and edited in the Fields and Field Sets Editor, and System fields are edited in the Fields and Field Sets Editor. Depending on what type of field you are working with, the Fields and Field Sets Editor is accessed differently and appears with slight variations. The following graphic shows the version of the Fields and Field Sets Editor for shared fields.

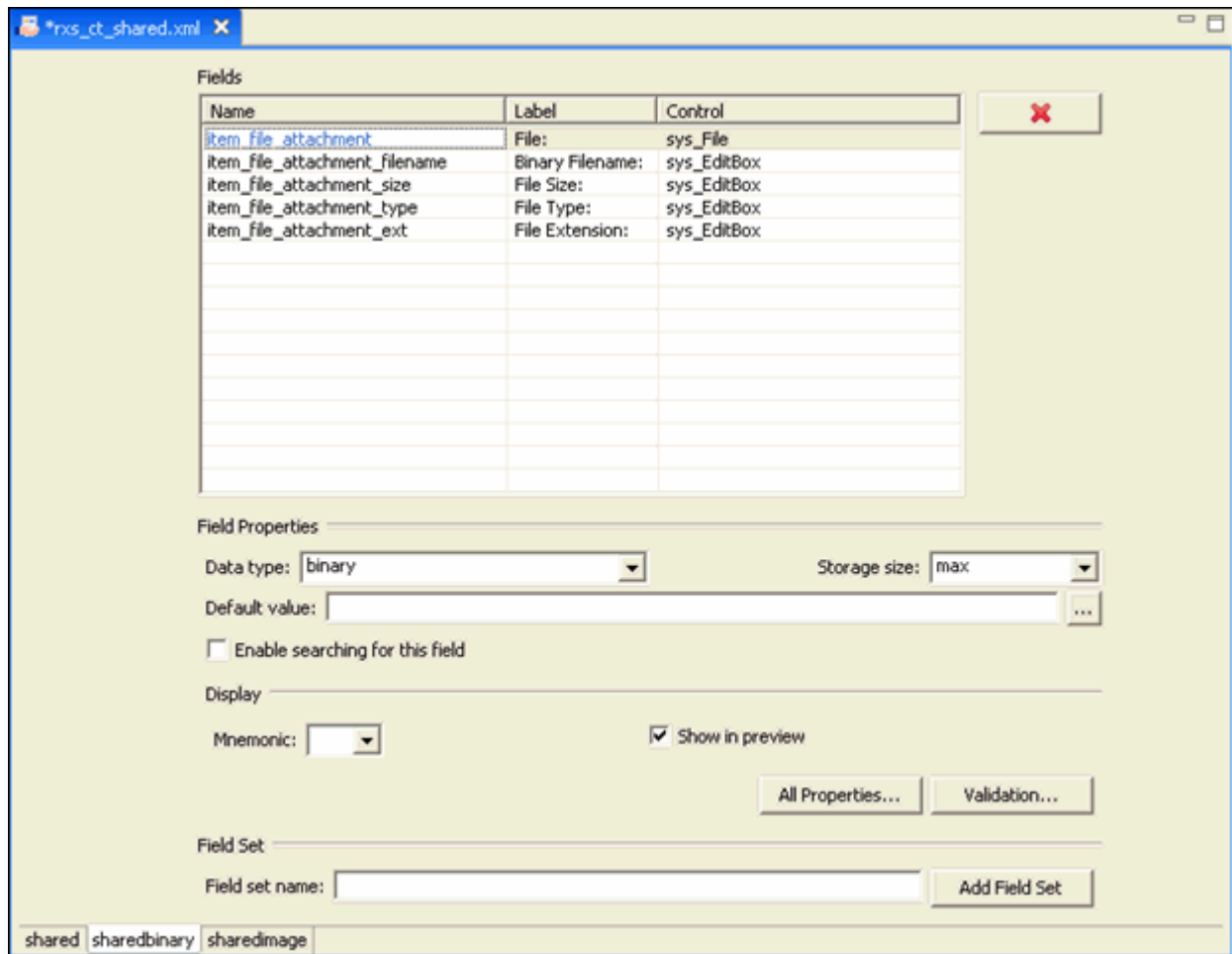


Figure 51: Field and Field Sets Editor

You can access shared, system, and local fields in the Rhythmyx Workbench's Content Design view.

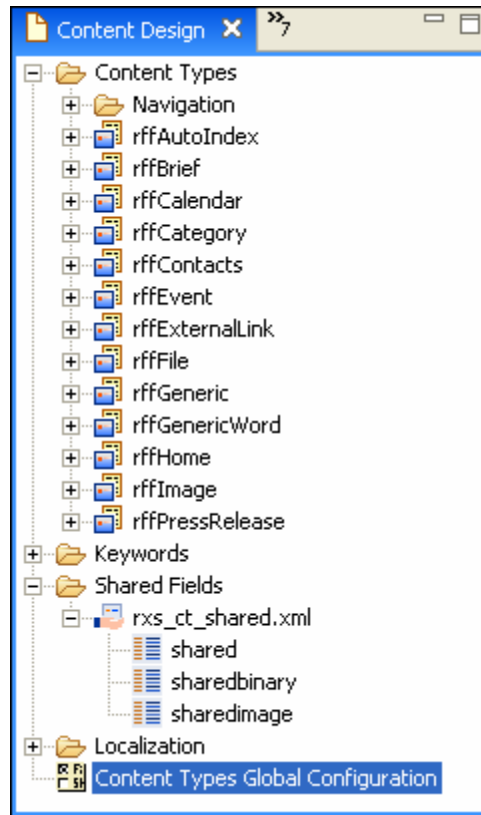


Figure 52: Content Design view

Shared field objects and the names of the shared field sets that they include are listed under the Shared Fields folder in Content Design view. To create a shared field object, right-click on the shared field folder and choose *New > Shared Field File*.

To view or modify the system field set, right-click on the Content Types Global Configuration folder and choose *Open*. The Fields and Field Sets Editor appears as it does for editing shared fields, except the delete button for deleting fields and the control for adding an additional field set are not present.

To create or edit a local field, create or open the Content Type that includes (or will include) the field. The Fields and Field Sets Editor appears within the Content Type Editor. To view the Fields and Field Sets Editor in this format, see **Creating Content Types** (see page 209).

For complete information about the Fields and Field Sets Editor for each type of field or field set, see the *Rhythmyx Workbench Online Help*.

---

# Creating Shared Field Sets and Configuring Fields

This section covers the procedure for entering the *shared* and *sharedimage* field sets. As we enter the field sets, we will demonstrate how to choose values for some of the individual properties and fields within specific fields in the field set. Since the process for choosing values for local field properties is identical to the process for choosing values for shared fields, the information given here applies to local fields as well as shared fields.

The topics in this section will explain the purpose of the fields' properties and discuss the impact of choosing different values for these properties. The topic *Implementing the "shared" Field Set* (see page 81) will cover common choices for frequently used fields, while the topic *Implementing the "sharedimage" Field Set* (see page 93) will discuss the required and optional fields for uploading images. The topic *Implementing the "sharedimage" Field Set* will also emphasize how other field properties may require special values because the data stored is binary.

In all topics, some emphasis will be given to choosing the appropriate control for displaying the field in a Content Editor. The topic *Implementing a List Control* (see page 90) is included to demonstrate the implementation of a list type control.

---

*Note: You cannot create a shared field object that includes shared and sharedimage field sets, since they already exist in FastForward. Instead, create a similar shared field object and shared fields sets that are included in your implementation plan or copy our steps but give your components different names.*

*You must also give your shared fields different names than those used in FastForward. In general, we recommend giving fields in different shared field sets different names.*

---

## Implementing the "shared" Field Set

This topic shows you how to enter the *shared* Field Set by focusing on the entry of the *displaytitle* and *body* fields. Walking through the process of filling in these fields demonstrates some important concepts, such as the difference between the *Name* and *Label* fields, the functions of the *sys\_EditBox* and *sys\_EditLive* controls, the use of a *Default Value* for a field and the function of the *Mnemonic*. At the end of the topic, you are instructed to enter the remaining fields in the *shared* field set, using the information you have learned in this topic and the details provided in the *shared Field Set specification* (see page 446). You are also introduced to process of entering the next shared field set in the shared field set object into the editor.

---

*Note: You cannot create a shared field object named `rxs_ct_shared.xml`, since it already exists in FastForward. Instead, create a shared field object included in your implementation plan or copy our steps but give your shared field object a different name. In addition, give your shared field set a different name the FastForward name used in the following procedure.*

---

To create the *shared field set*:

- 1 Open the Rhythmyx Workbench's Content Design view. In the Menu bar, choose *File > New > Other*.

The Select a Wizard dialog opens.

- 2 Choose *Shared Field File* and click [**N**ext].

The New Shared Field Definition File wizard opens.

- 3 In **Shared Definition file name**, enter *rxs\_ct\_shared.xml*. This is the name of the file that will hold the shared field set definitions. In the Rhythmyx Workbench, we refer to the file as an object.
- 4 In **First group name**, enter *shared*.

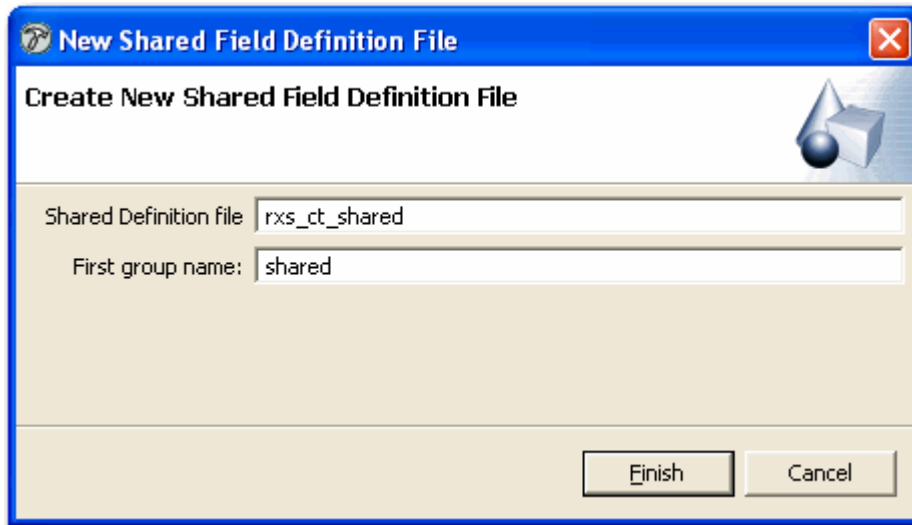


Figure 53: New Shared Field Definition File wizard

- 5 Click [**F**inish].

The Shared Field Definition File editor opens in the Workbench. At this point, it includes a single tab, which holds the *shared* field set. The Fields table for the shared field set is empty.

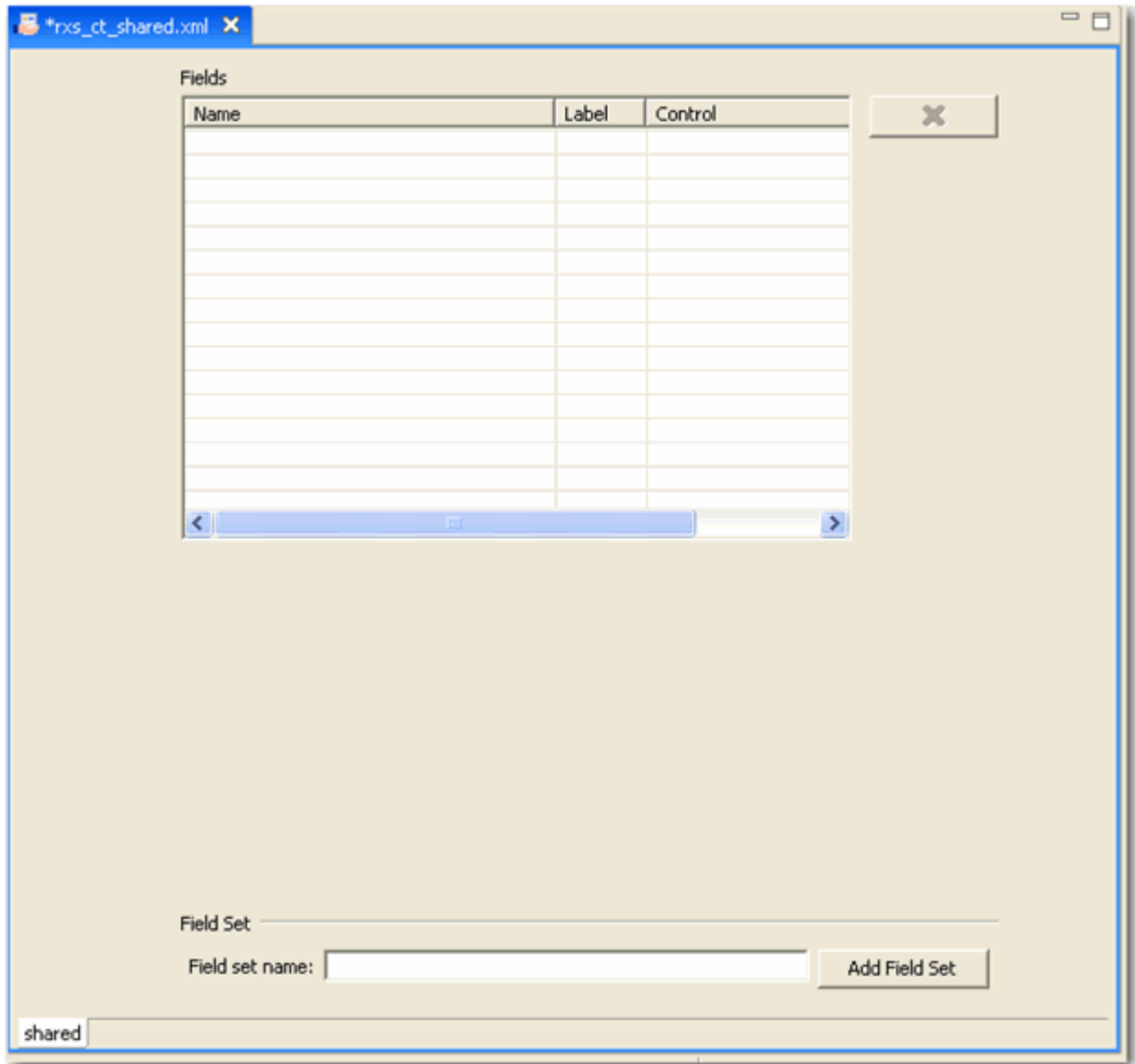


Figure 54: Shared Field Definition File editor

- 6 Begin by entering the *displaytitle* field.
  - a) In the Fields table, under Name enter *displaytitle*. *displaytitle* is the internal name that Rhythmyx uses for the field. It is best practice to enter all field names in lower case. The editor automatically enters *Displaytitle* under Label because the internal Name is frequently used as the Label. However, in this case change the entry under Label to *Title*.

- b) In the Fields table, under Control choose `sys_EditBox`. The `sys_EditBox` control presents a one-line box in which users can enter unformatted data. Since the displaytitle should be no more than one line of plain text, the `sys_EditBox` control is appropriate.

Investment Advice

- c) Click on the *displaytitle* row to display Field Properties in the lower part of the editor.
- d) Under Field Properties, in Data Type, leave the default value of *text*.
- e) Under Field Properties, in Storage size, change the default value of 50 to 512. Although it is unlikely that anyone will enter a value that uses this amount of space, it takes into account foreign characters that may require additional bytes and databases that may require extra space to store characters.
- f) In the FastForward version of this field, a Default value is not entered. Here, we will enter a value that will cause the *displaytitle* field to automatically take the value of the Content Type's `sys_title` field.
- Under Field Properties, click [...] beside Default value.
  - Choose *Other value*.

The Value Selector dialog opens.

- In the Type drop list, choose *Single HTML Parameter*. System fields appear in the Choices list.
- In the Choices list, choose *sys\_title*. If it does not appear, type it into the Value field.

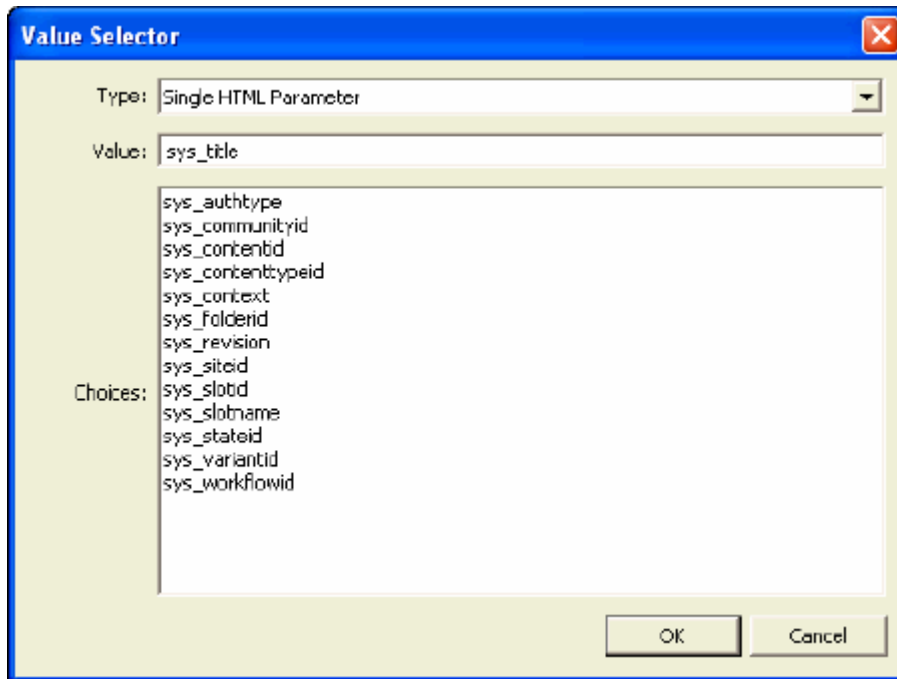


Figure 55: Value Selector

- Click [OK]. The Value Selector closes and `PSXSingleHtmlParameter/sys_title` appears in the Default value field.




For more information about the Value Selector, see the *Selecting Values* topic in the *Rhythmyx Workbench Online Help*.

- g) Under **Field Properties**, leave **Enable searching for this field** checked because users will want to search on the title of the field.
  - h) Under **Display**, choose a **Mnemonic** value (a letter) from the drop list. The user must enter ALT + [value] to access the field without using the mouse. The drop list only displays letters that have not yet been used for the *shared* field set. Since the name of the field will appear as Label to users, choose *L*.
  - i) Under **Display**, leave **Show in preview** checked so that users can see the *displaytitle* field when they preview templates of Content Items that contain it.
  - j) Do not click the **[All Properties]** button. You will leave the default values of the properties that it accesses.
- 7** Next enter the *body* field.
- a) In the next row in the **Fields** table, under **Name** enter *body*. *body* is the internal name that Rhythmyx uses for the field.  
  
The editor automatically enters *Body* under **Label** because the internal **Name** is frequently used as the display label that appears next to the field in Content Editors.  
  
Change the value of **Label**<sup>2</sup> to *Body*:
  - b) Under **Control** choose *sys\_EditLive*. The *sys\_EditLive* control uses Ephox's EditLive for Java (ELJ) rich text editor. No additional configuration of this control is necessary unless you want to customize it.

---

<sup>2</sup> In general, we think of the value that we enter for Label as the display name of the field that we want to appear to users. This explanation is usually true. However, in actuality, the Label is used as a key for looking up the translation value of the Label field for different Locales in the tmx file. For more information about Locales and the tmx file, see the Internationalizing and Localizing Rhythmyx.

The `sys_EditLive` control is appropriate for the *body* field because the control allows users to enter information in a WYSIWYG format but stores it in HTML markup, which enables text formatting and insertion of graphics and links. Use of the `sys_EditLive` control assumes that a browser, which can interpret the markup, will display the information to users when it is published. See the *Rhythmyx Technical Reference* for information about customizing the `sys_EditLive` control. (for details about the standard features and Rhythmyx features of ELJ, click the help button  in the control). This control works with all browsers that Rhythmyx supports.

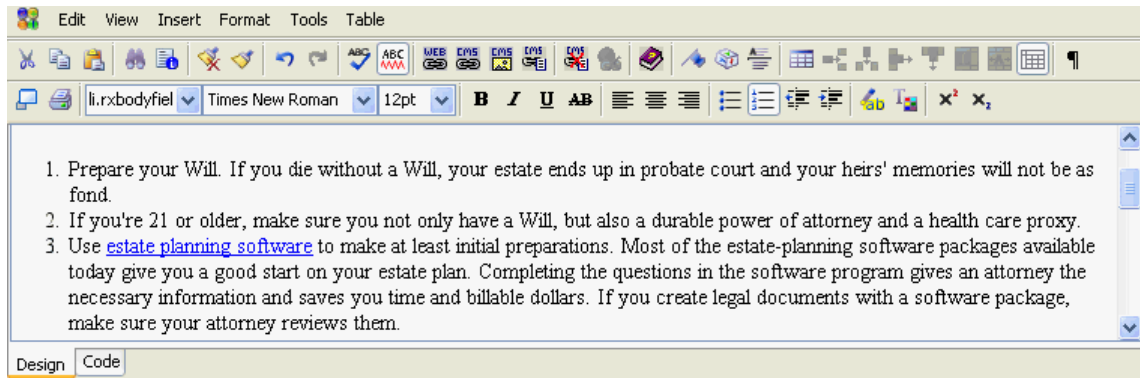


Figure 56: `sys_EditLive` control

- c) Click on the row for the *body* field to make sure you are displaying the Field Properties for this field.
- d) Under Field Properties, in Data Type, leave the default value of *text*. Note: If you are using the `sys_EditLive` control, you must use the Data Type *text*.
- e) Under Field Properties, in Storage size, change the value to *max*. The value *max* indicates that the storage size is the maximum allowed for the the text Data Type in the database used. (*max* is an appropriate choice for the *body* field because users may enter relatively large amounts of text in the field.)
- f) In the FastForward version of this field, a Default value is not entered. Here, under Field Properties, we will enter the Default value *Enter body here*. Since *Enter body here* is a literal value, you can type it into the edit box provided. You could also click [...] to choose a user-defined function to enter the default value or enter the value with a method chosen from the Value Selector.
- g) Under Field Properties, leave Enable searching for this field checked.
- h) Under Display, in Mnemonic, choose a key for accessing the field. Since the *body* field begins with the letter B, choose *B*.
- i) Under Display, leave Show in preview checked so that users can see the *body* field when they preview Content Item templates that contain it.

The editor filled in for the *body* field appears as:

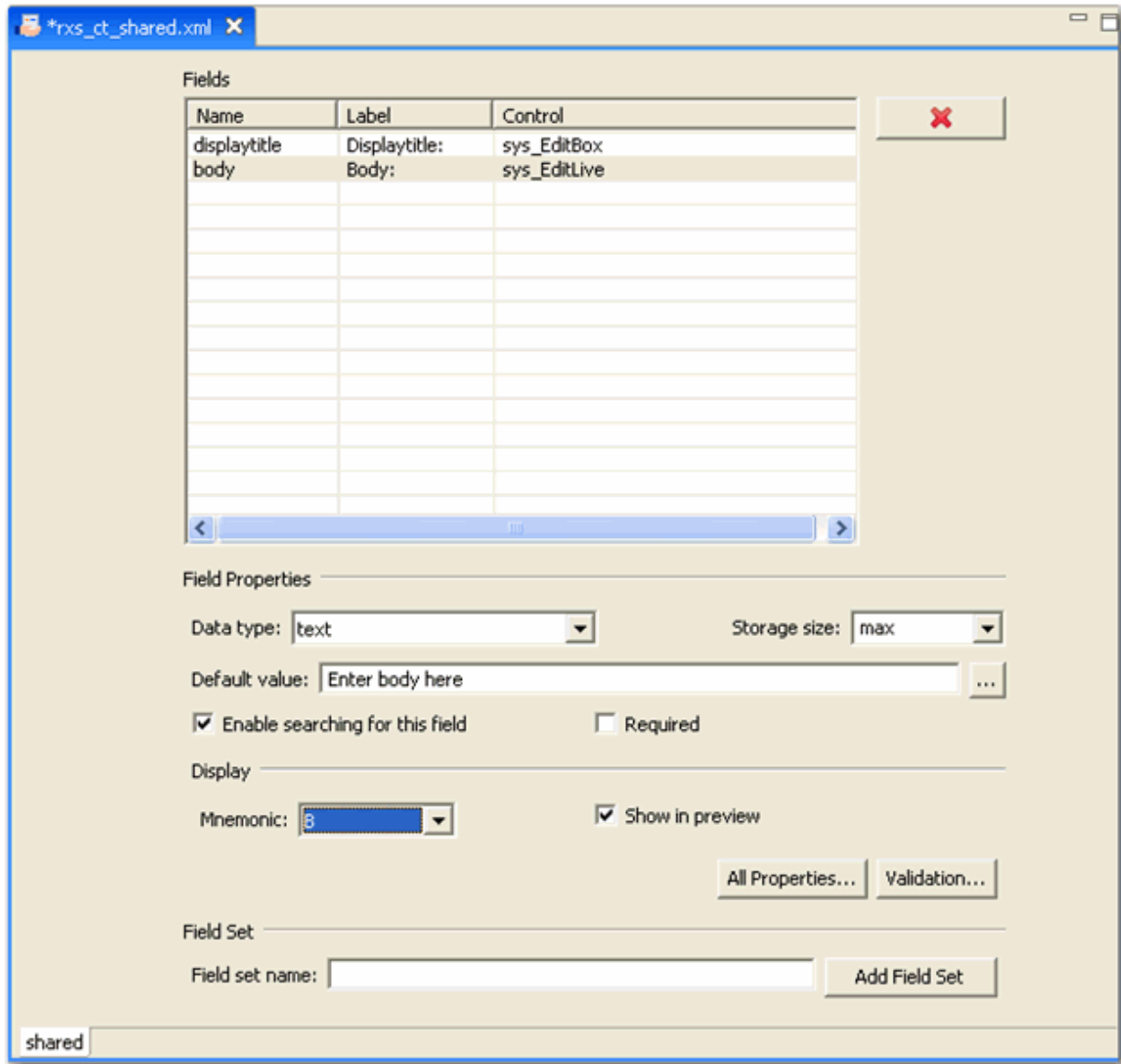


Figure 57: Fields and Field Sets Editor

- 8 Click [**All Properties**] to enter additional properties for the *body* field in the Field Properties dialog.
- 9 In the Field Properties dialog several fields are filled in from the previous screen; we will not modify these fields. The following steps refer to fields that appear only on this screen.
  - a) Leave **Treat data as binary** unchecked because we do not expect users to enter large enough amounts of data to require binary storage in this *body* field. If we did anticipate that extremely large amounts of data would be entered, we would check **Treat data as binary** so that Rhythmyx would know to store the information as non-character data in a column with additional storage space.

- b) Leave the default value of *Body*: in **Error label**. Rhythmyx will concatenate any validation errors with the Error Label value and display them in the Content Editor.
- c) **Show in summary** is disabled because the *body* field is not a child field set. Note: Shared fields cannot be child field sets; **Show in summary** is included because the dialog is also used for local fields.
- d) **Show "clear field" check box in binary control** is disabled because the *body* field does not use a control that uploads binary files.
- e) In this case, check **Index formatting, such as HTML tags**, because your users are knowledgeable enough to search for information that may be included in certain markup tags in the *body* field. For example, if a user wanted to search through Heading 3 titles, checking this option allows the user to search for the tag <H3> to view all of these titles.
- f) **Include in full text multi-field query** is checked to allow users to use the *body* field when creating searches in Content Explorer. When you checked **Index formatting, such as HTML tags** it became read-only because if you choose to index formatting, the system assumes that you want to include the field in user search queries.
- g) Do not check **Punctuation is part of word**. This is used for fields such as filenames that commonly include punctuation in words. If you check it for the *body* field, users might fail to find terms during exact searches if the terms have a comma or period following them.

The Field Properties dialog filled in for the *body* field appears as:

The screenshot shows the 'Field Properties' dialog box for the 'body' field. The fields are filled as follows:

- Field name: body
- Data type: text
- Storage size: max
- Default value: Enter body here
- Treat data as binary:
- Display section:
  - Label: Body:
  - Mnemonic: B
  - Error label: Body:
  - Control: sys\_EditLive
  - Show in summary:
  - Show in preview:
  - Show 'clear field' checkbox in binary control:
- Search section:
  - Allow this field to be searched:
  - Index formatting, such as HTML tags:
  - Include in full text multi-field query:
  - Punctuation is part of word (use for filenames, etc.):

Buttons at the bottom: Read Only..., Visibility..., Validation..., Transformations..., OK, Cancel.

Figure 58: Field Properties editor

- 10 Click [OK] to return to the Fields and Field Sets editor.
- 11 Proceed to enter the *filename*, *keywords*, *callout*, *description*, and *webdavowner* fields. Use the values indicated in the *shared Field Set specification* (see page 446). Leave the default values for any properties that are unspecified or use the information given to you in this topic. In this exercise, do not add visibility rules for hiding the *filename* and *webdavowner* fields as specified; the next section will demonstrate how to do this in the topic *Adding a Field Visibility Rule* (see page 100).

- 12 Add the *doc\_type* field in the next available row. Follow the instructions in the topic *Implementing a List Control* (see page 90).
- 13 When you have finished adding the *shared* field set, you can add the *sharedimage* field set as an additional field set within the same shared field set file. To add the *sharedimage* field set, you fill in the information under **Field Set** at the bottom of the Fields and Field Sets editor.  
  
See *Implementing the "sharedimage" Field Set* (see page 93) for information about entering this field set.

---

Note: Quick field creation is the process of entering a name for a field and pressing <Enter> and then using the default properties that the Workbench enters. You always have the option of editing the default properties and adding others. The default properties entered are:

Label: Same as Name, except capitalized and followed with ":",

Control: sys\_EditBox

Source: Local

Data type: Text

Storage size: 50

Enable searching for this field: checked

Show in preview: checked

Include in full text multi-field query: checked

---

## Implementing a List Control

To illustrate the process of implementing a list control, we will show how to enter and configure the *doc\_type* field in our *shared* field set. Here, we repeat the *doc\_type* field's specifications:

Name	Label	Description	Control	Data Type/ Storage Size	Default Value
doc_type	Type	The type of document.	sys_DropDownList	text	50

After entering the *webdavowner* shared field in the Fields and Field Sets editor, complete the following steps to enter the *doc\_type* field.

- 1 In the **Fields** table, under **Name** enter *doc\_type*. *doc\_type* is the internal name that Rhythmyx uses for the field. The editor automatically enters *doc\_type* under **Label** because the internal **Name** is frequently used as the **Label**. Change the **Label** value to *Type*.

- 2 Under control choose `sys_DropDownSingle`. The `sys_DropDownSingle` control lets users choose one option from a list of predefined choices:

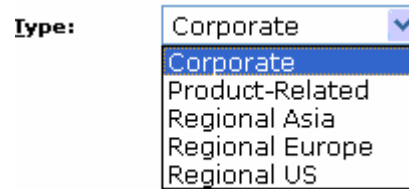


Figure 59: `sys_DropDownSingle` control

- 3 After choosing the drop-down single control, you must populate it with choices that a user can choose from.
- Click [...] beside the control to open the Control Properties editor.
  - Click the Choices tab.
  - To use a Keyword's choices in a drop down control, leave the **Use a Keyword** radio button selected and choose *FF Press Release Type* in the **Name** drop list.

The **Default values** box displays the Keyword Choices.

Note: See the section *Creating and Using Keywords* (see page 273) in the chapter *Creating Content Types* for information about setting up your own Keywords and Keyword Choices through the New Keyword Wizard and Keyword Editor.

- Initially display a dummy value in the drop list so that users are forced to choose a type. At the bottom of the dialog check **Display text for empty entry**. Enter *Choose type* in **Label** and **Value**. In **Include**, choose *Only if Null*, and in **Sort order**, choose *First*. This instructs a Content Editor to display *Choose type* as the first value in the drop list if no value is chosen for *Type*.

For more information about filling in the Display Control Properties dialog including additional methods of entering choices, see *Maintaining the Control Associated with a Field* topic in the *Rhythmyx Workbench Online Help*.

**Control Properties (sys\_DropDownSingle)**

Control Choices

Use a keyword

Name:

Retrieve from table

Data source:

Table:

Label column:

Value column:

Retrieve from xml application

URL:

Define choices for this control only

Label	Value

Default values:

Corporate  
Product-Related  
Regional US  
Regional Europe  
Regional Asia

Sort order

Ascending  
 Descending  
 User specified

Display text for empty entry

Label:

Value:

Include:

Sort order:

OK Cancel

Figure 60: Control Properties dialog, Choices tab

- e) Click [OK] to return to the Fields and Field Sets editor.



- 4 In **Data Type**, leave *text*.
- 5 The **Storage size** for *text* is automatically set to 50.
- 6 Since you set the default value for the field when you filled in the values for the drop down control, do not enter a **Default value**.
- 7 Leave **Enable Searching** for this field checked, since users may want to search on the value of the region.
- 8 Since users will have to manually choose a value for the field, choose the **Mnemonic T** from the drop list.
- 9 Check **Show in preview** so that users can see the region in any preview templates that include it.
- 10 Leave the default values of the properties in the Field Properties dialog (do not click the [**All Properties**] button).  
You have completed adding the *shared* field set to your shared.xml shared field object.
- 11 To add the *sharedimage* field set to the rx\_s\_ct\_shared.xml shared field object, do not close the editor. Instead proceed to the next section, **Implementing the "sharedimage" Field Set** (see page 93).

## Implementing the sharedimage Field Set

The *sharedimage* field set is part of the same shared field set object as the *shared* field set. To begin entering it, fill in the information at the bottom of the Fields and Field Sets editor for the *shared* Field Set, and click the [**Add Field Set**] button. The editor adds a tab for the *sharedimage* field set. When you are complete, Rhythmyx adds the *sharedimage* field set to the same object as the *shared* field set and displays the two shared field sets under the same object node in the Content Design view in the Rhythmyx Workbench.

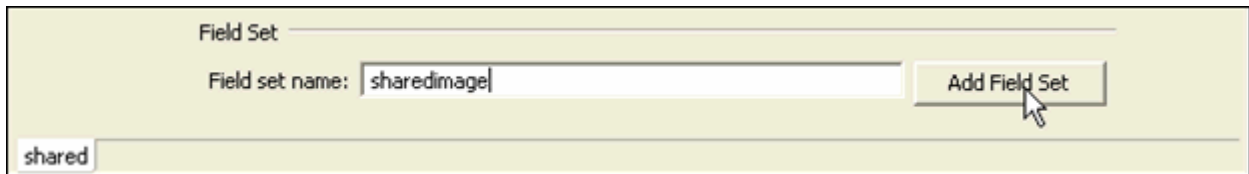
This topic focuses on the entry of the *img1* and *img1\_filename* fields to demonstrate when to use the *sys\_File* control, and the options **Treat Data as Binary**, **Show "clear field" checkbox in binary control**, and **Punctuation is part of word**, as well as others. After entering the *img1* and *img1\_filename* fields, you are instructed to enter the remainder of the fields in the *sharedimage* field set.

Note that we change the following default FastForward values from the *img1\_filename* field:

- We change the control from *sys\_EditBox* to *sys\_HiddenInput* to demonstrate use of the *sys\_HiddenInput* control. The *sys\_HiddenInput* control is usually only used if we never want users to see a field; if we may want them to see the field under some conditions, we apply a visibility rule, which is explained in the topic **Adding Field Visibility Rules** (see page 100).
- We uncheck **Show in Preview** to demonstrate why we would choose not to show a field in previews.

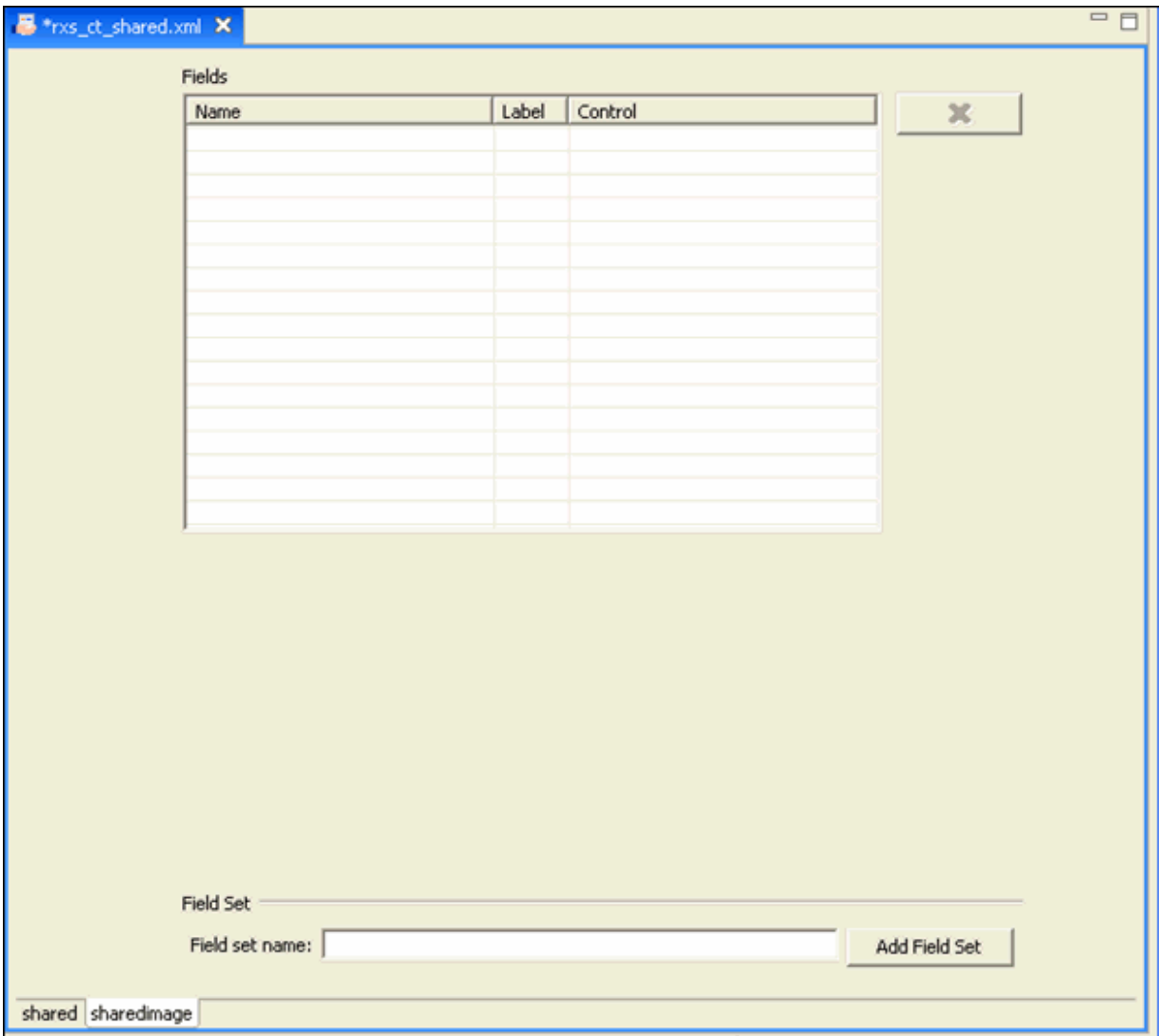
To enter the *sharedimage* field set:

- 1 After you enter the *shared* field set in the Fields and Field Sets Editor, under **Field Set**, enter *sharedimage* in **Field set name** and click **[Add Field Set]**.



*Figure 61: Adding another field set*

The editor adds a tab for the *sharedimage* field set and makes it the visible tab.



*Figure 62: sharedimage field set*

- 2 Now enter the fields specified in the topic *sharedimage Field Set specification* (see page 447).
- 3 Begin by entering the *img1* field.

- a) In the Fields table, under Name enter *img1*. *img1* is the internal name that Rhythmyx uses for the field. The editor automatically enters *Img1* under Label. Change this to *Image*.
- b) Under control choose *sys\_File*. The *sys\_File* control lets users enter or browse for a file that it uploads and stores. The control includes buttons for clearing the uploaded file or previewing it.

Figure 63: *sys\_File* control

Rhythmyx automatically adds the *sys\_fileInfo* extension as a dependency of Content Types with fields that use the *sys\_File* control. It functions as a pre-processing extension. The FastForward Image Content Type also includes the *sys\_ImageInfoExtractor* extension as a pre-processing extensions. A pre-processing extension is a java plugin that performs processing on a Content Item before a request is made to the database and before the Content Item is created. *sys\_FileInfo* and *sys\_ImageInfoExtractor* return various types of metadata that can be stored in other fields in the Content Item. In FastForward these metadata fields are included in the *sharedimage* field set.

In our example, we will add *sys\_ImageInfoExtractor* because it extracts file height and file width information as well as the metadata that *sys\_FileInfo* extracts, such as file name, MIME type, character length, and file encoding. Both extensions return the values to field names formed by combining the file name field (in this case, *img1* and *img2*) with specific suffixes. For example, *img1\_type* stores the MIME type. See ***sharedimage Field Set Specification*** (see page 447) for names of fields that these extensions use in the *sharedimage* field set.

Note that the Image Content Type must store extracted information into the *img1\_type* and *img1\_ext* fields in order to display the uploaded image in the browser, so you should always include these fields. You can choose to store other extracted values in the Content Type if they are useful for users to see or for processing. See *sys\_ImageInfoExtractor* and *sys\_fileInfo* in the *Rhythmyx Technical Reference* for more information and other metadata they can store.

- c) In **Data Type**, choose *binary*. All files are stored as binary data.  
The value in **Storage size** is automatically set to *max*.
- d) In **Storage size**, leave *max*. *max* represents the largest amount of space that the database used will allocate for binary data.
- e) Leave **Default Value** blank.
- f) By default, **Enable searching for this field** is checked. Uncheck this field because it is unlikely that the binary data that makes up the image will include any text that a user is searching for, and indexing the data and then searching uses unnecessary resources.
- g) Choose the **Mnemonic I** for the field.
- h) Click [**All Properties**] to set additional properties for the field.  
The Field Properties dialog opens.

- i) By default, **Treat data as binary** is unchecked. Check this field. Since image files may include large amounts of data, you should indicate that Rhythmyx should store the data *as binary*, which indicates that the database columns should include additional storage, and that they should store information in non-character format to save space.

The value in **Show 'clear field' checkbox in binary control** is automatically checked.

- j) Leave **Show 'clear field' checkbox in binary control** checked. It indicates that before the binary data is copied into the database the database column is cleared of all other data. This is important, because if a value is currently stored for the image in the database, and a user chooses to replace it with no data, unless Rhythmyx specifically knows to clear the original data, it may remain, and the field will not be empty as desired.
- k) All of the search fields are unchecked because you unchecked **Enable searching for this field** in the previous dialog. Leave the search fields unchecked.
- l) Click [**OK**] to return to the Field and Field Sets editor.

**4** Now enter the *img1\_filename* field.

- a) In the **Fields** table, under **Name** enter *img1\_filename*. The editor automatically enters *Img1\_filename* under **Label**. Change this to *Image file name*.
- b) Users do not enter the file name; the *sys\_file* control inserts it into the field. Although you want to save the filename for internal processing you do not want to display it to users because you do not want them to include it in content item output. Choose the *sys\_HiddenInput* control. This control stores the field invisibly in the Content Editor for processing or insertion in the database. By default it is a text field of 50 characters.
- c) In **Data Type**, leave *text*. All filenames should be stored as *text*.
- d) In **Storage size**, change the default value of 50 to 512. Although it is unlikely that anyone will enter a file name that uses this amount of space, it takes into account foreign characters that may require additional bytes and databases that may require extra space to store characters.
- e) Leave **Default value** blank.
- f) Leave **Enable searching for this field** checked, since advanced users may want to search on the filename of the image.
- g) Do not enter a **Mnemonic**. This field is filled in automatically when the file is uploaded and the user does not have to access the field. Note: Since the file is automatically uploaded and its properties are filled in by extensions, the only fields in the *sharedimage* field set that require mnemonics are *img1*, *img\_alt*, and *img2*, because users manually enter these fields.
- h) Uncheck **Show in Preview** so that the field is not shown in previews of templates of Content Items of this type. The field is used for internal processing, and is not included in Content Item outputs.
- i) Click [**All Properties**] to enter additional properties for the field.  
The Field Properties dialog opens.

- j) Punctuation is part of word is not checked by default. Since users might enter the entire file name into a search query, you want to include the punctuation that is included in filenames (the period between the filename and the extension) to be treated as part of the word. Check **Punctuation is part of word**.
- k) Click **[OK]** to return to the Fields and Field Sets editor.
- 5** Proceed to fill in the other fields in the *sharedimage* Field set. Use the data specified in the *sharedimage Field Set Specification* (see page 447) and apply any information explained in this topic. In this exercise, do not add visibility rules for the fields specified as hidden in the table in the *sharedimage* Field Set Specification; the next section will demonstrate how to do this in the topic *Adding a Field Visibility Rule* (see page 100). When you are done, the table should appear as:

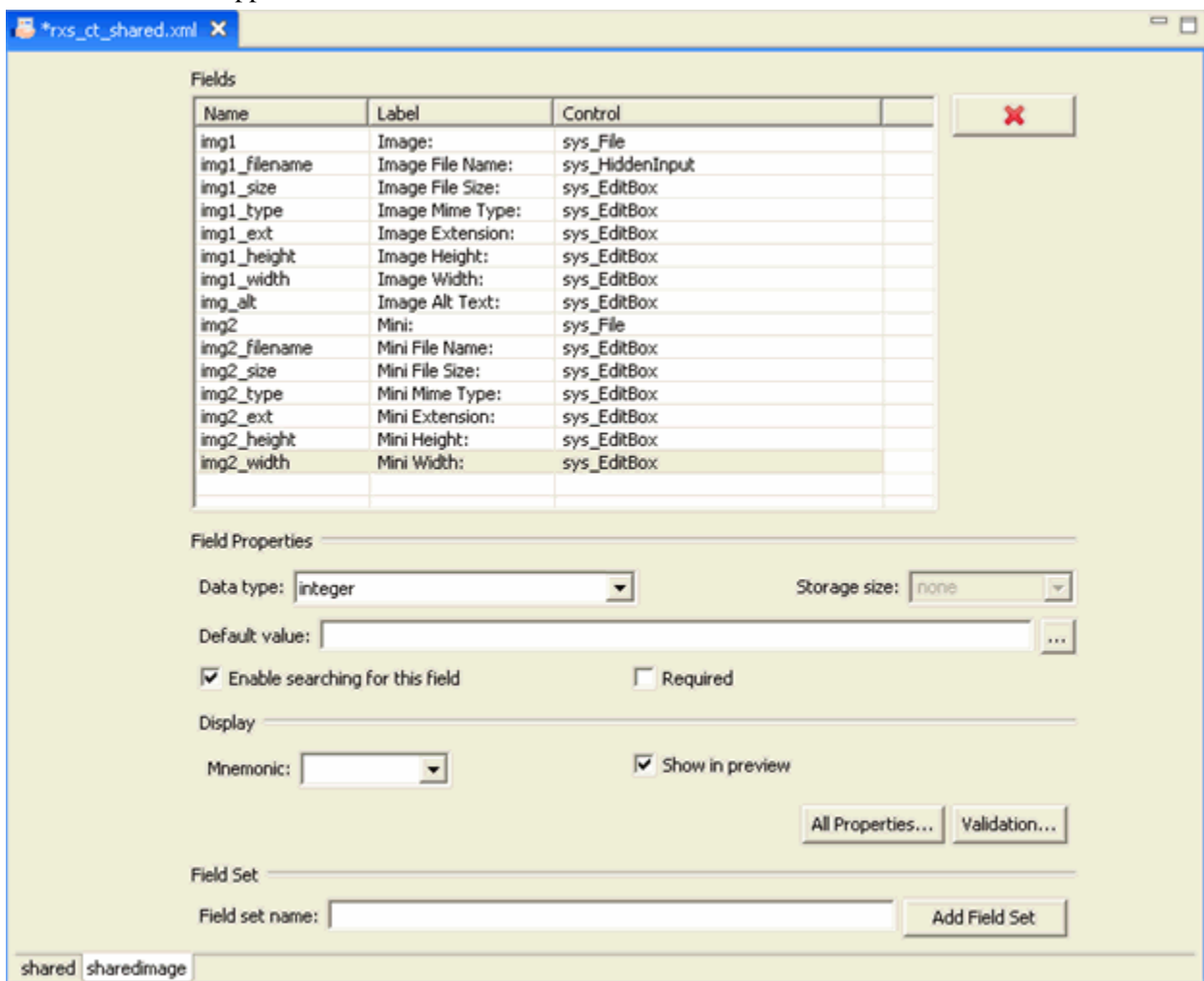

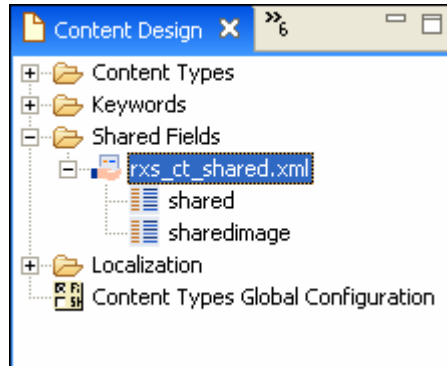


Figure 64: *sharedimage* field set entered in editor

- 6** Click X in the *rxs\_ct\_shared.xml* tab  to close the editor. A pop-up opens and prompts you to save your changes.
- 7** Click **[Yes]**.

The `rxs_ct_shared.xml` object appears in the Rhythmyx Workbench's Content Design View as:



*Figure 65: test\_shared.xml now appears in Content Design view*

---

You may have to refresh the Shared Fields folder to view the new object.

---

## Field Visibility, Validation, and Transform Rules

You can access editors for adding visibility, validation, read only, and transformation rules to fields at the bottom of the Field Properties dialog.

**Field Properties**

Field name:

Data type:

Storage size:

Default value:  ...

Treat data as binary

Display

Label:  Mnemonic:

Error label:

Control:  ...

Show in summary  Show in preview

Show 'clear field' checkbox in binary control

Search

Allow this field to be searched

Index formatting, such as HTML tags

Include in full text multi-field query

Punctuation is part of word (use for filenames, etc.)

Figure 66: Field Properties editor

The table below explains the functions of these rules and give some examples. The following topic, *Adding a Field Visibility Rule* (see page 100), shows you how to add one of these rules.

Type of rule	Function	Examples
read only	Specify under what conditions a field is read only in a Content Editor.	A read only rule could specify that a Workflow field is read only when the item is being modified because if the Workflow were changed the item could become inaccessible.
visibility	Specify under what conditions a field is visible in a Content Editor.	<p>A visibility rule could specify that a field that stores an uploaded file size is only visible to users in Administrator Communities since it is only used in internal processing.</p> <p>A visibility rule could specify that a file type field is visible to a user when the Content Item is being created but not when it is being edited.</p>
validation	Specify what values for a field are valid.	<p>A validation rule could specify that a value must be entered in a field.</p> <p>A validation rule could specify that a credit card number entered in a field must be located in an external database.</p>
transformation	<p>Input transformation rules specify how Rhythmyx should modify a value for storage in the repository when it is entered in a Content Editor field.</p> <p>NOTE: An input transformation rule should always be preceded by a validation rule to ensure that the data submitted is valid for the transformation.</p> <p>Output transformation rules specify how Rhythmyx should modify a value when it retrieves it from the repository to display in a Content Editor field.</p>	<p>An input transformation rule could specify that a date entered in the format Month dd, yyyy is stored in the format yyyyymmdd.</p> <p>An output transformation rule could specify that a two-digit state value stored in the repository is displayed as the full state name in the Content Editor.</p>

## Adding a Field Visibility Rule

Adding various types of rules to fields uses similar dialogs and similar procedures. To access one of the dialogs, click the corresponding button in the Field Properties dialog. You can also access the **[Validation]** button in the *Fields and Field Sets Editor* (see page 79).

Here, you will add a two-part visibility rule to the *img1\_size* field. You will add the first part of the rule by setting up a conditional statement, and you will add the second part of the rule by specifying an extension.



To add a visibility rule to the *img1\_size* field:

- 1 In Content Design view, in the Shared Fields folder, open your shared field object.
- 2 Access the tab for the *sharedimage* field set.
- 3 Click the row for *img1\_size* and click [**All Properties**].  
The Field Properties dialog opens.
- 4 Click [**Visibility**].  
The Field Visibility editor opens.
- 5 Since you are entering a new rule rather than editing an existing one, fill in the information at the bottom of the screen under **Rule Details**.
- 6 In the **Rule type** drop list, choose *Conditional*.

In the table that appears below the **Rule type** drop list, enter a rule that hides the *img1\_size* field for all Communities except Enterprise Investments Admin. The rule that you enter specifies when the field is visible, not when it is invisible.

Since the Enterprise Investments Admin Community ID is 1001 (confirm that this is true in your system), the rule you enter in the table is `sys_communityid = 1001`. Note that under value you can use the Value Selector to choose a method for entering a value. See *Selecting Values* in the *Rhythmyx Workbench Online Help* for information about using the Value Selector.

Rule Details

Rule type: Conditional

Variable	Op	Value
PSXSingleHtmlParameter/sys_communityid	=	1001

Add

Figure 67: Rule Details

- 7 Click [**Add**].

The rule appears in the Visibility table in the upper portion of the editor.

Visibility:	
Rule	Boolean
Conditional: <code>sys_communityid = 1001</code>	AND

Figure 68: Rule added in Field Visibility Rule dialog

- 8 If you were finished entering visibility rules, you would click **[OK]** to apply the rule. However, since you want to add a second statement to the rule select the next row.
- 9 Return to the **Rule Details** section of the editor and choose *Create Only* in the **Rule type** drop list.

This rule that makes the field visible when the Content Item is being created, but hides it when the Content Item is being edited.

- 10 Click **[Add]**.

The rule appears in the Visibility table below the first rule that you entered.

- 11 If *AND* is not already entered in the **Boolean** column beside the first rule, enter *AND*.

The entire visibility rule now states that the *img1\_size* field is visible if the Community is Enterprise Investments Admin and the Content Item is being created rather than edited.

Visibility:	
Rule	Boolean
Conditional: <code>sys_communityid = 1001</code>	AND
<i>Create Only</i>	AND

Figure 69: Completed Visibility Rule

- 12 Click **[OK]**.

The rule is associated with the shared *img1\_size* field and applies in any Content Editor that uses the field unless the rule is overridden within the Content Editor.

- 13 Click **[OK]** to close the Field Visibility editor and return to the *sharedimage* tab.

The most common field visibility rule is a simple conditional statement that is never true. This rule hides the field that it applies to. It is usually used for fields that you may choose to make visible in the future, because you can easily make the field visible by removing the rule. Now you will apply the rule to one of your *img1\_ext* field which does not use a `sys_HiddenInput` control, but that your specification lists as hidden. (Use `sys_HiddenInput` controls to hide fields that you do not intend to make visible in the future.)

To apply a simple conditional to the *img1\_ext* field:

- 1 Open the Field Visibility editor for the *img1\_ext* field.

- 2 In **Rule Type**, choose *Conditional*.
- 3 In the table that appears below the **Rule type** drop list enter  $1 = 2$ . You can enter *1* and *2* directly into the cells.
- 4 Click [**Add**].

The rule appears in the upper portion of the editor.

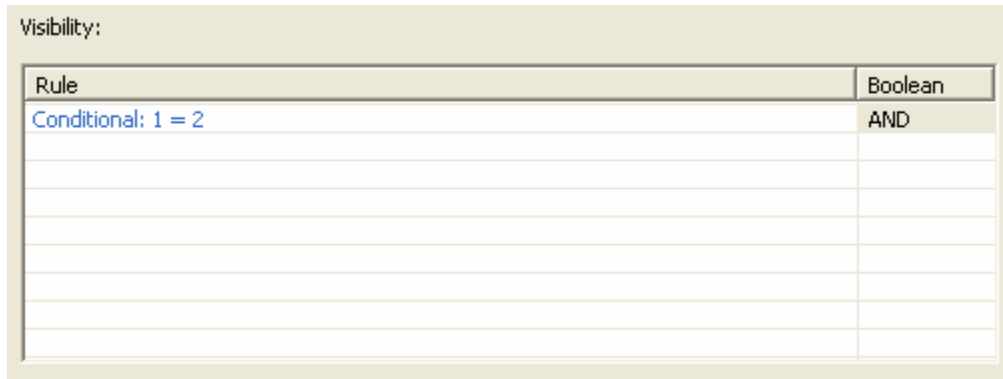


Figure 70: Visibility Rule for hiding a field

- 5 Click [**OK**].  
The Field Visibility editor closes.
- 6 Click [**OK**] in the Field Properties dialog.
- 7 Now, return to your shared and sharedimage shared field sets and add this simple visibility rule to all of the other fields that you want hidden by default. They are listed in the following table.

shared	sharedimage
filename	all of the fields beginning with img2_
webdavowner	

## Adding a Field Validation Rule

In this topic, we will add a common validation rule to the *img\_alt* field. The rule checks that the field, which is required, is entered. If not, it displays a message in the Content Editor stating that the field must be entered.

To add a validation rule to the *img\_alt* field:

- 1 In your shared field object's tab for the *sharedimage* field set, click the row for *img\_alt* and click [**Validation**].

The Field Validation dialog opens.

Field Validation

**Prerequisites** Validations run only if the prerequisites are satisfied.

Validation on insert or update:

Rule	Boolean

Rule Details

Validation type: Required

There are no parameters for this transform.

Add

Validation failure message:

OK Cancel

Figure 71: Field Validation Editor

- 2 No prerequisite conditions must be met before the validation occurs, so do not press the **[Prerequisites]** button.
- 3 In the drop list in the **Rule Details** box, the first validation listed is *Required*. Since this is the validation that you want to use, leave the drop list as it is.

- 4 In the Validation Failure Message text box, enter *The Image Alt Text field is required.*
- 5 Click [Add].

The Validation on insert or update table at the top of the dialog now displays the validation rule.

Field Validation

Prerequisites Validations run only if the prerequisites are satisfied.

Validation on insert or update:

Rule	Boolean
Required	AND

Rule Details

Validation type: Required

There are no parameters for this transform.

Apply

Validation failure message: The Image Alt Text field is required.

OK Cancel

Figure 72: Field Validation dialog

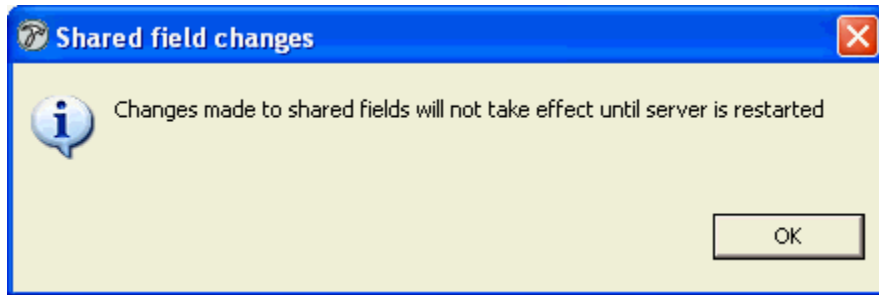
**6** Click **[OK]**.

The Field Validation dialog closes.

Now when a Content Item with the *img\_alt* field is entered or updated, the field validation rule will check if the field is entered. If not, it will display the message *The Image Alt Text field is required* in the Content Editor and prevent further processing until the field is filled in.

**7** Click **[X]** in the tab of the shared field editor, and click **[Yes]** when you are prompted to save your changes.

A dialog appears warning you that changes will not take effect until you restart the Rhythmyx server.



*Figure 73: Shared field changes dialog*

**8** Click **[OK]**.

The shared field editor closes.

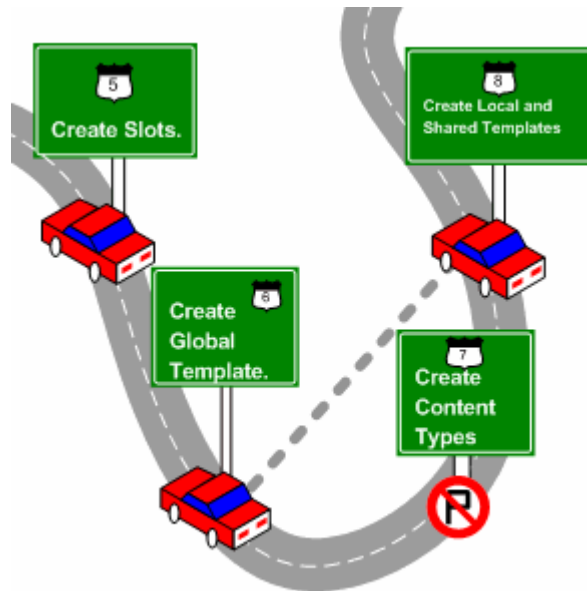
---

Note that we also could have demonstrated adding a field transformation rule, and the steps would have been similar to those for adding a field visibility or validation rule; the dialog lets you add standard transforms or extensions for setting up a rule. For more details about adding each type of rule, see the sections *Maintaining Field Validations*, *Maintaining Field Visibility Rules*, and *Maintaining Field Transforms* in the *Rhythmyx Workbench Online Help*.

---

## CHAPTER 6

# Creating Slots and Templates



After you complete development of your Shared Fields, the recommended implementation roadmap calls for the development of new Slots, and then Global Templates. The process of implementing all Templates, whether Global, shared, or local, is very similar, so this chapter includes a comprehensive discussion of the process of implementing Templates. When following the roadmap, however, local and shared Templates are typically implemented after *implementing Content Types* (see page 209); therefore, although the roadmap in the graphic above goes directly from creating the Global Template to creating local and shared Templates, it is only reflecting our procedure in this document.

During modeling and design, at least one Template is specified for each Content Type defined in the implementation. The Template specifies

- Which fields will be published; and
- The formatting that will be applied to the published field data.

For example, the `rffSnTitleCalloutLink` Template of the Generic Page Content Type includes the following fields:

- Title
- Callout

The Template formatting makes the text of the Title field bold, and adds a hypertext link to a full-page version of the Content Item. The Callout text is formatted as body text with rich formatting.

The screenshot shows the 'Content Properties' dialog box for a content item. The 'System Title' is 'E1 12 Easy Steps to preparing your estate plan'. The 'Title' field is '12 Easy Steps to preparing your estate plan' and is circled in red. The 'Start Date' is '2004-08-15 00:00:00'. The 'Expiration Date' and 'Reminder Date' are empty. The 'Keywords' field is empty. The 'Description' field contains 'Outlines the steps to preparing an estate'. The 'Callout' field is circled in black and contains a richly formatted paragraph with a background image of a New York Stock Exchange sign. The 'Body' field contains the text 'By Paul Baker' followed by a list of three items: '1. Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories w fond.', '2. If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health ca', and '3. Use [estate planning software](#) to make at least initial preparations. Most of the estate-planning software packag'. A callout box on the right shows the rendered output of the Title and Callout fields. The rendered Title is '12 Easy Steps to preparing your estate plan' in bold, and the Callout is a richly formatted paragraph with a background image of a New York Stock Exchange sign.

Figure 74: Generating an output using the `rffSnTitleCalloutLink` Template



The rffPgGeneric Template of the Generic Content Type, on the other hand, includes these fields:

- Title
- Body

The Template again formats the Title Field as bold text, this time with a serif font. The Body field is formatted as body text with rich formatting.

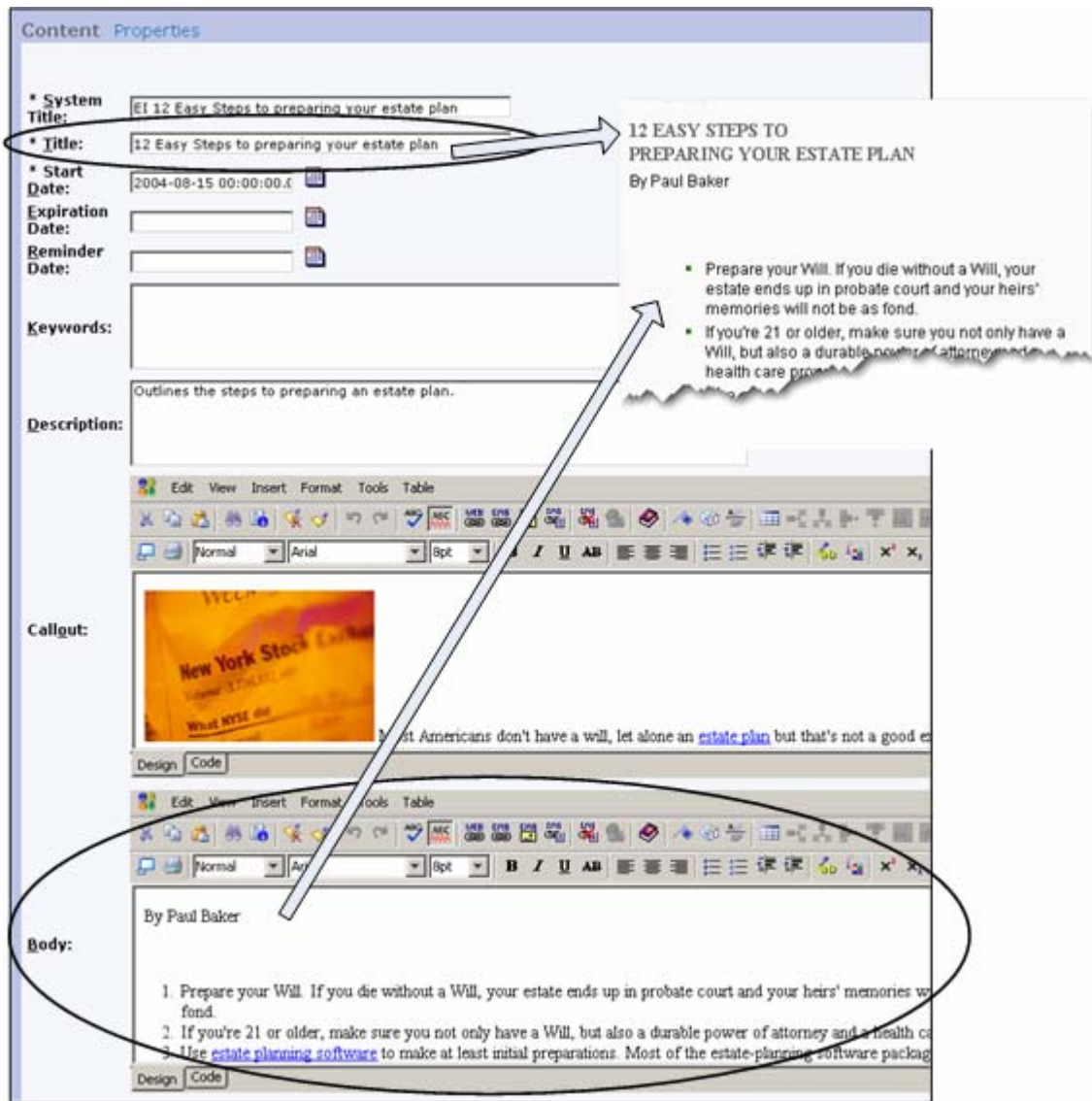


Figure 75: Generating an output using the rffPgGeneric Template

In addition to the data derived from the Content Item being formatted, a Template may include one or more Slots, or spaces where formatted data from other Content Items may be added. For example, the rffPgGeneric Template includes a List Slot, which allows users to add links to other Content Items that are related to the topic of the page:

## 12 EASY STEPS TO PREPARING YOUR ESTATE PLAN

By Paul Baker

- Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
- If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
- Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
- Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
- If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
- Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
- Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
- Choose a guardian for your children.
- Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
- Review your estate plan every few years, even if your situation is pretty much the same. Laws change constantly, and your planning may be out of date.
- Don't keep your insurance policies in your safe-deposit box. This delays filing for death benefits.
- There are three kinds of joint ownership. If you die, your share does not automatically go to the other owner. Make sure you have the right kind of joint ownership for your needs.

It's important to keep in mind that your estate-planning needs are probably much less complicated than they seem, even if they don't include all of the topics touched upon here.

<i>List Slot</i>
<p><b>Related...</b></p> <p><b>Before planning your estate, plan to see a lawyer</b></p> <p>Estate planning shouldn't be so complicated, but it is. Even if you're the do-it-yourself type, you should consult a lawyer to protect yourself and your heirs.</p> <p><b>Get your estate in order while you're capable</b></p> <p>When you're medically incapable of making decisions, a well thought-out estate plan lets you control your destiny and takes the burden off your family.</p> <p><b>Pick the right executor to handle your estate</b></p> <p>Consider this person the chief executive officer of your life -- after your life.</p>

Figure 76: Page Template showing the List Slot

Each of the Content Items in a Slot is formatted by a Template. In this case, the Content Items in the Slot are formatted using the `rffSnTitleCalloutLink` we saw illustrated above. Thus, Templates and Slots are recursive: a Template contains Slots, which themselves specify the Templates to use when formatting the Content Items in the Slot. The Templates in the Slot may themselves include Slots, and so forth.

The Workbench illustrates the relationships among these objects. For example, in Content View, you can list the Templates associated with each Content Type. For each Template, you can show the list of associated Slots. Note, however, that the View does not show the complete recursion. The Slots used in the Templates are not listed.

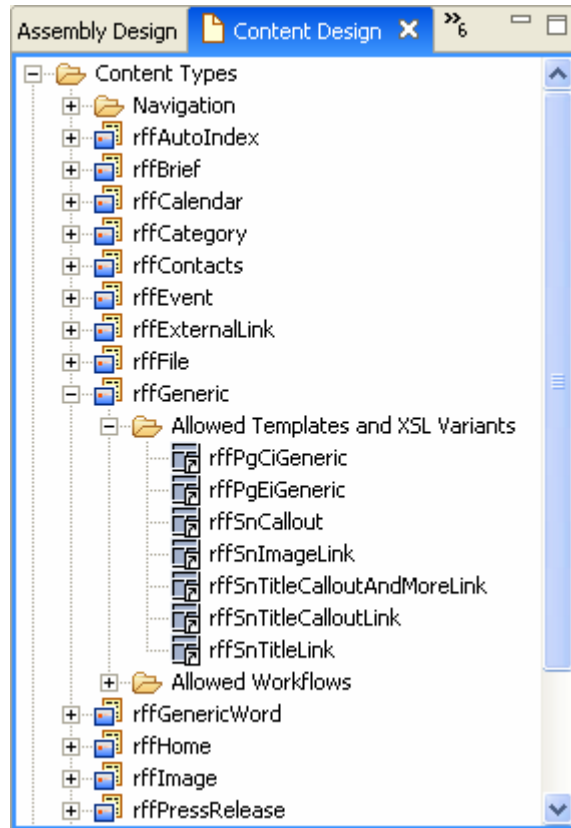


Figure 77: Content View showing Templates

In the Assembly View, you can list the Content Types associated with each Slot. For each Content Type, you can list the Templates that can be used to format Content Items for the Slot.

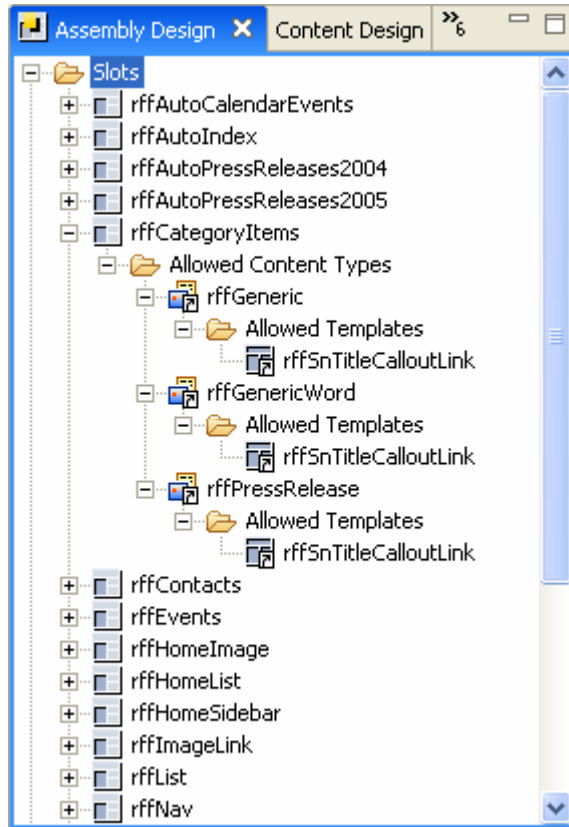


Figure 78: Assembly View Showing Slots, Content Types, and Templates

When viewing Templates, you can list the Slots contained by that Template, and the Content Types that can be formatted using that Template.

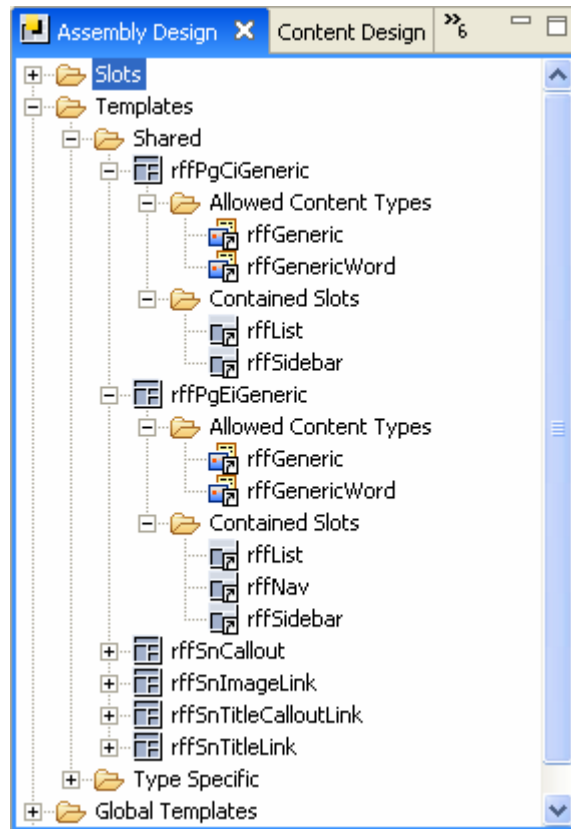


Figure 79: Assembly View showing Templates, associated Content Types and Slots

## Creating Slots

In Rhythmyx, Slots fall into one of two general categories: Regular and Inline. In addition to the standard Regular Slot, most implementations include two particular variations of Regular Slots:

- Automated Slots
- Managed Navigation Slots

Managed Navigation Slots are discussed in detail in the chapter *Managed Navigation* (on page 261).

Inline Slots require special implementation assistance from Percussion Software's Professional Services Organization. They are not discussed in this document.

A Slot definition includes two key pieces of information:

- the Content Finder

A Content Finder is a Rhythmyx extension that is used to populate a Slot with Related Content. Four Content Finders are available by default:

- `sys_RelationshipContentFinder`  
This is a standard Content Finder. It returns the Content Items added to the Slot by the user.
- `sys_AutoSlotContentFinder`  
This Content Finder is used to populate the Slot with an automatically-generated list of Content Items. For additional details see *Creating an Automated Slot* (on page 187).
- `sys_ManagedNavContentFinder`  
This Content Finder is used to populate the Managed Navigation Slot.
- `sys_LegacyAutoSlotContentFinder`  
This Content Finder is used in systems upgraded from Rhythmyx Version 5.7 or earlier. It uses the Automated Content Query Resources used to populate Automated Index Content Items in those Versions of Rhythmyx.
- `sys_TranslationContentFinder`  
This Content Finder is used to populate a Slot with links to Content Items associated in a Translation Relationship with the Content Item being assembled.

You can also implement additional Content Finders if none for the default Content Finders meet your needs. For details, see the *Rhythmyx Technical Reference Manual*.

- Allowed Content

The list of Allowed Content specifies which Content Types can be assigned to the Slot, and which Templates can be used to format Content Items of each Type in the Slot.

## Creating a Standard Slot

To illustrate the process of creating a Slot, we will implement the `rffImageLink` Slot used in FastForward. This Slot is relatively simple but still includes all of the important characteristics of a Regular Slot. The `rffImageLink` Slot has the following characteristics:

Slot Name	Description	Allowed Relationship Type	Content Finder
<code>rffSnImageLink</code>	General Slot for Images	Active Assembly	<code>sys_RelationshipContentFinder</code>

The following Content Types are allowed in this Slot:

Content Type	Template
Image	<code>rffSnImage</code>
Image	<code>rffSnImageandTitle</code>

---

*Note: You cannot create a Slot named `rffImageLink`, since it already exists in FastForward. Instead, create a similar Slot included in your implementation plan or copy our steps but give your Slot a different name.*

---

To create the Image Link Slot:

- 1 In the Rhythmyx Workbench, from the Menu bar, choose *File > New > Slot*.

The Rhythmyx Workbench displays the New Slot wizard.

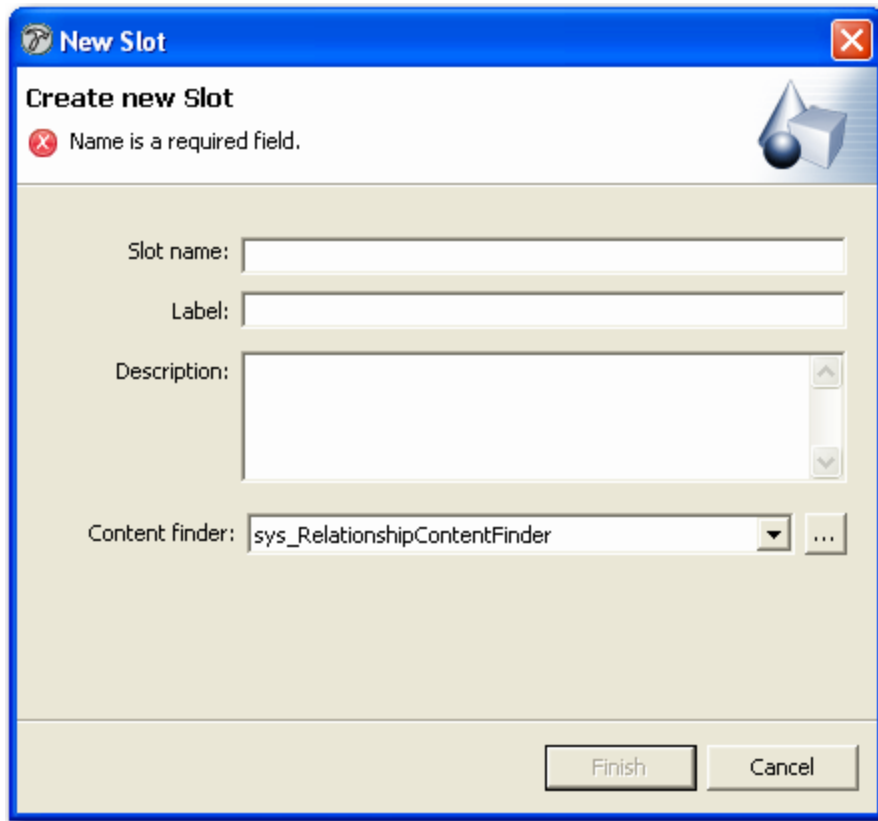
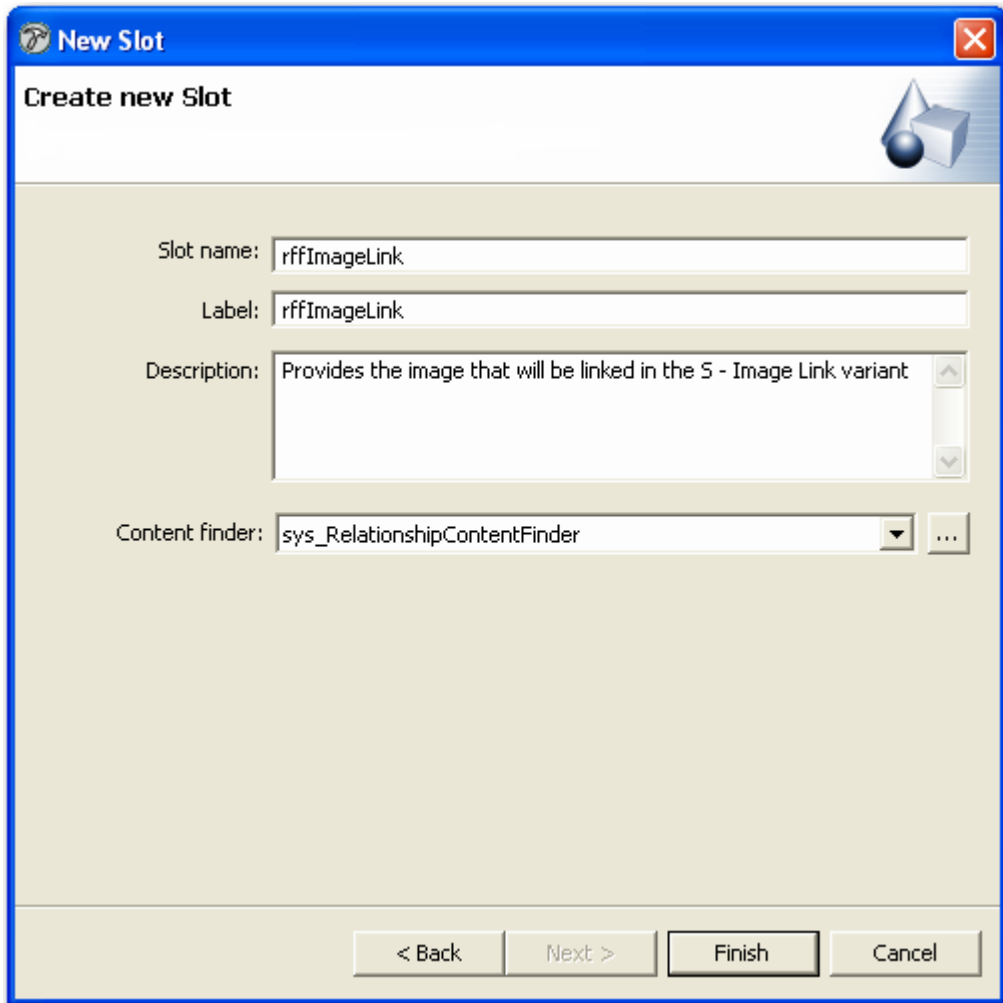


Figure 80: Slot Wizard

- 2 In the Slot name field, enter *rffImageLink*. This is the name defined for the Slot in the Implementation Plan. It is used for internal processing.
- 3 Note that the value *rffImageLink* is automatically entered in the Label field. The value in the Label field is displayed in Content Explorer, so we want a more user-friendly value. Change the value to *Image Link*.
- 4 In the Description field, enter *General Slot for Images*. This is the description for the Slot defined in the Implementation Plan.



- 5 In the Content finder drop list, choose `sys_RelationshipContentFinder`. This is the Content Finder we selected when developing the Implementation Plan. It is a generic Content Finder that retrieves content for the Slot. For additional information about Content Finders, see *Creating Sots* (see "Creating Slots" on page 114).



*Figure 81: Slot wizard with the rffImageLink Slot defined*

- 6 The New Slot wizard does not provide us with fields to add any more data, so click the **[Finish]** button.

Rhythmyx saves the Slot and displays it in the Slot editor.

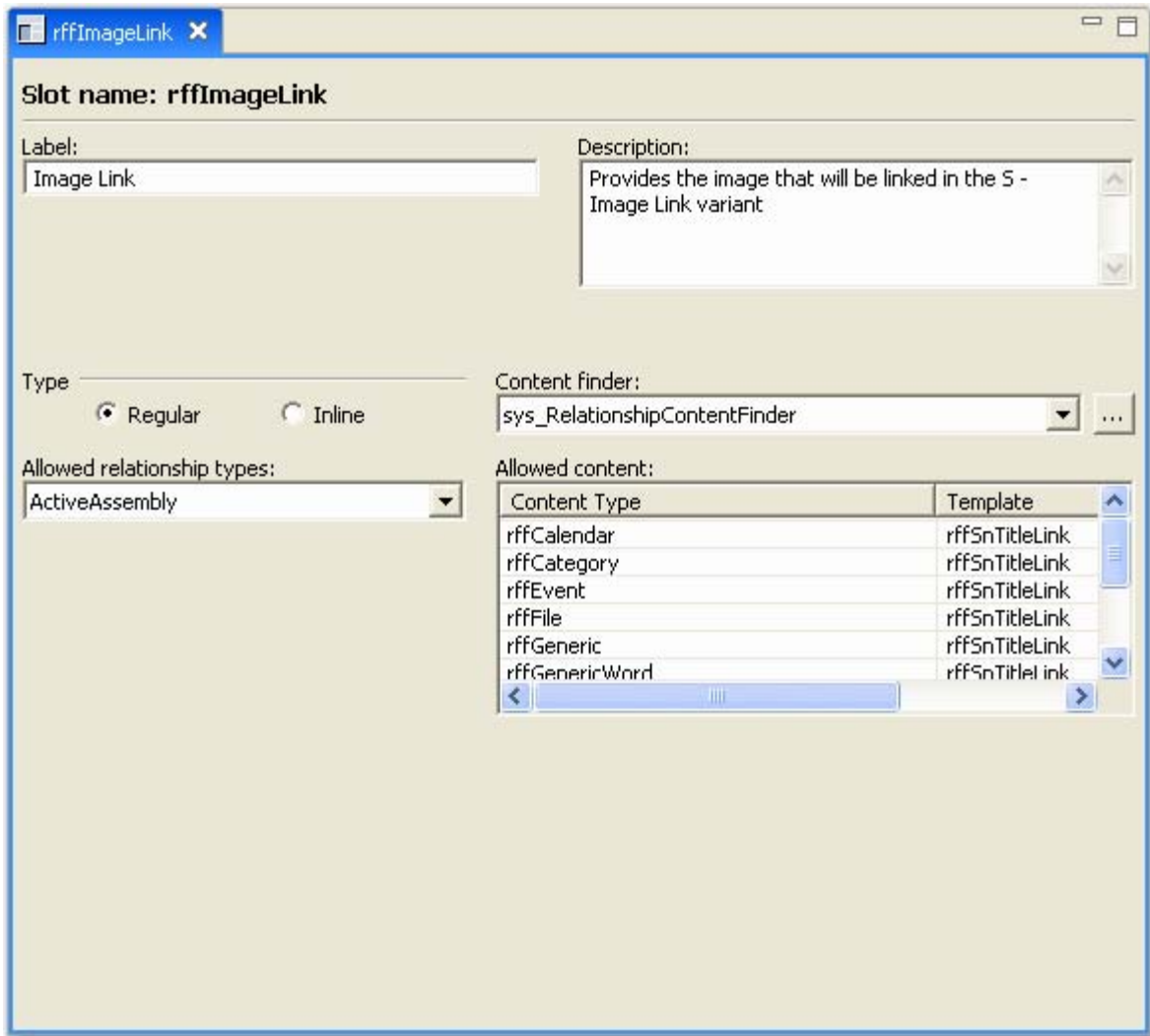


Figure 82: *rffImageLink* in Slot editor

- 7 We are implementing a Regular Slot, so leave the **Regular** radio button selected. Similarly, leave *ActiveAssembly* as the value in the **Allowed relationship types** field.

The options for the **Allowed relationship types** field include all Relationship Types in the Active Assembly Category. In the default installation of Rhythmyx, that Category includes the Active Assembly and Active Assembly – Mandatory Relationships. The Active Assembly – Mandatory Relationship includes processing that forces both Content Items in the Relationship to go Public together. That behavior is not required for this Slot.

- 8 To specify the Content Types and Templates for this Slot:
  - a) Click in the first row of the **Content Type** column and select *rffCalendar* from the drop list.
  - b) In the same row of the **Template** column, click and select *rffSnTitleLink* from the drop list.
  - c) Repeat Steps a and b, to add the remaining **Allowed content** to the Slot.

- 9 In the Button bar of the Rhythmyx Workbench, click the save button.

## Controlling the Contents of a Slot

In some cases, you may want more control over the contents of a Slot. For example:

- You might want to specify the Template used to format related Content Items in the Slot. In that case, specify a value in the `template` parameter for the `sys_RelationshipContentFinder`. The Template you specify for the Slot will override the Template specified by the Active Assembly Relationship associating the related Content Item to the Slot.
- You might want to control the maximum number of related Content Items in the Slot. In that case, specify the maximum number of related Content Items in the `max_results` parameter. Only the number of results specified will be added to the Slot. Usually, this parameter is used with the `order_by` parameter to specify how the related Content Items will be ordered in the Slot. If the `order_by` parameter is not specified, the results will be order as defined in the Slot.
- You might want to control the order of the Content Items in the Slot. In that case, use the `order_by` parameter. Specify the field you want to use to order the related Content Items in the field, and whether the order should ascending (`asc`) or descending (`dsc`). For example, in the `rffCategoryItems` Slot, the `order_by` parameter is specified as `rx:displaytitle asc`, which orders the related Content Items in ascending order alphabetically by the value in the Display Title field. If you wanted to see the most recently created Content Items, you would define the parameter as `rx:createdate dsc`. Use the `max_results` parameter to control the number of results included in the published output.

# Creating Templates

A Template is a Rhythmyx object that defines the assembly processing to generate an output. Three types of Templates are available:

- Local Templates specify the formatting specific to each Content Item.
- Global Templates provide a wrapper around the Local Template that controls the formatting and provides navigation for the pages. Global Templates provide a shortcut to applying consistent formatting when many different pages on the site share the same look and feel.



Figure 83: Global Template and Local Template

- Database Publishing Templates define the configuration to publish to a database. (NOTE: For additional information on Database Publishing Templates, see *Database Publishing in Rhythmyx* (on page 359).

When defining a Template, you must specify the plugin used to assemble the content in the Slot. The following plugins are shipped with Rhythmyx:

- Velocity  
Velocity Templates are used to format text outputs using the Velocity templating technology. Velocity Templates specify the fields that will be included in the published output, and define the formatting for the output. For additional detail about the Velocity technology, see <http://jakarta.apache.org/velocity> (see jakarta.apache.org/velocity - http://), or one of the following books:
  - Joseph D. Gradecki and Jim Cole, *Mastering Apache Velocity*
  - Rob Harrop, *Pro Jakarta Velocity*
- Binary  
Binary Templates are used to extract binary data, such as image files, from the Repository.
- Dispatch  
Dispatch Templates are used to calculate which Template will be used to format an output if multiple Templates are available.

Both Binary Templates and Dispatch Templates require data bindings. Velocity Templates can also use bindings. For additional information about bindings, see “*Bindings* (see page 136)”.

Local Templates fall into two categories, Page Templates and Snippet Templates. A Page Template outputs a fully formatted HTML page, while a Snippet Template outputs a portion of a page, which is rolled up with other Snippets in the Page output. Note that a Snippet Template may include Slots containing additional Snippets.

When creating a Snippet that uses fields defined locally for a specific Content Type, You should designate the Snippet as a Type-specific Template of that Content Type. This designation ensures that when you delete the specified Content Type, Rhythmyx will delete the Template automatically. This designation is not required by the Rhythmyx server, but it will save you additional manual cleanup when you delete the Content Type.

The recommended roadmap calls for the following implementation order:

- 1 Slots
- 2 Global Templates
- 3 Content Types
- 4 Local Templates

The general procedures for implementing both Global and Local Templates are essentially the same, however, differing only in specific details. We will begin by implementing Snippet Templates, which are the simplest form of Templates, moving on later to Page Templates and finally to Global Templates.

## Preparing HTML for Use in Templates

Before creating any Template, you should clean up the HTML to match best practice. Snippet Templates must be well-formed, although it is good practice to ensure that all of your templates are well-formed (and HTML used for splitting in legacy applications must be well-formed regardless of whether it is for a Snippet or a Page).

Local Template HTML should not include DOCTYPE declarations. Global Templates can include DOCTYPE declarations, but it is the responsibility of the implementer to ensure that the HTML meets the criteria for the DOCTYPE specified. If you specify a DOCTYPE of XHTML 1.0 Transitional, for example, you must ensure that the HTML formatting meets the requirements of XHTML 1.0 Transitional. Rhythmyx does not validate your markup to ensure that it matches the specified DOCTYPE.

It is also good practice to remove inline scripting HTML pages and store the scripts in support files, and remove any inline style markup, which should be stored in supporting CSS files. The supporting files should be added to a subdirectory of the web\_resources directory on your Rhythmyx server, as well as to the web resources directories on your production and staging servers.

## Implementing Snippet Templates

Snippets are used to include related Content Items on a page, either directly or contained within another Snippet. For example, the rffSnNameAndAddress Snippet allows business users to add contact information stored in a Contacts Content Item to an output page. This Template is a simple text-only Snippet that allows us to demonstrate the basic process of creating a Template. The Implementation Plan for this Snippet species the following data:

**Name:** rffSnNameAndAddress

**Label:** S - Name And Address

**Content Type:** Contact

**Assembler:** Velocity Assembler

**Output:** Snippet

**Publish:** Always

**Active Assembly Format:** Normal

**MIME Type:** <null>

**Character Set:** <null>

**Location Prefix:** <null>

**Location Suffix:** <null>

**Bindings:** None

**Communities:** Enterprise Investments, Corporate Investments

**Contained Slots:** None

**Sites:** Enterprise Investments, Corporate Investments

**Included Fields:** firstname, lastname, address1, city, state, zipcode, displaytitle

We will assume that the HTML for this Snippet is stored in an HTML file named rffSnNameAndAddress.html, which was created during the modeling and design process.

---

**NOTE:** A Snippet Template that is going to be used inline (to add Content Item text inline to a field on another Content Item), the <body> tag of the Snippet cannot have more than one child. If you intend to use a Snippet inline, the immediate child of the <body> tag must be a <div> tag.

---

## Creating a Text Template Object

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Template object for the rffSnNameAndAddress Snippet:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.

The Rhythmyx Workbench displays the Type dialog of the Template wizard.

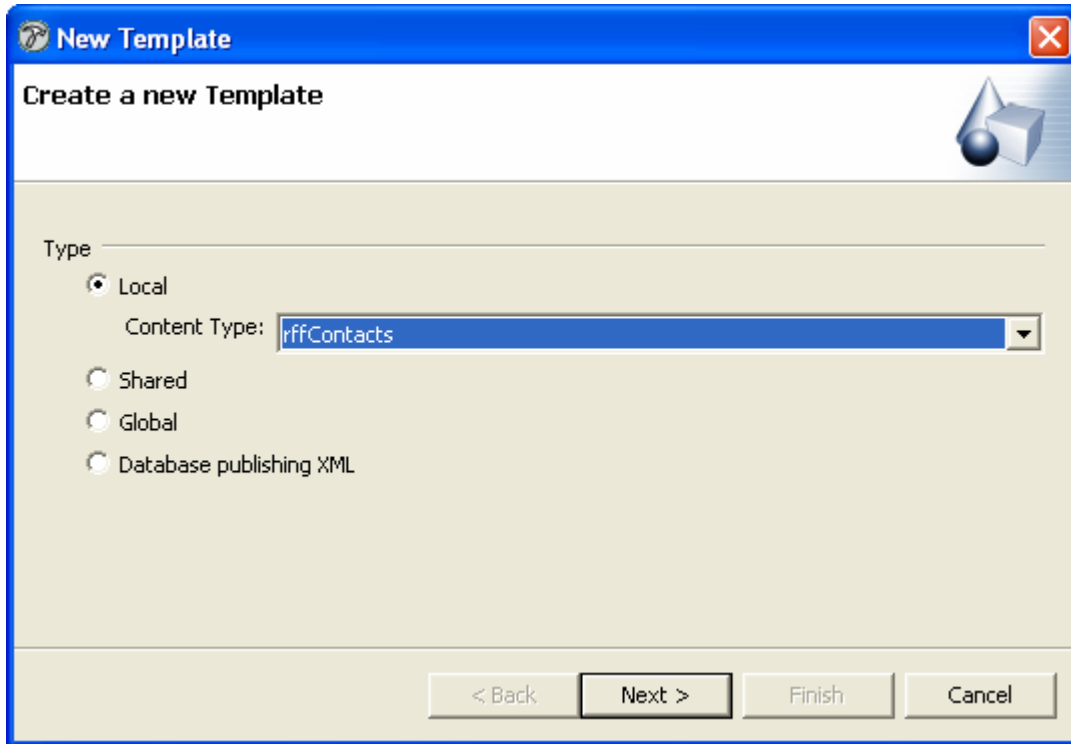
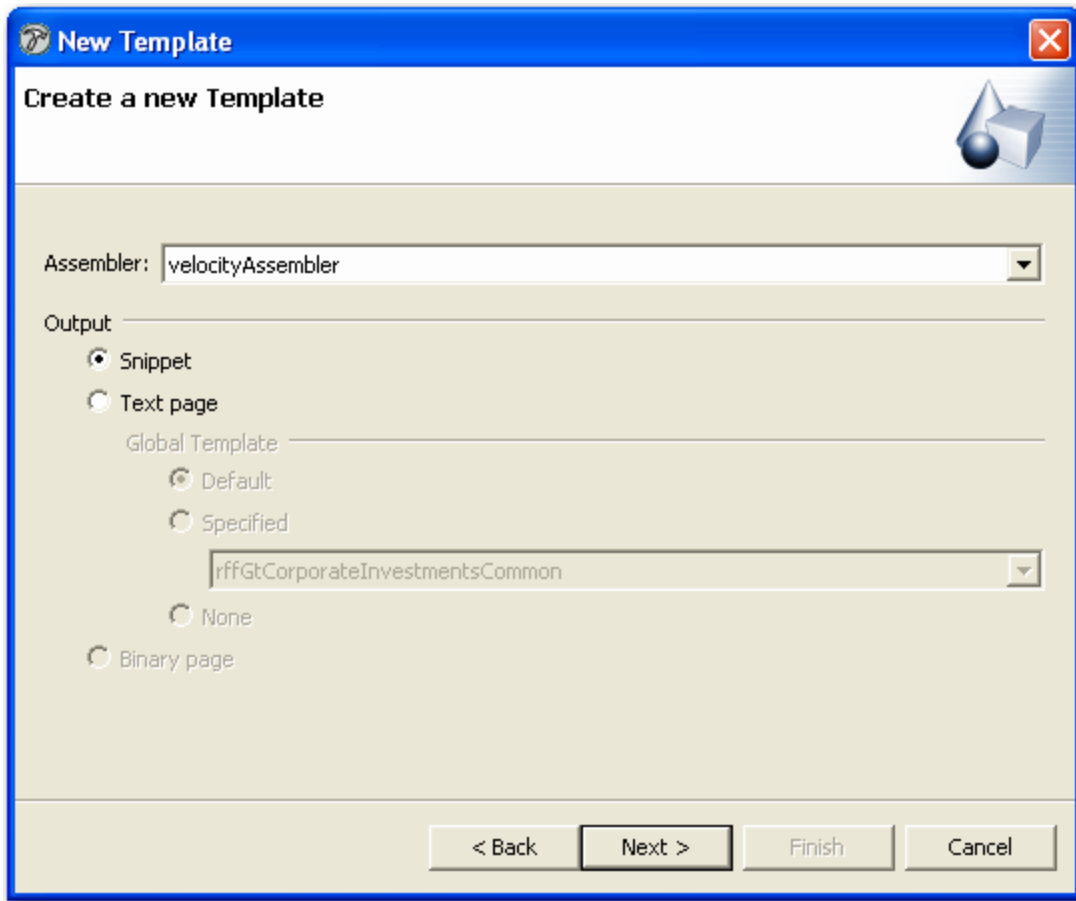


Figure 84: Type dialog for rffNameAndAddress Local Template

- 2 The rffSnNameAndAddress Snippet is specific to the Contacts Content Type, so make this a Type-specific Template. Choose the Type-specific radio button and click the [Next] button.

The Rhythmyx Workbench displays the Output format dialog of the Template wizard.



*Figure 85: New Template wizard Output dialog*

- 3** In the Assembler drop list, choose *Velocity Assembler* (this is the default option). In the Output section of the dialog, choose the **Snippet** radio button. Click the **[Next]** button.



The Rhythmyx Workbench displays the General properties dialog of the Template wizard.

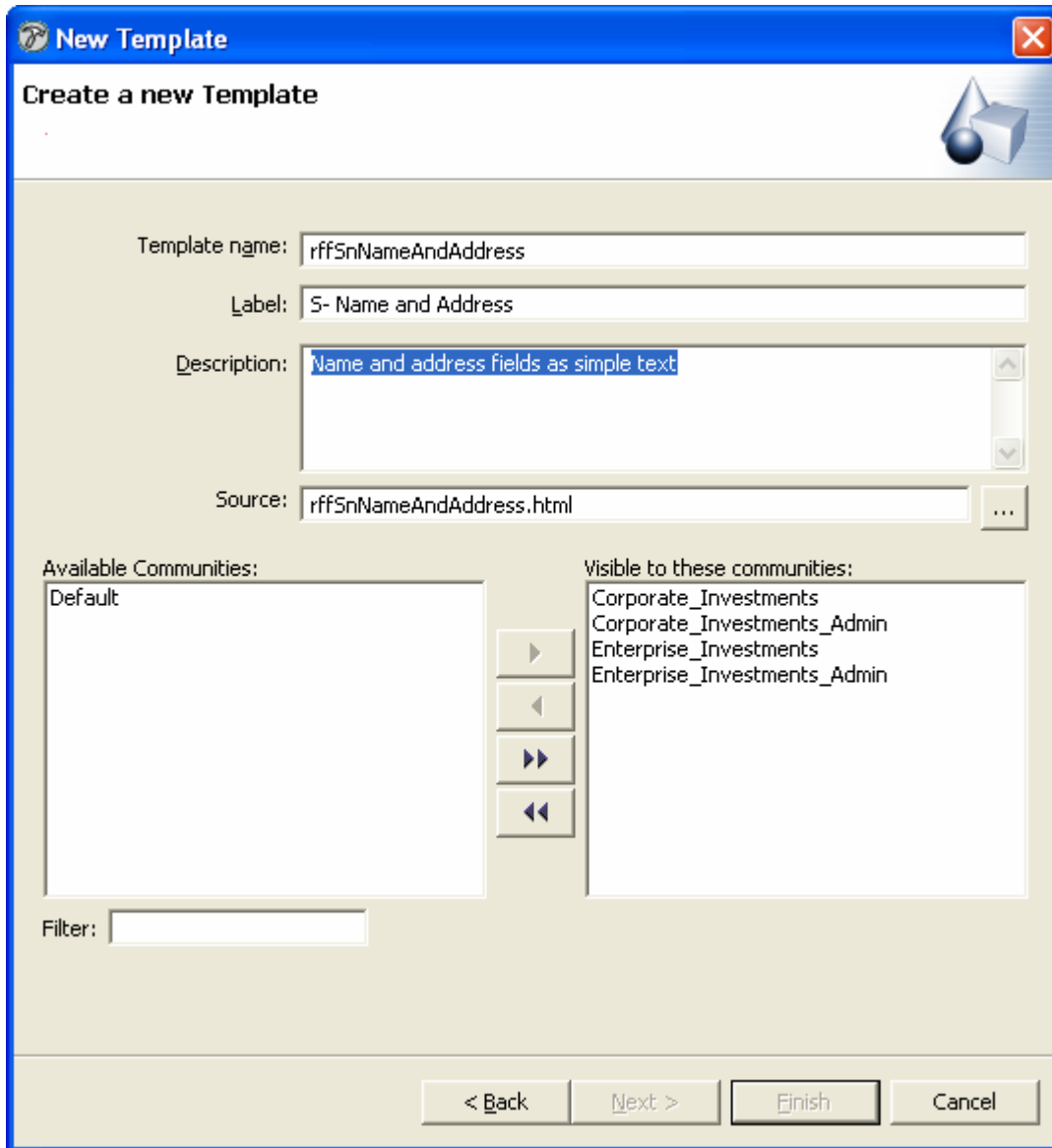


Figure 86: General dialog for *rffNameAndAddress* Local Template

- 4 In the Template name field, enter *rffSnNameAndAddress*. This value defaults to the Label field. In the Label field, change the value to *S - Name and Address*.
- 5 In the Description field, enter *Name and address fields as simple text*.
- 6 Click browse button next to the Source field, and use the browse dialog to find the file *rffSnNameAndAddress.html*, and add it to the field.
- 7 In the Available Communities field, select *Enterprise Investments* and *Corporate Investments* and click the [>] button to make this Template available to those Communities..
- 8 Click the [Next] button.

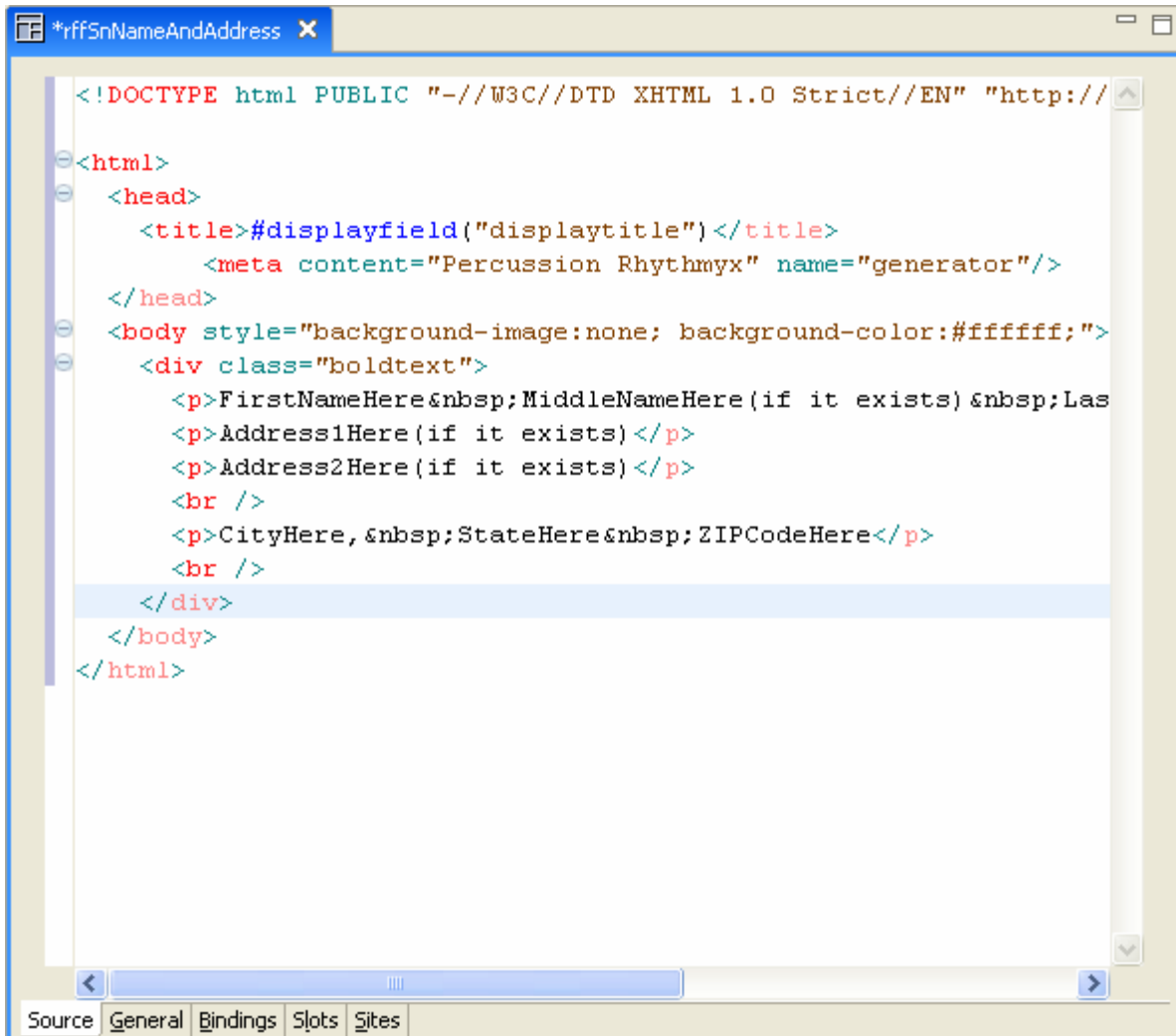
The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

- 9 This Template does not contain any Slots, so click the **[Finish]** button.

Rhythmyx creates the Template and displays the Template editor for the rffSnNameAndAddress Template.

## Adding Velocity Macros to a Text Snippet

For a text snippet, the source code is edited on the Velocity tab of the Template editor. The following screenshot shows the basic rffSnNameAndAddress HTML in the Velocity editor.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
<html>
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta content="Percussion Rhythmyx" name="generator"/>
  </head>
  <body style="background-image:none; background-color:#ffffff;">
    <div class="boldtext">
      <p>FirstNameHere MiddleNameHere(if it exists) Las
      <p>Address1Here(if it exists)</p>
      <p>Address2Here(if it exists)</p>
      <br />
      <p>CityHere, &nbsp;StateHere ZIPCodeHere</p>
      <br />
    </div>
  </body>
</html>
```

Figure 87: rffNameAndAddress Snippet source HTML in the Velocity editor

The Implementation Plan for this Snippet specifies that the dynamic content of this Snippet includes the following fields:

- displaytitle
- firstname
- middlename

- lastname
- address1
- city
- state
- zipcode

The `displaytitle` field provides the dynamic data for the `<title>` tag. The location for the remaining fields is specified in the source HTML illustrated above.

To include this data when assembling the Template, Rhythmyx provides a set of pre-defined Velocity macros. All of the macros shipped with Rhythmyx are available in the Snippet Drawer:

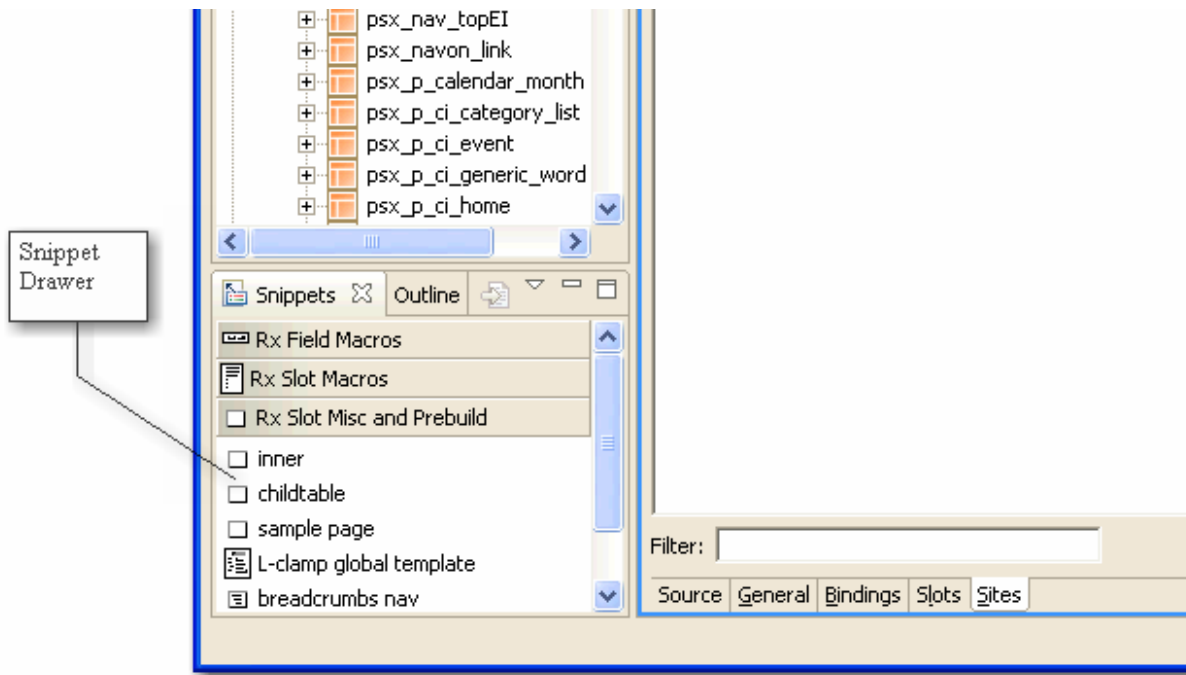


Figure 88: Snippet Drawer

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To add most fields, use the `#field` macro. For example, to add the `firstname` field

- 1 Select the Field Macros drawer.
- 2 In the Field Macros drawer, double-click `#field` macro.

The Rhythmyx Workbench displays the Insert Template: field dialog.

- 3 In the Variables table, click in the Value column next to the `fieldname` parameter and enter *firstname*.

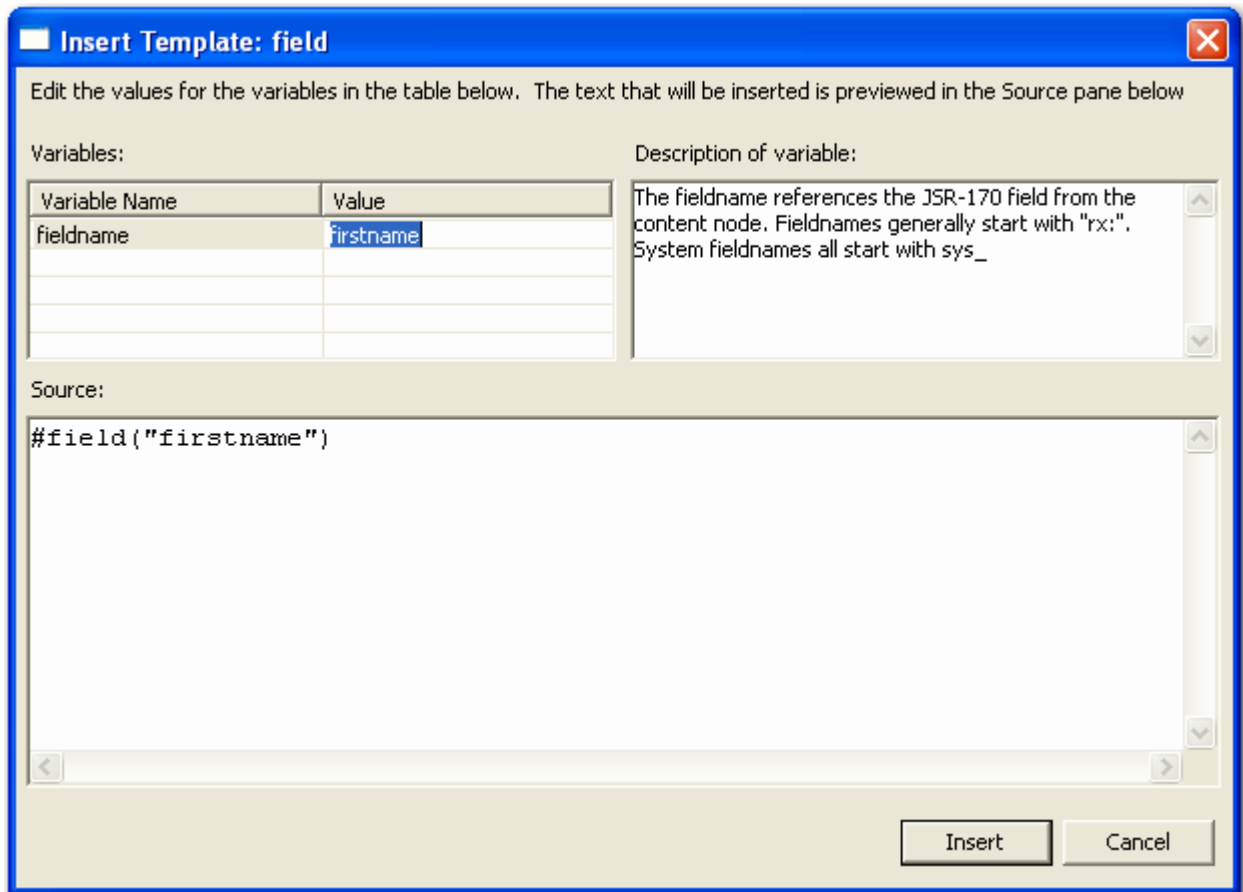
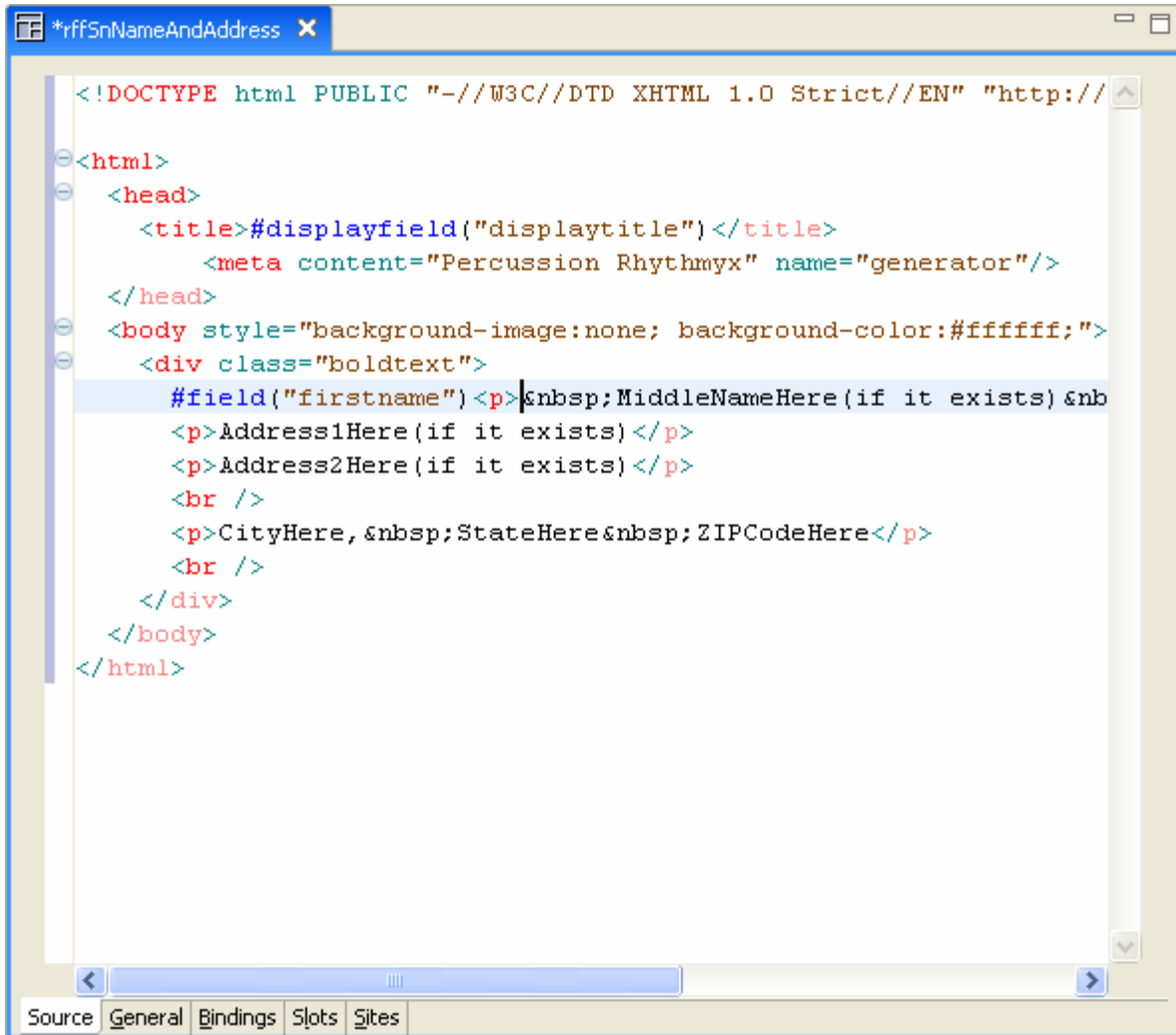


Figure 89: Insert Template dialog for the `#field` macro, with the value "firstname" for the `fieldname` parameter

The value of the `fieldname` parameter specifies the field to add to the Template output.

4 Click the **[Insert]** button to insert the field.



```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
<html>
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta content="Percussion Rhythmyx" name="generator"/>
  </head>
  <body style="background-image:none; background-color:#ffffff;">
    <div class="boldtext">
      #field("firstname")<p>&nbsp;MiddleNameHere(if it exists)&nb
      <p>Address1Here(if it exists)</p>
      <p>Address2Here(if it exists)</p>
      <br />
      <p>CityHere, &nbsp;StateHere&nbsp;ZIPCodeHere</p>
      <br />
    </div>
  </body>
</html>

```

Figure 90: *rffNameAndAddress* Template with *#field* macro

The `#field` macro should be used for most required fields. Many of the fields in a Content Editor are not required, however, and if the field does not have a value, the `#field` macro will return an error. For fields that are not required, use the `#field_if_set` macro. This macro includes a value in the HTML output only if the specified field in the Content Item has a value. If it does not have a value, this macro outputs no result.

For example, it is common for people to omit their middle names when entering contact information, so we would want to use the `#field_if_set` macro to ensure that this field was handled properly. This macro requires three parameters:

- before
  - This parameter defines text output before the contents of the field. Typically this text will be a non-breaking space or some punctuation.

- `field`  
This parameter defines the field to be displayed.
- `after`  
This parameter defines text output after the contents of the field. Typically this text will be a non-breaking space or some punctuation.

We want to put a non-breaking space before the middlename field to separate it from the firstname field. We could also add a non-breaking space after the field to separate the middlename field from the lastname field, but if the middlename field does not have a value, the lastname field will need to own that space, so we will include a null value for the `after` parameter; to specify a null value, use an empty set of quotation marks: `""`. As a result, our middlename field looks like the following code:

```
#field_if_set("&nbsp;", "middlename", "")
```

When added to the HTML, the result resembles the following:



Figure 91: `rffNameAndAddress` with the `#field_if_set` macro for the middlename field

Since the rest of the fields in the Content Type are optional, they all use the `#field_if_set` macro:

Field	Macro Markup
lastname	<code>#field_if_set("&amp;nbsp;" "lastname" "")</code>
address1	<code>#field_if_set("&lt;br /&gt;" "address1" "")</code>
address2	<code>#field_if_set("&lt;br /&gt;" "address2" "")</code>
city	<code>#field_if_set("" "city" ",")</code>
state	<code>#field_if_set("&amp;nbsp;" "state" "")</code>
zipcode	<code>#field_if_set("&amp;nbsp;" "zipcode" "")</code>

When all the markup is complete, it resembles the following screenshot:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
<html>
<head>
  <title>displaytitle</title>
  <meta content="Percussion Rhythmyx" name="generator"/>
</head>
<body style="background-image:none; background-color:#ffffff;
<div class="boldtext">
  #field("firstname")#field_if_set("&nbsp;" "middlename" "")
  #field_if_set("<br />" "address1" "")
  #field_if_set("<br />" "address2" "")
  <br />
  #field_if_set("" "city" ",")#field_if_set("&nbsp;" "state"
</div>
</body>
</html>

```

Figure 92: `rffNameAndAddress` HTML with all fields marked up

When using Active Assembly, the `#field` and `#field_if_set` macros include the Active Assembly icons that allow the user to edit and manipulate content. In some cases, manipulating content using Active Assembly is either not possible or not desirable. For example, content of the `<title>` tag in the HTML header of output is not eligible for Active Assembly.

For those cases, a different macro is available: `#displayfield`:

```
#displayfield(fieldname)
```

The difference between the `#field` macro and the `#displayfield` macro is that the latter does not include Active Assembly icons when using Active Assembly. We use this macro for the contents of the `<title>` tag:

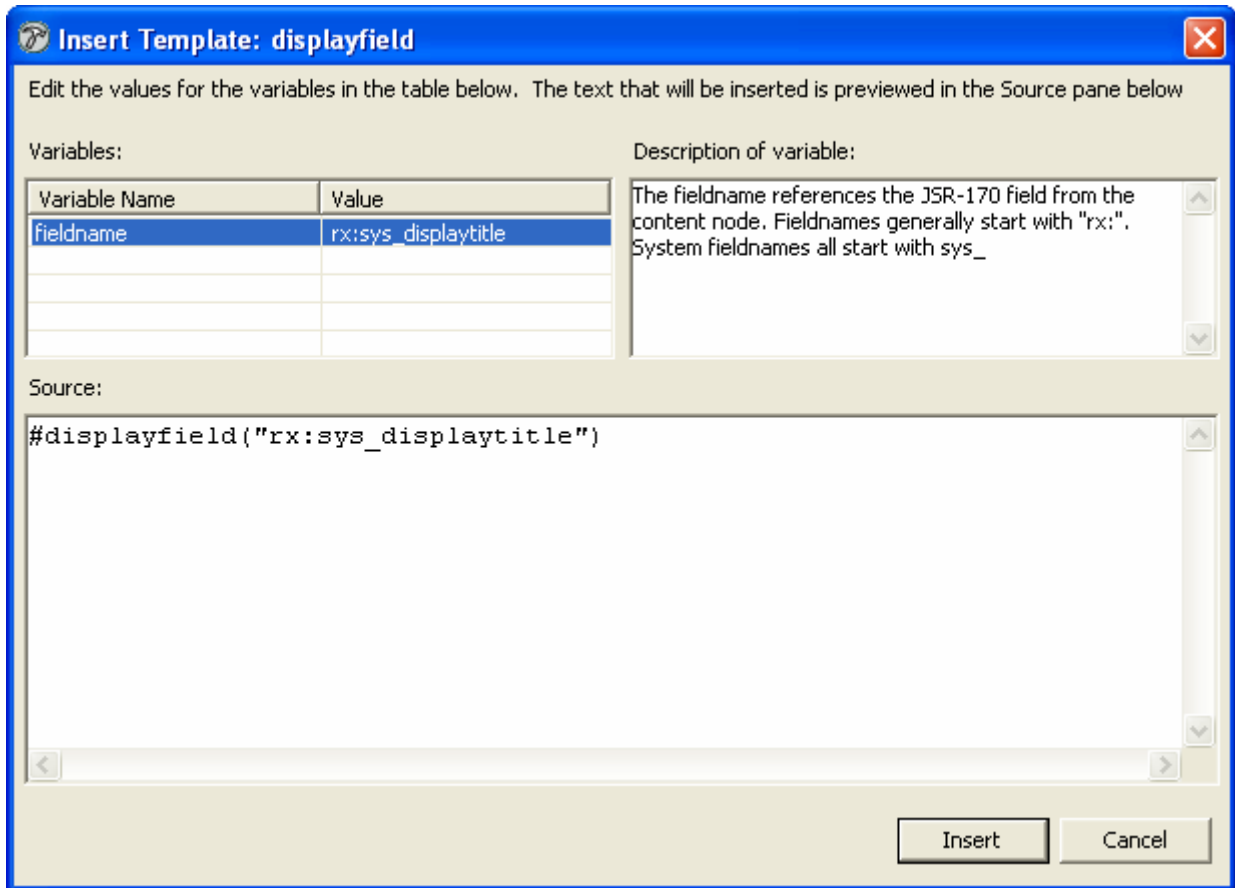


Figure 93: Insert Template dialog showing the `#displayfield` macro



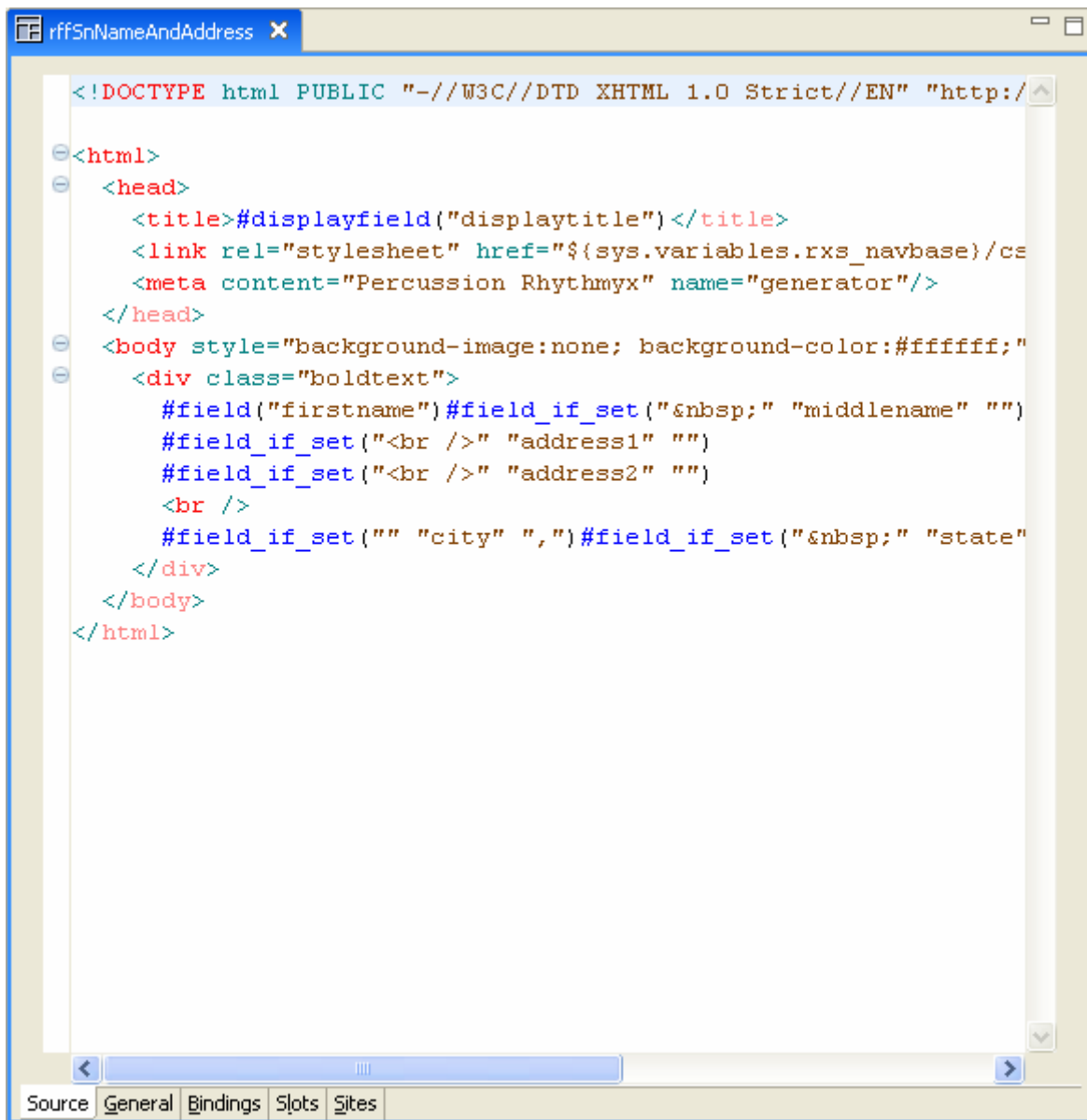


Figure 94: *rffNameAndAddress* Template with markup complete

NOTE: The `<link>` tag in the HTML `<head>` block defines a link to the CSS file used with the site. For additional details, see "*Converting References to Static Files*" (on page 174).

When previewing a Content Item using this Template, Rhythmyx produces the following result:

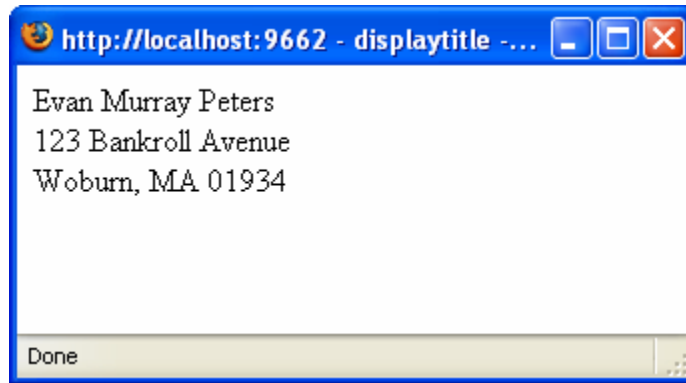


Figure 95: *rffSnNameAndAddress* preview

The final rendering of the Snippet depends on the Global Template applied to the page.

When adding a date field to a Template, use the `#datefield`, `#dispaydatefield`, and `#datefield_if_set` macros.

## Debugging Templates

A debugging output is available to help diagnose Templates that generate errors. To see the debug output, in the URL of the preview, change `/assembler/render` to `/assembler/debug`.

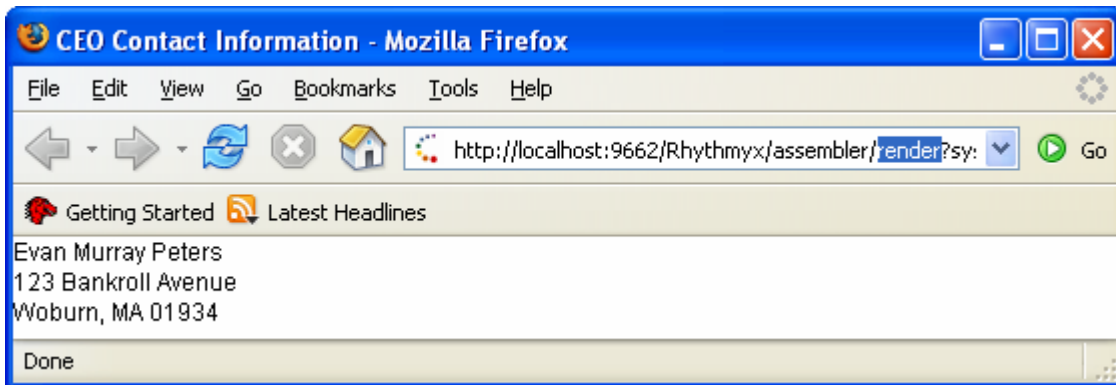


Figure 96: Preview of *rffSnNameAndAddress* Template with `render` selected in the URL.

When you change the *render* to *debug*, the browser displays all of the bindings, Content Item Nodes (including any Managed Navigation Nodes that would be included in a Page Template), and bound Slots.

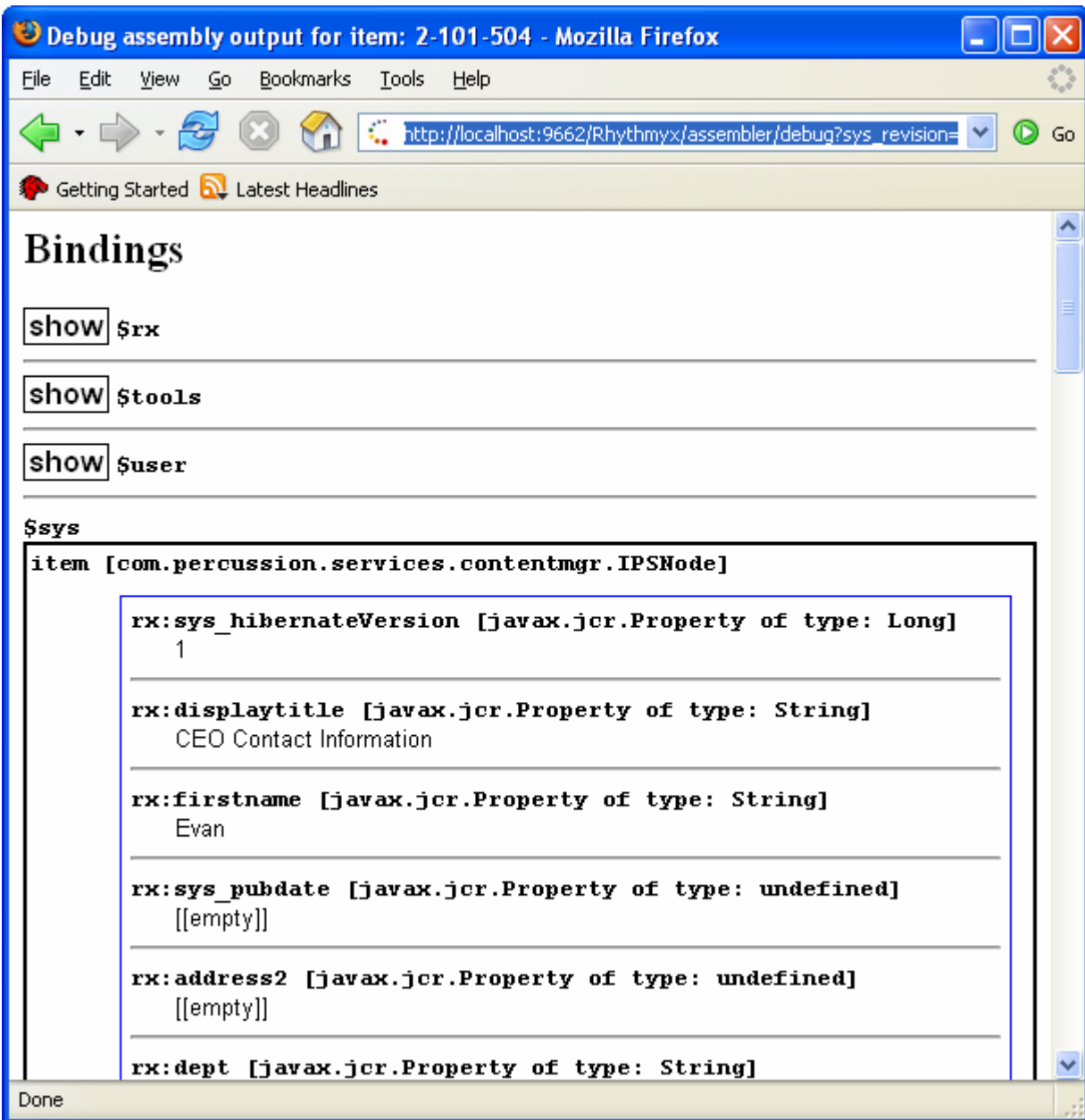


Figure 97: Debug output of the *rffSnNameAndAddress* Template previewed above

Use this view to check that the bindings, macros, and Slots are all defined correctly.

NOTE: When debugging Templates, you may see references to the Java class *PSAAUtils*. The methods of this class are used internally by Rhythmyx and are not publicly documented.

## Bindings

Bindings provide a mechanism for pairing data with a name that can be used in marking up HTML.

A binding consists of two parts:

- a variable name (binding variable).

Variable names must begin with the character “\$”, but rest of the name can use any alphanumeric string value; for example, `$name` is a valid variable. You can also define compound variables by separating the terms with dots. For example, you could define the variables `$circle.diameter` and `$circle.area`. This code specifies a variable `$circle`, which contains two additional variables, `diameter` and `area`. You can also define a variable as a list by specifying the index value for each element in the list. For example, a variable defined as `$name[ 0 ]` defines a variable `$name` which contains a list with a single entry. If you assign a second value, `$name[ 3 ]`, you have defined the variable `$name` as a list of four elements. The first and fourth elements of the list would have values, while the second and third elements in the list would be empty.

Rhythmyx includes a set of predefined binding variables. For details, see *Appendix I, Binding Variables* (see "Binding Variables" on page 409).

- a value definition

The value definition is an expression in Java Expression Language (JEXL) that defines the data for the variable. For details about JEXL, see <http://jakarta.apache.org/commons/jexl/>. (see [akarta.apache.org/commons/jexl/](http://jakarta.apache.org/commons/jexl/) - <http://>) The currently supported version of JEXL in Rhythmyx is JEXL 1.1. This version allows scripting and includes an if-else function. Neither of these capabilities were supported in JEXL 1.0, which was used in Rhythmyx Version 6.0 and 6.1.

Bindings are defined in a specific order. The order is important because a binding defined later in the order can use bindings defined earlier in the order as part of their calculations. For example, suppose we defined the following bindings:

```
$pi = 3.14159
$radius = $sys.item.getProperty("radius").number
```

(Note that binding variables are always prefixed by the "\$" character. Rhythmyx adds this character even if you define a binding without it. For example, if you define a binding as "variable", in Rhythmyx it will be returned as "\$variable".)

We can then define the calculation of the diameter in a new binding:

```
$circumference = $radius*$pi*2.0
```

In fact, we have already used bindings, as Rhythmyx Velocity macros are effectively a pre-defined set of bindings. For example, the definition of the `#displayfield` macro is:

```
#macro(displayfield
$fieldname)$sys.item.getProperty($fieldname).String#end
```

This code defines the macro “displayfield” which requires one argument, “fieldname”. In this case, the system uses a method of another internally-defined binding (the `getProperty` method of the `$sys.item` binding) to retrieve the value of the field specified. The value is returned as a string.

You must use bindings for Binary Templates and Dispatch Templates, which do not include Velocity markup. For a Binary Template, the bindings are used to define the source of the binary data in the Repository. For a Dispatch Template, the bindings are used to calculate the Template that will be used to produce the final output. Database Publishing Templates also use bindings, which determine the database location where the output will be published as well as the data to be published. Bindings are also used to generate the path for hypertext links.

Bindings can invoke any Java method, although a specific set binding variables and functions are provided for use in assembly and Location Scheme generation.

## Defining Bindings

Define bindings on the bindings tab of the Template in the Workbench.

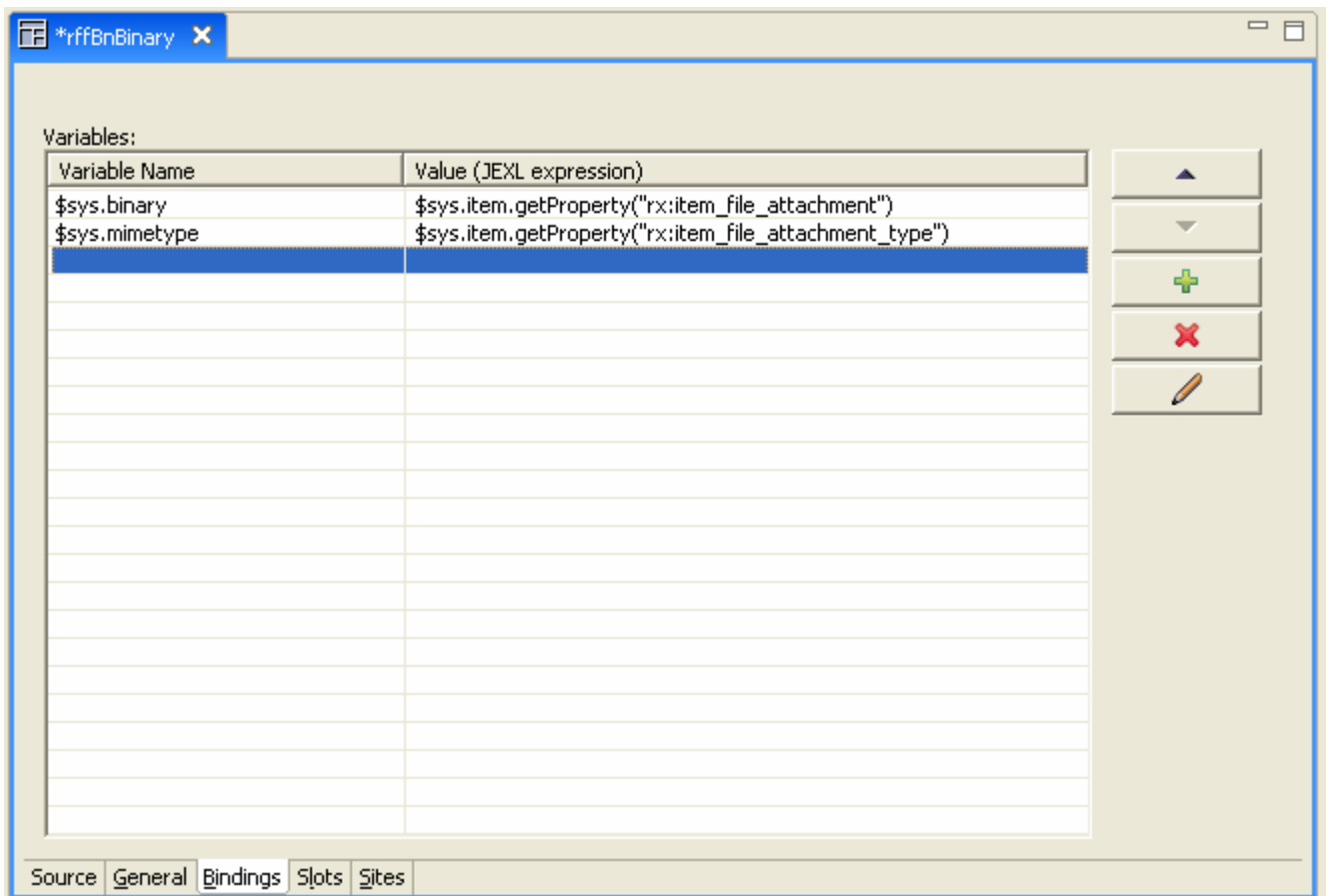


Figure 98: Template editor Bindings tab

To illustrate the process of defining a binding, we will create the variable `$fullname`, which consists of the values in the `firstname` and `lastname` fields in the current Content Item. We will include a non-breaking space to separate the two names.

In the `rffSnNameAndAddress` Snippet Template we defined earlier, we used the following code to define the first and last names in the Template:

```
<div><span>#field(firstname)</span>&nbsp;<span>#field(lastname)</span></div>
```

We can replace this code with the new variable, `$fullname`.

To define the `$fullname` binding:

- 1 On the Bindings tab of the Template editor, click in the first empty row of the **Variable Name** column and enter `$fullname`.
- 2 To retrieve data from a field, use the `getProperty` method of the `$sys.item` variable, specifying the field whose value to return; in this case, the value of the `firstname` field. The value should be a string, so specify the `.getString` method in the specification of the values.
  - a) Double-click in the Value (JEXL Expression) column of the same row, then click in the Expression Editor.
  - b) Enter `$sys.item.getProperty("rx:firstname").getString`.

Note that the Expression Editor has an autocomplete function. As you begin to enter text, the Rhythmyx Workbench displays a list of available binding variables and functions that match the text you entered. Thus, as you enter `$sys.i`, autocomplete displays `$sys.item` and `$sys.index`. You can select the binding variable or function that you want from the list presented.
- 3 After the `firstname` field, add the `lastname` field:  
`$sys.item.getProperty("rx:lastname").getString`.
- 4 JEXL requires an operator to combine, or concatenate, the two strings. The JEXL operator for concatenation is the plus sign (“+”). Insert a plus sign between the two values

- 5 As currently defined, the two strings will run together: *firstnamelastname*. To insert a space, add the string + '&nbsp;'; between the firstname variable and the plus sign:

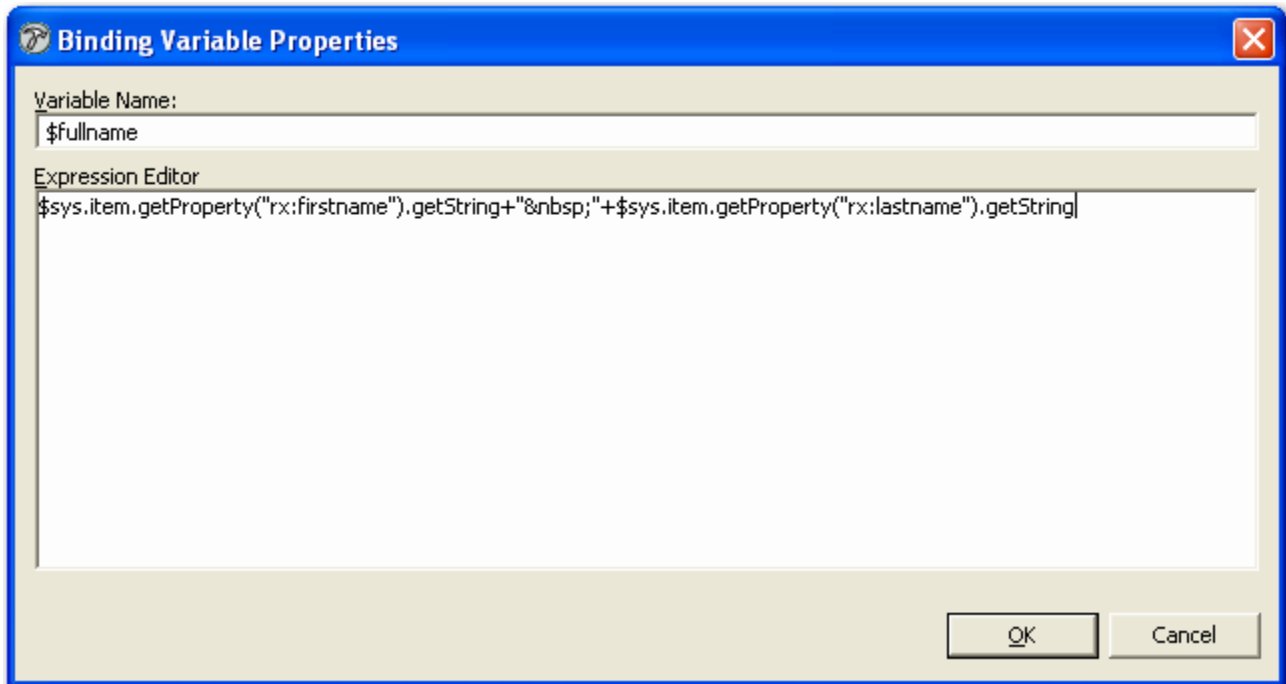


Figure 99: \$fullname Binding

To see how this works, in the code on the Velocity tab, replace the following line:

```
<div><span>#field(firstname)</span>&nbsp;<span>#field(lastname)</span></div>
```

with this code:

```
<div>$fullname</div>
```

When you preview, you will see the same result using the \$fullname binding variable as the original line of code produced. Note, however, that ActiveAssembly would not be available for these fields since they are provided by the binding.

You can copy and paste bindings between Templates. To copy and paste bindings:

- 1 Select the binding you want to copy.
- 2 Right-click and from the popup menu, choose *Copy*.
- 3 Open the Template to which you want to paste the binding and select the Bindings tab.
- 4 Right-click in an empty row and from the popup editor, choose *Paste*.

## Implementing a Binary Template

Binary Templates provide a simple practical example of a Template that uses bindings. A Binary Template retrieves binary files from the Repository for publishing. Binary Templates use the Binary Assembler rather than the Velocity Assembler. These Templates use a binding to specify the field from which to retrieve the data. Binary Templates must use the variables `$sys.binary` (mapped to the value `$sys.item.getProperty`, specifying the field where the binary data is stored) and `$sys.mimetype` (mapped to the value `$sys.item.getProperty`, specifying the field where the MIME type of the binary file is specified.) For example, the Image Content Type in FastForward uses the `img1` field to store the binary image file, so the binding would be:

```
$sys.binary=$sys.item.getProperty("img1")
$sys.mimetype=$sys.item.getProperty("img1_type")
```

When creating a binary file, you can define a default MIME type on the General tab of the Template editor. If the binary field stores only one MIME type, you do not need to do anything else. If the binary field stores more than one MIME type (for example, if it stores .gif, .jpg, and other image formats), you should define an additional binding for the `$sys.mimetype` variable to the Content Editor field that specifies the MIME type of the file.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To illustrate the creation of a binary Template, we will create the `rffBnImage` Template from FastForward. To create the `rffBnImage` Template:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.

The Rhythmyx Workbench displays the Type dialog of the Template wizard.

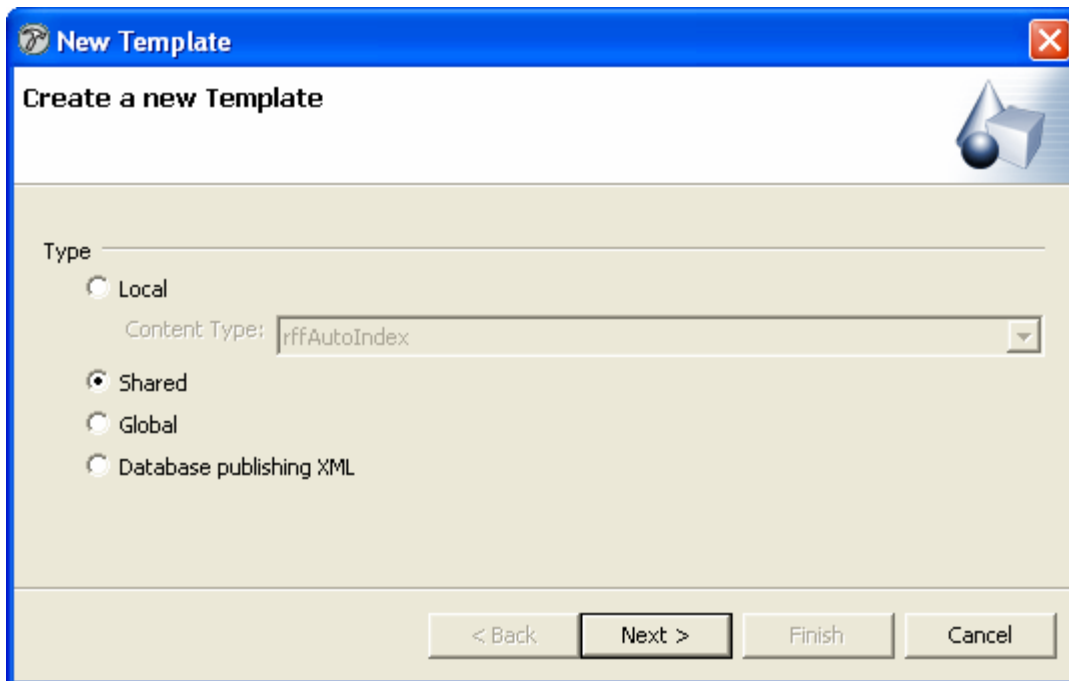


Figure 100: Creating a Binary Template as a Shared Template

- 2 Choose the **Shared** radio button and click the [Next] button.



The Rhythmyx Workbench displays the Output format dialog of the Template wizard.

- 3 In the Assembler drop list, choose *Binary Assembler* and click the [Next] button.

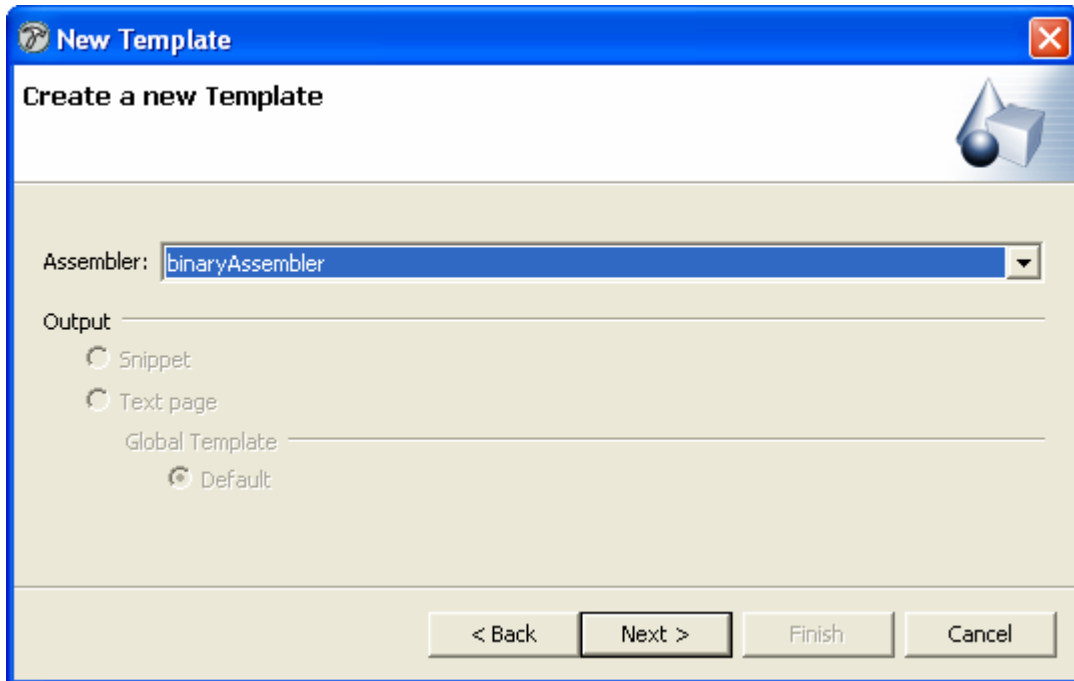


Figure 101: Choosing the binary Assembler for the Binary Template

The Rhythmyx Workbench displays the General Properties dialog of the Template wizard.

- 4 In the Template name field, enter *rffBnImage*. In the Label field, change the value to *Image*.
- 5 Add the Corporate Investments and Enterprise Investments Communities to the Visible in these Communities field.

Binary Templates do not use HTML markup, so ignore the Source field.

- 6 Click the [Next] button.
- 7 The Rhythmyx Workbench displays the Slots dialog. Binary Templates cannot contain Slots, so click the [Next] button.

The Rhythmyx Workbench displays the Content Types dialog.

- 8 In the Available Content Types field, select the *Image* and *NavImage* Content Types. Click the Add button (>) to move these Content Types to the Associated Content Types field.
- 9 Click the [Finish] button.

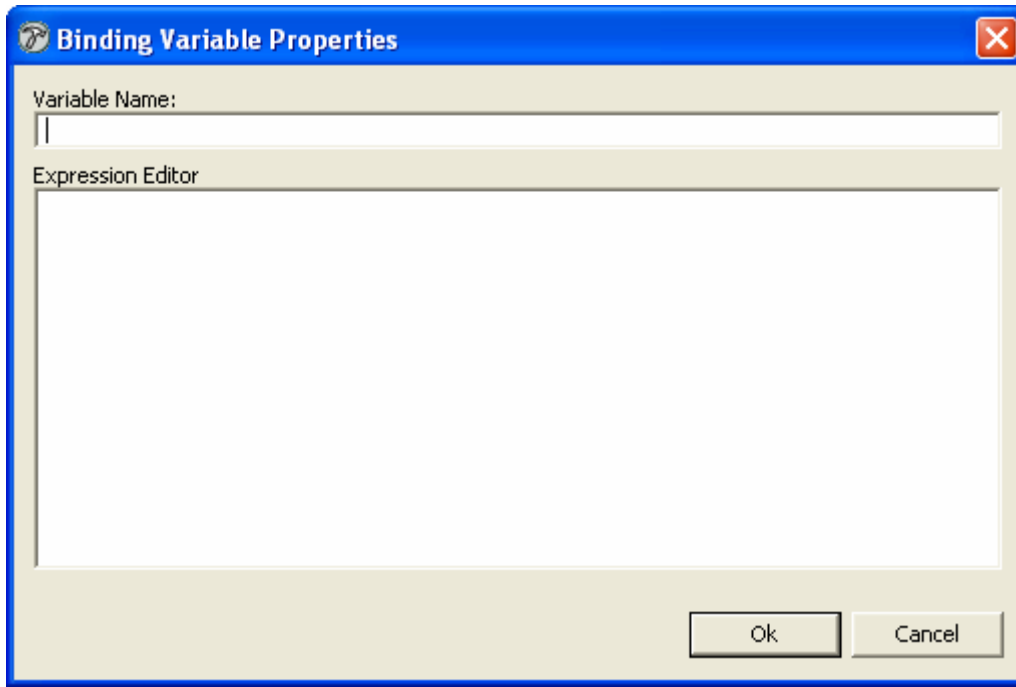
Rhythmyx displays the Template in the Template editor, with the Velocity tab selected.

- 10 Click on the Generate tab.
- 11 In the Mime type drop list, choose *image/gif*.
- 12 Click the Bindngs tab.

**13** To create the `$sys.binary` binding:

- a) Double-click in the first empty row of the **Variables** column

The Rhythmyx Workbench displays the Binding Variable Properties dialog.



*Figure 102: Binding Variable Properties dialog*

- b) In the **Variable Name** field, enter `$sys.binary`.
- c) In the Expression editor, enter `$sys.item.getProperty("img1")`.

- 14 To create the `$mimetype` binding, repeat step 14, entering `$sys.mimetype` in the Variables column and `$sys.item.getProperty(img1_type)` in the Value column.

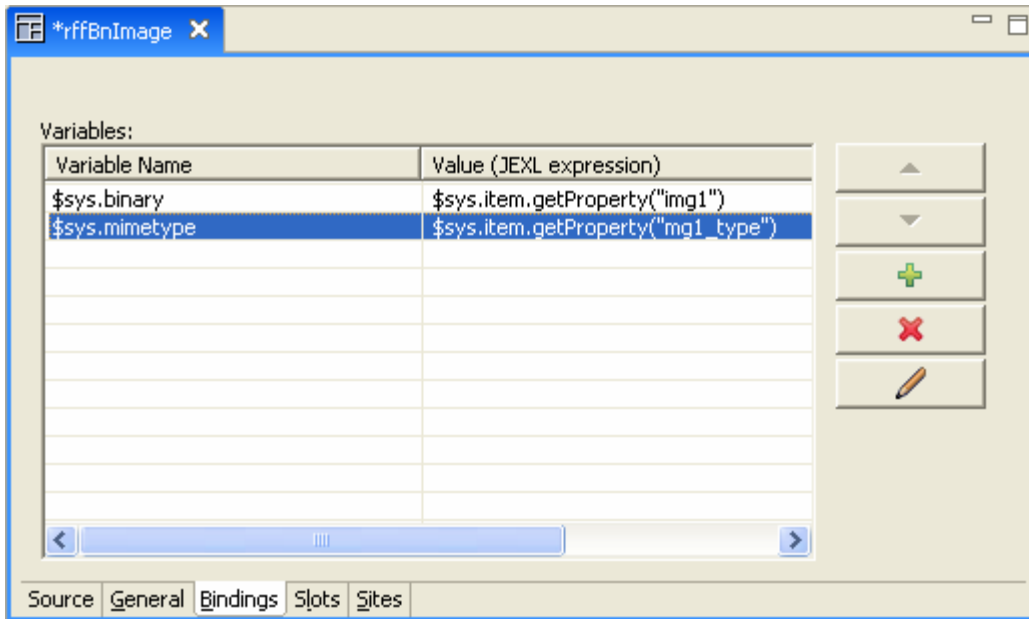


Figure 103: Image Template bindings

- 15 In the Button bar of the Rhythmyx Workbench, click the save button. To confirm that the Template works correctly, preview an image Content Item.

## Complex Snippets

To implement most Snippets, you can get by with the `#field` and `#displayfield` macros discussed earlier. In a few cases, however, Snippets require bindings to work correctly. Snippets that require bindings include:

- any Snippet that includes a hypertext link;
- any Snippet that includes a link to a binary file.

## Implementing Hypertext Links

To implement a hypertext link, create a binding to generate the location of the assembled Content Item using the `$sys.location.generate($sys.assemblyitem)` function. For example:

```
$pagelink=$sys.location.generate($sys.assemblyitem)
```

This code creates a link to the default Page Template of the Content Item. Typically, each Content Type has only one Page Template per site. If you implement more than one Page Template on a Site, add the template parameter:

```
$pagelink=$sys.location.generate($sys.assemblyitem,template)
```

This code links to the specified Template. Best Practice is to use a Dispatch Template that calculates the Page Template to use.

Whenever the value of a field is used as the text for a hypertext link, use the `#fieldLink` macro to add the field value if you want users to be able to follow the link in Active Assembly (users can follow a link by pressing the ALT key while clicking on the link). (If you use the standard `#field` macro in a hypertext link, users will not be able to follow the link.) Note that the `#fieldLink` macro requires two parameters: `fieldname` and `$pagelink`. The `fieldname` parameter specifies the field whose content will be included in the link. The `$pagelink` parameter is the `$pagelink` binding variable. This parameter must always be specified as `$pagelink`.

In Rhythmyx, hypertext links are most commonly based on the title of the Content Item, so we will use the `rffSnTitleLink` Snippet to illustrate how to create a hypertext link. The Hypertext link uses the following HTML:

```
<html>
  <head>
    <title>DisplayTitle</title>
  </head>
  <body>
    <div>
      <a href="path to Page Template of Content
Item">#fieldLink("displaytitle", $pagelink)<a/>
    </div>
  </body>
</html>
```

We will assume that this code is stored in a file named `rffSnTitleLink.html`. The `rffSnTitleLink` Template is associated with the following Content Types:

- `rffCalendar`
- `rffEvent`
- `rffFile`
- `rffGeneric`
- `rffGenericWord`
- `rffHome`
- `rffPressRelease`

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To implement the `rffSnTitleLink` Snippet:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.  
The Rhythmyx Workbench displays the Type dialog of the Template wizard.
- 2 Choose the **Shared** radio button and click the **[Next]** button.  
The Rhythmyx Workbench displays the Output format dialog of the Template wizard.
- 3 In the **Assembler** drop list, choose *Velocity Assembler* (this is the default option). In the Output section of the dialog, choose the **Snippet** radio button. Click the **[Next]** button.  
The Rhythmyx Workbench displays the General properties dialog of the Template wizard.
- 4 In the **Template name** field, enter *rffSnTitleLink*. Update the value in the **Label** field to *Title Link Snippet*..
- 5 In the **Description** field, enter *Renders the title as a hypertext link*..

- 6 Click browse button next to the Source field, and use the browse dialog to find the file `rffSnTitleLink.html` and add it to the field.
- 7 In the **Available Communities** field, select *Enterprise Investments* and *Corporate Investments* and click [>] button to make this Template available to those Communities..
- 8 Click the [**Next**] button.

The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

- 9 This Template does not contain any Slots, so click the [**Next**] button.

The Rhythmyx Workbench displays the Content Types dialog of the Template wizard.

- 10 In the **Available Content Types** field, select the following Content Types:

- `rffCalendar`
- `rffEvent`
- `rffFile`
- `rffGeneric`
- `rffGenericWord`
- `rffHome`
- `rffPressRelease`

Click the  button to move these Content Types to the **Associated Content Types** field.

- 11 Click the [**Finish**] button.

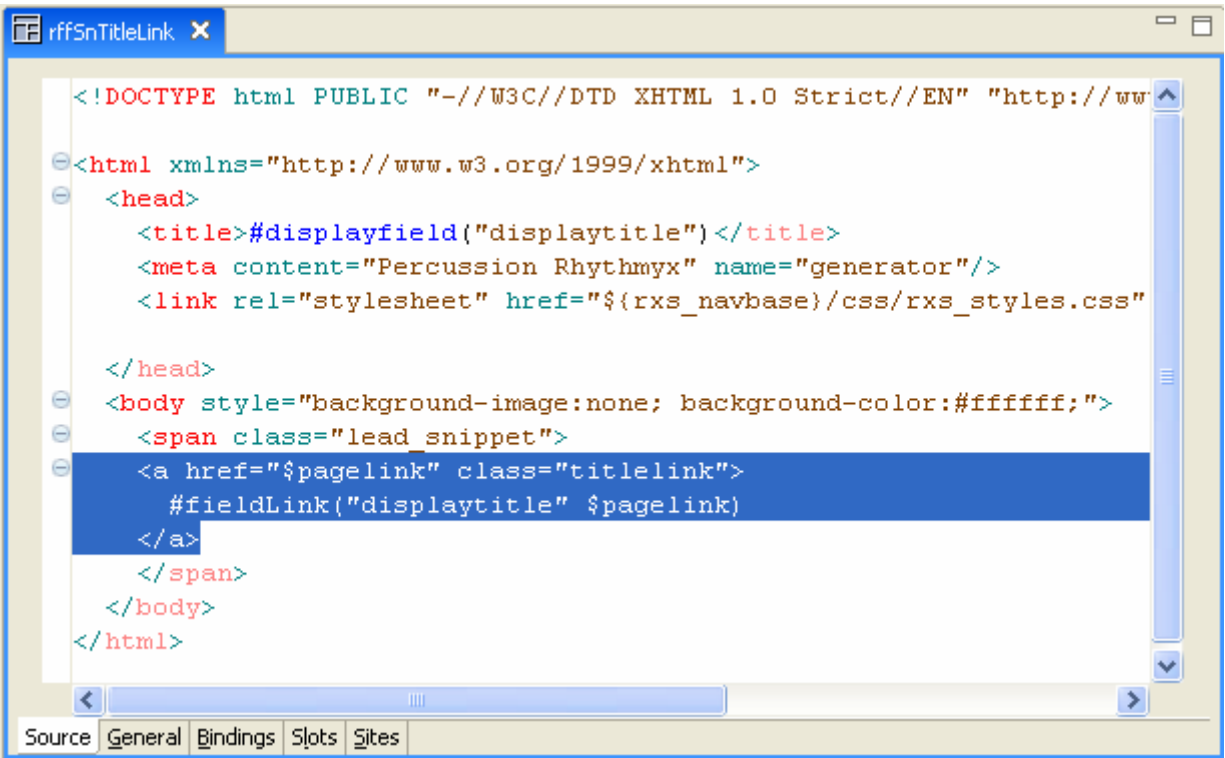
Rhythmyx creates the Template and displays the Template editor for the s-titlelink Template.

- 12 Add the Velocity macros to render the fields as illustrated in *Adding Velocity Macros to a Text Snippet* (on page 126).

- 13 To code the anchor tag:

- a) Specify `$pagelink` as the value of the `href` attribute of the anchor tag.
- b) For the contents of the anchor tag, specify `#fieldLink("displaytitle" $pagelink)`. Note that the binding variable `$pagelink` must be used both as the value of the `href` attribute of the anchor tag and as the `$pagelink` parameter of the `#fieldLink` macro.

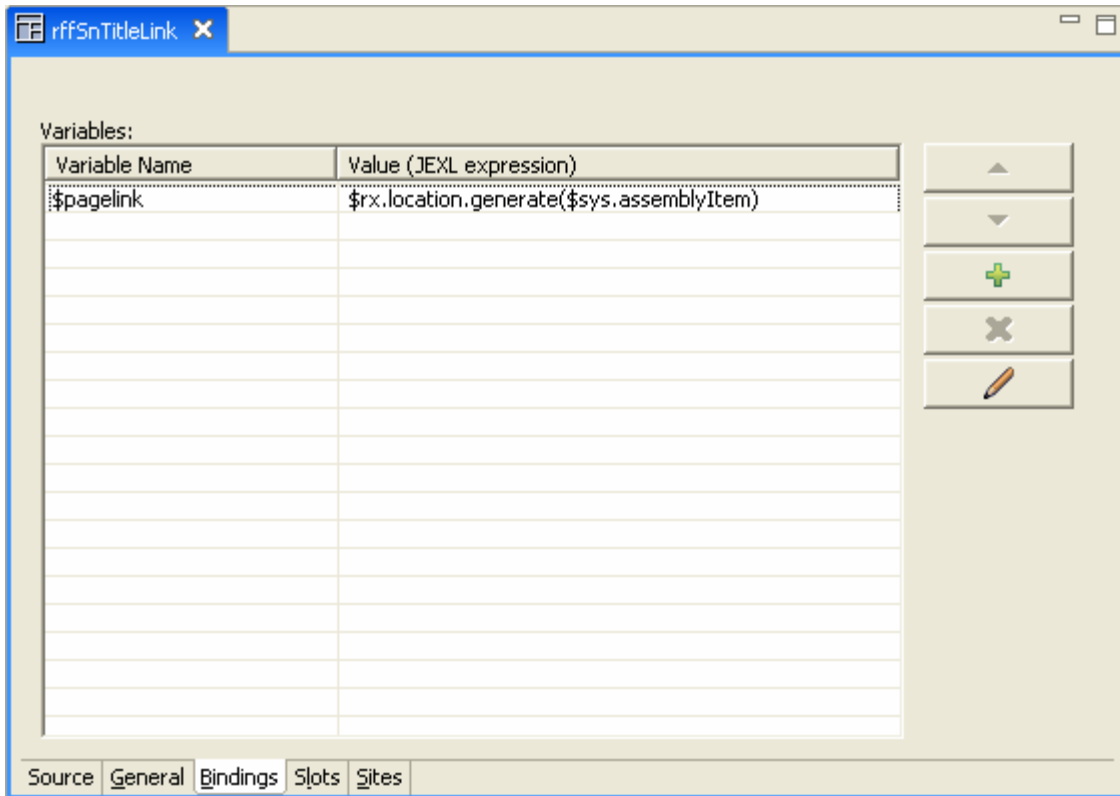
When you are finished, the Snippet Template HTML code resembles the following screenshot:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta content="Percussion Rhythmyx" name="generator"/>
    <link rel="stylesheet" href="${rxs_navbase}/css/rxs_styles.css" />
  </head>
  <body style="background-image:none; background-color:#ffffff;">
    <span class="lead_snippet">
      <a href="$pagelink" class="titlelink">
        #fieldLink("displaytitle" $pagelink)
      </a>
    </span>
  </body>
</html>
```

Figure 104: *rffSnTitleLink* HTML with the anchor tag highlighted. Note the binding variable *\$pagelink* as the value of the href attribute.

- 14** On the Bindings tab, add the binding  
`$pagelink=$rx.location.generate($sys.assemblyItem).`



*Figure 105: rffSnTitleLink Bindings tab showing the \$pagelink binding*

- 15** On the Button bar of the Rhythmyx Workbench, click the save button.

To test the template, in Content Explorer, find a Generic page Content Item, and preview the Template. It should render the text of the Display title field formatted as a hypertext link. When you click on the link, Rhythmyx should render a preview of the Generic page template of the Content Item (assuming a Page Template exists for the Content Type).

## Adding a Link to an Image File

An image Snippet includes the `<img>` tag, which uses the `src` attribute to specify the location of the image file. The file location must be generated dynamically using the `$rx.location.generate` function. In this case, you must also specify the Template that will be used to retrieve the image file.

We will use the `rffSnImageAndTitle` Snippet to illustrate the implementation of a Snippet that includes an image reference. This Snippet includes some Velocity markup to provide some context for the `<img>` tag. We will use the `rffBnImage` Template created earlier as the Image Template. The `rffSnImageAndTitle` Snippet uses the following HTML:

```
<html>
  <head>
    <title>Display Title"</title>
  </head>
  <body>
```

```
<div class="leftTables">
  <img src=published location of the image file alt=img_alt
field</img>
  <div>
    <span>Display Title)</span>
  </div>
</div>
</body>
</html>
```

We will assume that this code is stored in a file named `rffSnImageAndTitle.html`.

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To implement the `rffSnImageAndTitle` Snippet:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.  
The Rhythmyx Workbench displays the Type dialog of the Template wizard.
- 2 Choose the **Shared** radio button and click the **[Next]** button.  
The Rhythmyx Workbench displays the Output format dialog of the Template wizard.
- 3 In the **Assembler** drop list, choose *Velocity Assembler* (this is the default option). In the Output section of the dialog, choose the **Snippet** radio button. Click the **[Next]** button.  
The Rhythmyx Workbench displays the General properties dialog of the Template wizard.
- 4 In the **Template name** field, enter *rffSnImageAndTitle*. Change the value in the **Label** field to *Image and Title Snippet*.
- 5 In the **Description** field, enter *Renders the image with the Display Title*.
- 6 Click browse button next to the **Source** field, and use the browse dialog to find the file `rffSnImageAndTitle.html` and add it to the field.
- 7 In the **Available Communities** field, select *Enterprise Investments* and *Corporate Investments* and click **[>]** button to make this Template available to those Communities..
- 8 Click the **[Next]** button.  
The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.
- 9 This Template does not contain any Slots, so click the **[Finish]** button.  
Rhythmyx creates the Template and displays the Template editor for the `rffSnImageAndTitle` Template.
- 10 Add the Velocity macros to render the Display Title field as illustrated in *Adding Velocity Macros to a Text Snippet* (on page 126).  
Specify `$image` as the value of the `src` attribute of the `<img>` tag.
- 11 On the **Bindings** tab, add the binding  
`$image=$rx.location.generate($sys.assemblyitem, "rffBnImage")`
- 12 On the **Button** bar of the Rhythmyx Workbench, click the save button.

To test the template, in Content Explorer, find an Image Content Item, and preview Template. It should render the graphic with the Display Title underneath.



## Implementing Page Templates

A Page Template outputs a complete HTML page. In most cases, Page Templates contain Slots (although some Snippets may contain Slots as well). The main difference between a Page Template and a Snippet Template is that a Page Template produces a complete output HTML page, while a Snippet Template produces HTML for assembly into a Page or another Snippet.

The basic Page Template in FastForward is the rffPgGeneric Template; FastForward includes an example of this Template for each of the Sites included in the implementation (rffPgEIGeneric for the Enterprise Investments Site and rffPgCIGeneric for the Corporate Investments Site). These Templates contain two local fields (Display Title and Body) and two Slots:

- Sidebar Slot
- List Slot

We will use the rffPgEIGeneric Page Template to illustrate the creation of Page Templates.

Name: rffPgEIGeneric

Label: P-EI Generic

Content Type: Generic

Assembler: Velocity Assembler

Output: Page

Global Template: Default

Publish: Always

Active Assembly Format: Normal

MIME Type: Text/HTML

Character Set: <null>

Location Prefix: <null>

Location Suffix: <null>

Bindings: None

Communities: Enterprise Investments

Contained Slots: Sidebar Slot, List Slot

Sites: Enterprise Investments

Included Fields: Display Title, Body

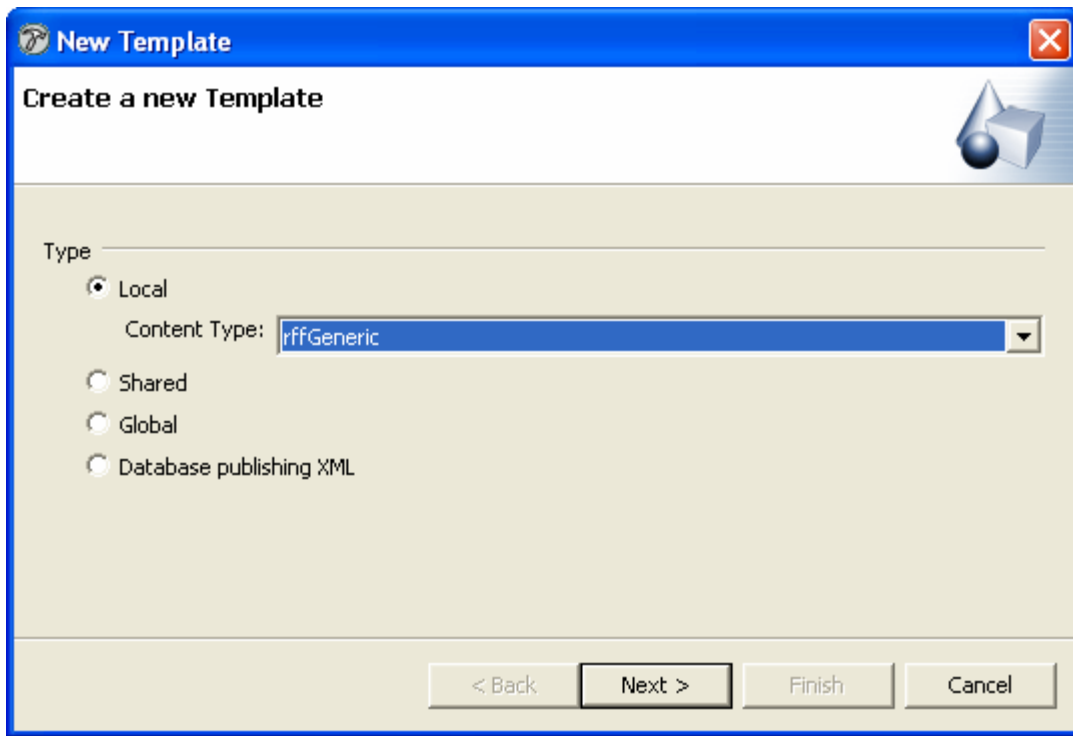
We will assume that the HTML for this Snippet is stored in an HTML file named rffPgEIGeneric.html, which was created during the modeling and design process.

## Creating the Page Template Object

To create the rffPgEIGeneric PageTemplate object:

- 1 In the Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.

The Rhythmyx Workbench displays the Type dialog of the Template wizard.



*Figure 106: Type dialog for rffPGEIGeneric Page Template*

- 2 Choose the Type-specific radio button. In the Content Type field, choose *Generic*. Click the [Next] button.

The Rhythmyx Workbench displays the Output format dialog of the Template wizard.

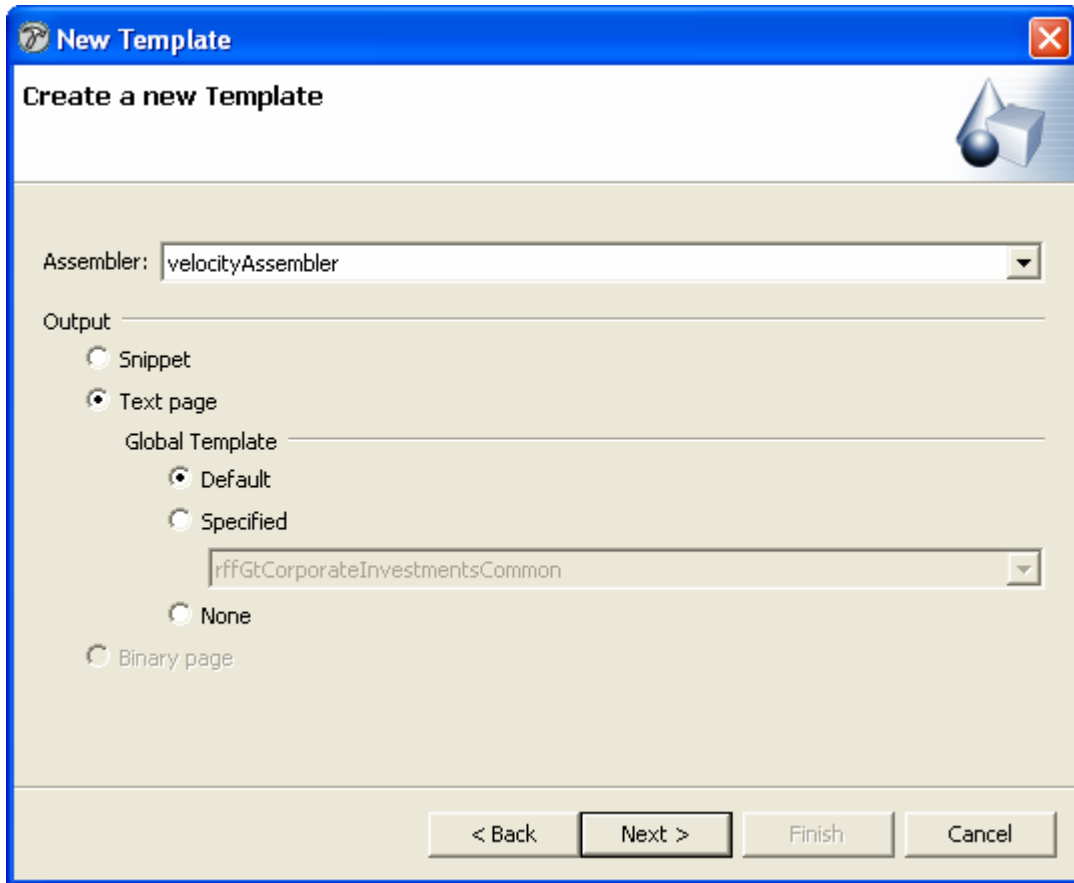


Figure 107: Assembler dialog for *rffPGEIGeneric Page Template*

- 3 In the **Assembler** drop list, choose *Velocity Assembler* (this is the default option). In the **Output** section of the dialog, choose the **Page** radio button. Under **Global Template**, leave **Default** selected, which uses the default **Global Template** for the Site. (NOTE: This is the default option.) Click the **[Next]** button.

The Rhythmyx Workbench displays the **General properties** dialog of the Template wizard.

- 4 In the **Template name** field, enter *rffPgEIGeneric*. Modify the value in the **Label** field to *Generic Page Template*.
- 5 In the **Description** field, enter *Renders Generic Content Items as HTML pages*.
- 6 Click **browse** button next to the **Source** field, and use the browse dialog to find the file *rffPgGeneric.html*, and add it to the field.
- 7 In the **Available Communities** field, select *Enterprise Investments* and *Corporate Investments* and click **[>]** button to make this Template available to those Communities.
- 8 Click the **[Next]** button.

The Rhythmyx Workbench displays the **Contained Slots** dialog of the Template wizard.

- 9 Select the **List Slot** and **Sidebar Slot** and the click **[>]** button to add them to the Template.

**10** Click the [Finish] button.

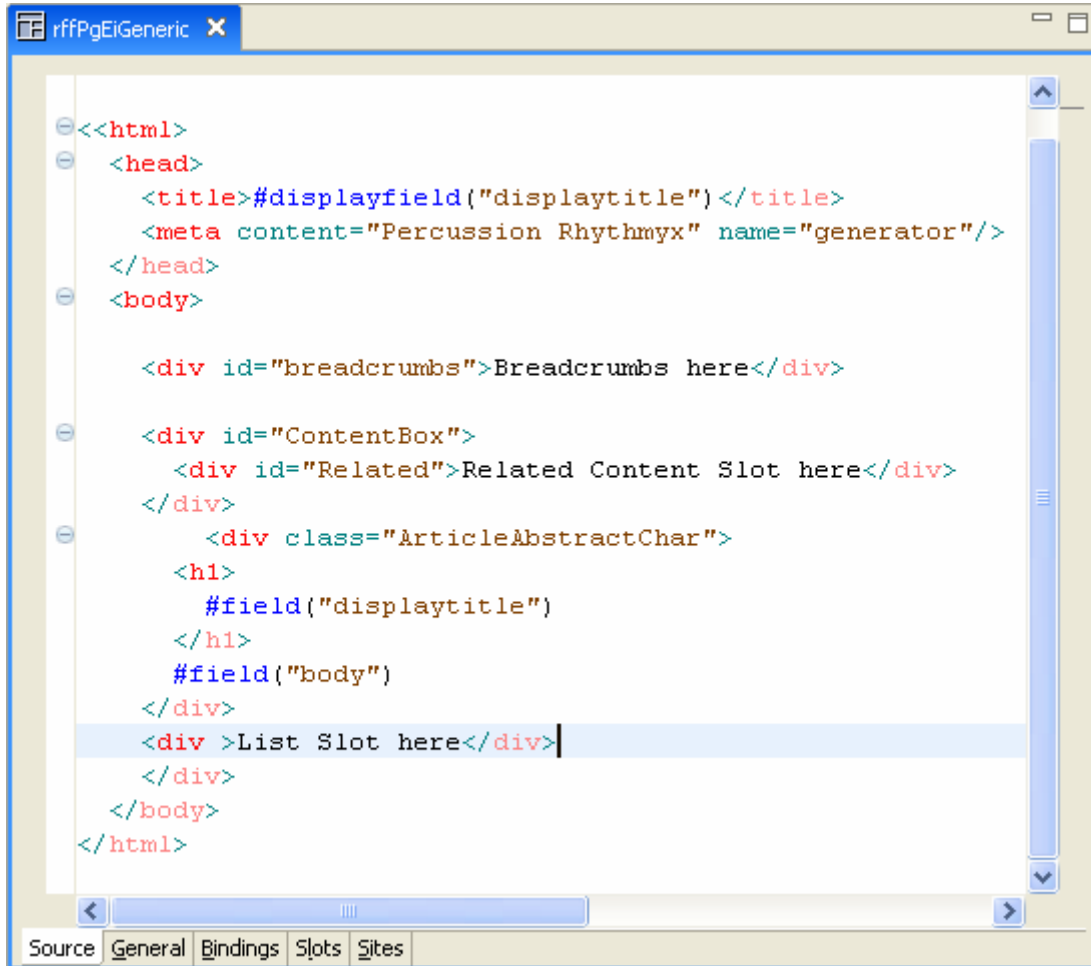
Rhythmyx creates the Template and displays the Template editor for the rffPgEIGeneric Template.

## Adding Velocity Macros to a Page Template

The following screenshot illustrates the rffPgGeneric HTML before adding Velocity markup.

Figure 108: Original HTML of the rffPgEIGeneric Template

Add the Display Title and Body fields as illustrated in “*Adding Velocity Macros to a Snippet* (see “Adding Velocity Macros to a Text Snippet” on page 126)”. When these macros have been added, the Velocity tab resembles the following screenshot:



```

<<html>
<head>
  <title>#displayfield("displaytitle")</title>
  <meta content="Percussion Rhythmyx" name="generator"/>
</head>
<body>

  <div id="breadcrumbs">Breadcrumbs here</div>

  <div id="ContentBox">
    <div id="Related">Related Content Slot here</div>
  </div>
  <div class="ArticleAbstractChar">
    <h1>
      #field("displaytitle")
    </h1>
    #field("body")
  </div>
  <div >List Slot here</div>
</div>
</body>
</html>

```

Figure 109: HTML of the *rffPgEIGeneric* Template with field macros

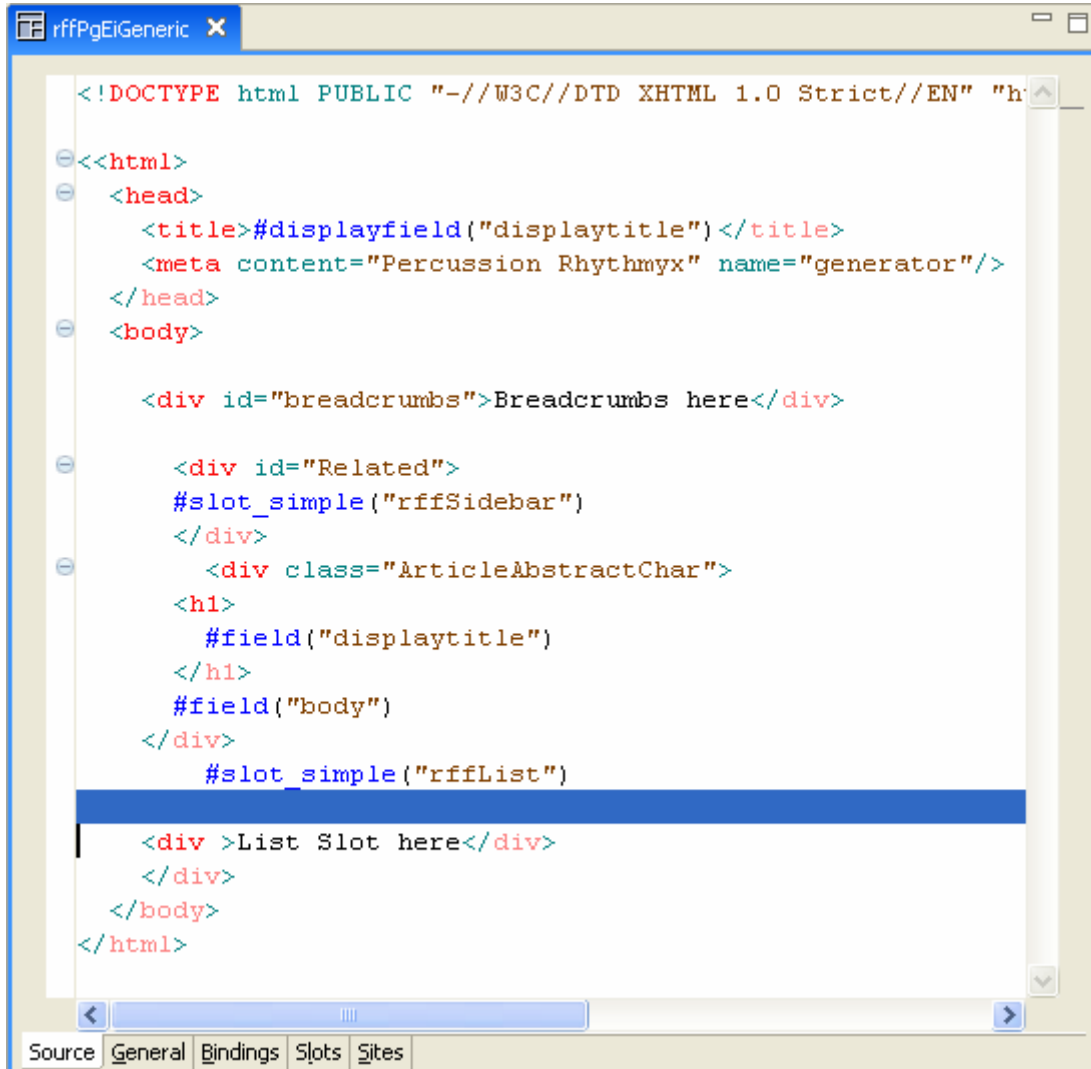
Rhythmyx includes three predefined macros for Slots. The simplest Slot macro is the `#slot_simple` macro.

```
#slot_simple(slotname)
```

The `slotname` property specifies the name of the Slot you want to include in the output. This macro renders only the Content Items in the Slot.

Thus, the markup for the two Slots specified would be the following:

```
#slot_simple("rffSidebar")  
#slot_simple("List Slot")
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "h  
<<html>  
<head>  
  <title>#displayfield("displaytitle")</title>  
  <meta content="Percussion Rhythmyx" name="generator"/>  
</head>  
<body>  
  
  <div id="breadcrumbs">Breadcrumbs here</div>  
  
  <div id="Related">  
    #slot_simple("rffSidebar")  
  </div>  
  <div class="ArticleAbstractChar">  
    <h1>  
      #field("displaytitle")  
    </h1>  
    #field("body")  
  </div>  
    #slot_simple("rffList")  
  
  <div >List Slot here</div>  
  </div>  
</body>  
</html>
```

Figure 110: rffPgIEGeneric Template with #slot\_simple macros

This markup produces the following output:

Breadcrumbs here

**12 EASY STEPS TO PREPARING YOUR ESTATE PLAN** [Get your estate in order while you're capable](#) [Pick the right executor to handle your estate](#)

By Paul Baker

Sidebar Slot

- Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
- If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
- Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
- Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
- If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
- Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
- Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
- Choose a guardian for your children.
- Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
- Review your estate plan every few years, even if your situation is pretty much the same. Laws change constantly, and your planning may be out of date.
- Don't keep your insurance policies in your safe-deposit box. This delays filing for death benefits.
- There are three kinds of joint ownership. If you die, your share does not automatically go to the other owner. Make sure you have the right kind of joint ownership for your needs.

List Slot

It's important to keep in mind that your estate-planning needs are probably much less complicated than they seem, even if they don't include all of the topics touched upon here.

[Better Investing National Convention & World Federation of Investors](#) [Putting patients in the driver's seat](#) [Six Things You Should Know When Getting a Mortgage Loan](#)

Figure 111: Generic Page Preview showing the output of the #slot\_simple macro

For users familiar with earlier versions of Rhythmyx, this markup produces the equivalent of only the Snippet.

Another option is to wrap each Snippet instance in some HTML markup. To implement this option, use the `#slot_wrapped` macro:

```
#slot_wrapped(slotname beforetext aftertext)
```

where `beforetext` and `aftertext` is the text (usually HTML markup) you want to output with each Content Item in the Slot. For users familiar with earlier versions of Rhythmyx, the `beforetext` and `aftertext` are equivalent to the Snippet Wrapper.

For example, if we change the markup in the Sidebar Slot to add a break before and after each Content Item in the Slot:

```
#slot_wrapped("rffSidebar" "<br>" "</br>")
```



Rhythmyx returns the following output:

Breadcrumbs here

**12 EASY STEPS TO PREPARING YOUR ESTATE PLAN**

By Paul Baker

**Sidebar Slot**

[Get your estate in order while you're capable](#)

[Pick the right executor to handle your estate](#)

- Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
- If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
- Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
- Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
- If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
- Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
- Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
- Choose a guardian for your children.
- Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
- Review your estate plan every few years, even if your situation is pretty much the same. Laws change constantly, and your planning may be out of date.
- Don't keep your insurance policies in your safe-deposit box. This delays filing for death benefits.
- There are three kinds of joint ownership. If you die, your share does not automatically go to the other owner. Make sure you have the right kind of joint ownership for your needs.

**List Slot**

It's important to keep in mind that your estate-planning needs are probably much less complicated than they seem, even if they don't include all of the topics touched upon here.

[Better Investing National Convention & World Federation of Investors](#)

[Putting patients in the driver's seat](#)

[Six Things You Should Know When Getting a Mortgage Loan](#)

Figure 112: Generic Page Preview showing the output of the #slot\_wrapped macro

Notice the extra whitespace around the Content Items in the Slots. (Note: In the `rffPgGeneric` Template in `FastForward`, a break is defined after each Content Item in each Slot. We modified the markup in this case to demonstrate both the `beforetext` and `aftertext` attributes of the `#slot_wrapped` macro.)

The richest Slot macro is the `#slot` macro:

```
#slot(slotname header beforetext aftertext footer params)
```

Where

- `slotname` is the name of the Slot
- `header` is any text to include before any Slot contents
- `beforetext` is any text to include before each Content Item in the Slot, as illustrated above with the `#slot_wrapped` macro
- `aftertext` is any text to include after each Content Item in the Slot, as illustrated above with the `#slot_wrapped` macro
- `footer` is any text to include after any Slot contents

(For users of earlier versions of Rhythmyx, the header and footer are equivalent to the Slot Wrapper)

- `params` are any parameters you want to pass with the Slot.

Thus, the markup for the List Slot on the `rffPgGeneric` Template in `FastForward` is:

```
#slot("rffList" "<div class="list"><span  
class="relatedHeader">Related...</span><br />" "</div>" "" "<br/>" "")
```

where

`"rffList"` is the name of the Slot.

`<div class="list"><span class="relatedHeader">Related...</span><br />` is the header for the Slot.

`</div>` is the footer for the Slot

`<br/>` is the `aftertext` for each Content Item in the Slot

Note that there is no value for either the `beforetext` attribute or the `params` attribute, but these must be included in the markup as nulls. Nulls are denoted by an empty set of quotation marks.

Diagnosing errors when adding so much text can be problematic, so Best Practice is to specify the text as a set of local bindings, then specify the bindings as the values for the parameters. The bindings are defined using Velocity `#set` directives, as illustrated in the following code:

```
#set( $start_slot = '<div class="list"><span  
class="relatedHeader">Related...</span> <br />' )  
#set( $start_snippet = '' )  
#set( $end_snippet = '<br/>' )  
#set( $end_slot = '</div>' )  
  
#slot("rffList" $start_slot $start_snippet $end_snippet $end_slot '')
```

This markup produces the following output:

Enterprise Investments Home

## 12 EASY STEPS TO PREPARING YOUR ESTATE PLAN

By Paul Baker

Sidebar Slot

Get your estate in order while you're capable. Pick the right executor to handle your estate.

- Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
- If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
- Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
- Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
- If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
- Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
- Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
- Choose a guardian for your children.
- Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
- Review your estate plan every few years, even if your situation is pretty much the same. Laws change constantly, and your planning may be out of date.
- Don't keep your insurance policies in your safe-deposit box. This delays filing for death benefits.
- There are three kinds of joint ownership. If you die, your share does not automatically go to the other owner. Make sure you have the right kind of joint ownership for your needs.

List Slot

It's important to keep in mind that your estate-planning needs are probably much less complicated than they seem, even if they don't include all of the topics touched upon here.

**Related...**

[Better Investing National Convention & World Federation of Investors](#)

[Putting patients in the driver's seat](#)

[Six Things You Should Know When Getting a Mortgage Loan](#)

Figure 113: Generic Page Preview showing the output of the #slot macro

## Using the params attribute of the #slot Macro

Any parameters you define in the params attribute of the #slot macro are passed directly to the Slot Content Finder for the specified Slot. Uses of the params attribute include:

- Use these parameters instead of the parameters of the Slot Content Finder parameters, hard-coding the parameters into the Slot.
- Use these parameters in a specific instance of the Slot in a Template to override the parameters defined for the Slot.

## Adding Child Data to a Page Template

To include content from a Child Editor on a Page Template, use the #children macro:

```
#children(childname template $header $beforetext $aftertext $footer)
```

where

`childname` is the name of the child editor whose contents you want to add to the Template

`template` is the Template used to format the content from the child editor

`header` is any text to include before any child table rows; typically, this is the `<table>` tag, with its formatting; if the table has a heading row, it would also be included in the header.

`beforetext` is the text you want to include before each child row

`aftertext` is the text you want to include after each child row

`footer` is any text to include after any child table rows; typically, this is the closing tag for the table (`</table>`)

For example, to add the `event_location` child table we added to the Event Content type (see *Creating a Content Type with a Child Field Set* (see page 240) for details), we would need to create a Snippet Template to format the child content. This Template consists of two `<td>` tags to define two columns in the child table: one for the address fields with commas inserted between them, and one for the contact field. Assume for the purposes of this example that we have created a Template named `rffSnEventLocation` consisting of the following markup:

```
<td>#field("rx:event_address"),  
#field("rx:event_city"),#field("rx:event_state")</td>  
<td> contact:#field("rx:event_contact")</td>
```

---

**NOTE:** Templates used to format child snippets should not be associated with any Content Type. If you associate the Template with a Content Type, it will be listed in the available previews for that Content Type; previewing of these Templates returns an error, however. To preview a child Snippet Template, add it to a Page Template, then preview the Page.

---

The #children macro would be coded as follows:

```
#set ($header = ' <table> ')  
#set ($beforetext = '<tr>')  
#set ($aftertext = '</tr>')  
#set ($footer = '</table>' )
```

```

#children("event_location" "rffSnEventLocation" $header
$beforetext $aftertext $footer )

```

The screenshot shows a code editor window titled "rffPgEIEvent". The code is as follows:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta content="Percussion Rhythmyx" name="generator"/>
  </head>
  <body>
    <div id="breadcrumbs">
      #slot("rffNav" "" "" "" "" "template=rffSnEiNavBreadcrumbs")
    </div>
    <div id="ContentBox">
      <div id="MainContent">
        <div class="ArticleAbstractChar">
          <h1>
            #field("displaytitle")
          </h1>
          #field("body")
          <br />
          <span class="displaytitle">Details</span>
          #set( $header = '<table> <tr> <td>Event City</td> <td>Event Sta
          #set( $beforetext = '<tr>' )
          #set( $aftertext = '</tr>' )
          #set( $footer = '</table>' )

          #children("event_location" "rffSnEventLocation" '$beforetext' '

        </div>

        #slot("rffList" '<div class="list"><span class="relatedHeader">
          </div>
        </div>
      </body>
    </html>

```

The editor interface includes a scroll bar on the right and a tab bar at the bottom with buttons for "Source", "General", "Bindings", "Slots", and "Sites".

Figure 114: rffPgEIEvent Template with #children Macro

This code results in the following output:

Enterprise Investments Home > Products and Services > Funds

### ENTERPRISE INVESTMENTS TO OFFER FREE "INVESTING 101: INTRODUCTION TO INVESTING" SEMINARS TO BENEFIT THE JUMP\$TART COALITION

NEW YORK, November 12, 2004 - Enterprise Investments Group, Inc. (aElou), a leading online financial services firm and the world's second-largest discount broker, today announced it will conduct a free series of "Investing 101: Introduction To Investing" seminars at customer retail outlets across the U.S. For every person attending one of these seminars, Enterprise Investments will make a donation to the Jump\$tart Coalition - a charitable organization that promotes financial literacy among students.

The seminars, which will cover the basics of investing and begin in mid-December, will be presented at all of the more than over 170 Enterprise Investments customer retail outlets nationwide, as well as other select locations. They are open to all adults and children over the age of 14.

To kick-off this effort, Frank J. Peter, President and Chief Operating Officer of Enterprise Investments, will make the first presentation of the "Investing 101: Introduction to Investing" seminar on December 23, 2004, at 6:00 PM EST, in New York City at the Waldorf Astoria Hotel. A total of twenty seminars will be held that week in major markets around the country.

"Enterprise Investments is presenting a summer school session that everyone will want to attend," says Mr. Peter. "You're never too young or too old to take control of your financial future. At times, getting started is the most difficult step. With these educational seminars, we want to empower individuals to make their own choices and acquaint them with the basics of investing."

"Additionally, Enterprise Investments will make a contribution to the Jump\$tart Coalition for every student that attends a seminar. This is a tremendous opportunity for us to invest in the future of America's young men and women."

These presentations will include basic investment principles and how to assess which types of investment instruments may be appropriate for individual investors and their particular situation. Furthermore, investment services available through Enterprise Investments will be examined.

In conjunction with the seminars, Enterprise Investments Associates are now listed as speaker resources to appear at schools nationwide through Jump\$tart's Educational Clearinghouse. You can gain access to this resource database at [www.jumpstart.org](http://www.jumpstart.org).

To reserve a seat at an "Investing 101: Introduction to Investing" seminar, please visit our website, [www.EnterpriseInvestmentsBankingOnline.com](http://www.EnterpriseInvestmentsBankingOnline.com), and click on Introduction to Investing Seminar to locate the seminar nearest you, or call 800-934-4448.

Enterprise Investments, (aElou), provides investors with a broad range of brokerage, mutual fund, banking and other consumer financial products on an integrated basis. In the United States, Enterprise Investments is the only online discount broker to have a retail outlet in all 50 states. Worldwide, Enterprise Investments currently services 3.8 million customer accounts in the United States, Canada, the United Kingdom, Australia, Hong Kong, Japan and India. Enterprise Investments can be found on the Internet at [www.EnterpriseInvestmentsBankingOnline.com](http://www.EnterpriseInvestmentsBankingOnline.com) and on America Online at Keyword: Enterprise Investments.

The Jump\$tart Coalition for Financial Literacy's purpose is to evaluate the financial literacy of young people; develop, disseminate, and encourage the use of guidelines for grades K-12; and promote the teaching of personal finance.

The Jump\$tart Coalition believes that all young people need to have the financial literacy necessary to make informed financial decisions.

**Details**

<b>Start:</b>	12.23.2004
<b>End:</b>	12.23.2004
<b>Type:</b>	Conference

**Locations**

216 Granite Avenue , Bangor , Maine	contact: Lisa Kerr
24816 Camino Real Way , Monterey , California	contact: Ed Wong
4873 Brazos Road , Bryan , Texas	contact: Rita Perez
1587 Wasburn Road , Topeka , Kansas	contact: Kent Hoyt

Figure 115: Preview of Event Content Item showing child data table

## Implementing Global Templates

Generally, all of the pages on a web site share a common “look and feel”, meaning they share a common page structure, use the same color palette, and share common graphics. In some cases, different sections of a site may vary in format, but all of the pages of each individual section share the same look and feel.

Rhythmyx uses Global Templates to ensure this consistency. A Global Template defines the general structure of the page and is usually responsible for rendering the outer wrapper for most, if not all, pages on the site. The wrapper includes page headers and footers and elements of the HTML <head>, such as references to the CSS files that implement the specific formatting of the HTML markup in the published page. When publishing a page, Rhythmyx merges the Local Template with a specified Global Template to produce the final page markup for rendering. The Global Template also commonly includes Managed Navigation elements that Rhythmyx adds to the final published page.

The simplest approach to page design is to add a common banner across the top of the page. The banner may include some basic navigation:



### 12 EASY STEPS TO PREPARING YOUR ESTATE PLAN

By Paul Baker

- Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
- If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
- Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
- Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
- If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
- Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death and your estate's administrative and probate costs, as well as your heirs' legal fees.

*Figure 116: Page with Banner Global Template*

All pages on the site include the banner, but the content below the banner differs from page to page.

Another approach is the “inverted-L”. This design starts with a banner and adds a dynamic navigation bar down the left-hand side of the page.

**EI Enterprise Investments** Search **GO ▶** Region/

[About Enterprise Investments](#)
[Investment Advice](#)
[Mortgages and Home Finance](#)

**PICK THE RIGHT EXECUTOR TO HANDLE YOUR ESTATE**

One of the most important decisions you'll make in your estate planning is whom you name as the executor of your estate.

The executor controls your estate from the time you die until the last federal and state tax returns have been filed and all your assets have been distributed to your beneficiaries. It is this person who ensures that what you wanted to happen - does happen.

***If you don't find one, the state will***

If you don't have a will, or you make one without naming an executor, the probate court will name an administrator who basically does the same job, except that the state decides who gets your money. Unlike someone that you selected, this executor may be someone you don't know or, even worse, someone you do know but intensely dislike.

***Not a glamorous job***

An executor's job is not a glamorous role. It's typically unappreciated by other family members and involves tedious work with few obvious rewards. He or she has to set up the estate with the Internal Revenue Service and the respective state revenue department, get a tax identification number, and file estate and income taxes until the estate is settled and closed.

***What's the value of Mom's blue teapot?***

The easy part is establishing a value on your bank accounts or stock or mutual fund investments. There are specific values placed on those assets and your executor can either choose to assess their value at the time of your death or wait until nine months after your death. It's a different story for items such as furniture, real estate, automobiles, jewelry and antiques, which don't have established prices. And if you own your own business, the executor faces serious issues of determining its value.

***The surviving spouse as executor***

Married people usually name each other as executor, although it's not uncommon for a spouse to discover that someone else has been named after her or his spouse has died.

Naming your spouse as the executor is usually a good idea, but you'll want to consider a few things. Your spouse will already be emotionally drained upon your death, so taking on the role of executor may be too much of a burden. If your spouse is not

**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

**Markets**

DJIA	8,022.02	+108.81
NASDAQ	1,327.99	+14.16
S&P500	850.17	+11.24
RJQ	45.09	-1.18
WKM	13.16	+4.98
YTB	23.73	-18.71
TWUR	56.27	-1.01

**Rates**

MORTGAGES	Rate	APR
30-Year Fixed	5.25	5.55
15-Year Fixed	4.75	5.17
7-Year Arm	4.37	4.60

Home Equity	Rate	APR
Line of Credit	3.90	4.25
Installment	6.75	6.75

Figure 117: Page with "inverted-L" Global Template



In this design, the banner and left navigation are shared by all pages. The content contained in the “inverted-L” changes with each page.

A third common design is the “C-clamp, which adds navigation to the bottom of the inverted-L. The unique content of each page is contained inside of the “C-clamp”.



Figure 118: Page with "C-clamp" Global Template

The Global Template defines the overall page structure and common outer wrapper. The Local Template specifies the formatting of the content that differs from page to page.

In the Site registration, you must specify the default Global Template for the Site. Rhythmyx uses this Global Template unless a different Global Template is specified. You can override the default Global Template in two ways:

- You can specify a Global Template for a specific Folder. Rhythmyx will use the Global Template to format all Content Items in the Folder, and in any Subfolders.
- You can specify a Global Template for a specific Local Template. Rhythmyx will use that Global Template whenever formatting Content Items using the Local Template.

NOTE: New Global Templates are not available in Content Explorer until Content Explorer has been restarted after the Global Template has been saved.

To demonstrate the process of creating a Global Template, we will create the Enterprise Investments Global Template (rffGtEnterpriseInvestmentsCommon), which is the only Global Template defined for the Enterprise Investments Site. This Template should only be available on the Enterprise Investments Site. We will assume that the HTML is defined in a file named rffGtEnterpriseInvestmentsCommon.html, which was developed during Modeling and design.

## Creating the Global Template Object in the Rhythmyx Workbench

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Enterprise Investments Global Template object:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.

The Rhythmyx Workbench displays the Type dialog of the Template wizard.

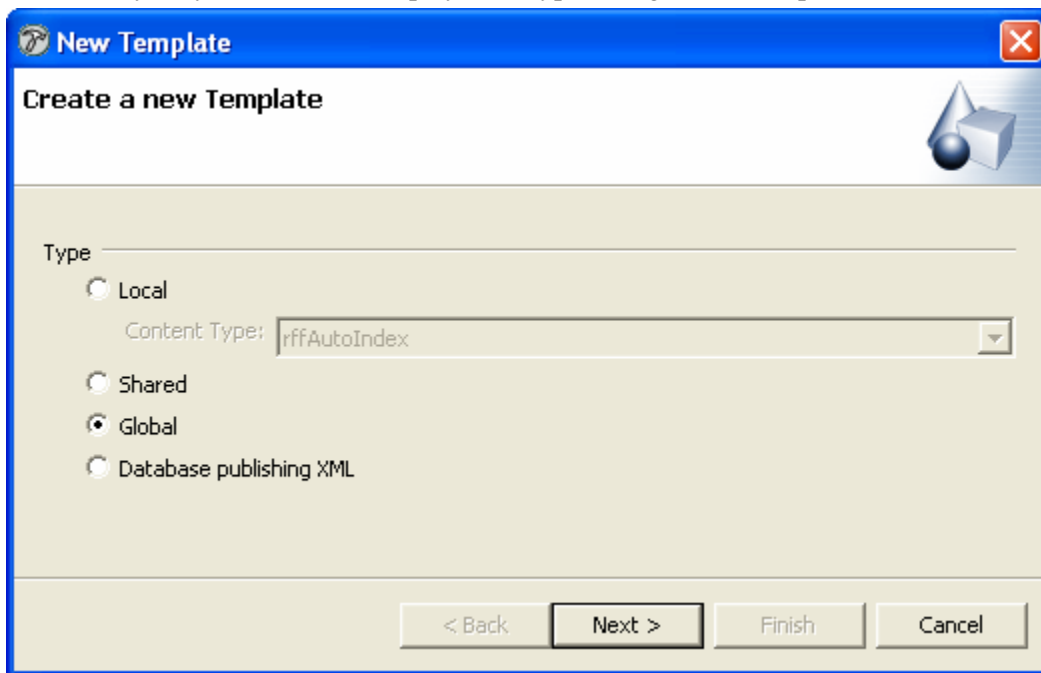


Figure 119: Template Wizard Type dialog with Global radio button selected.

- 2 Choose the **Global** radio button and click the [Next] button.

The Rhythmyx Workbench displays the General properties dialog of the Template wizard.

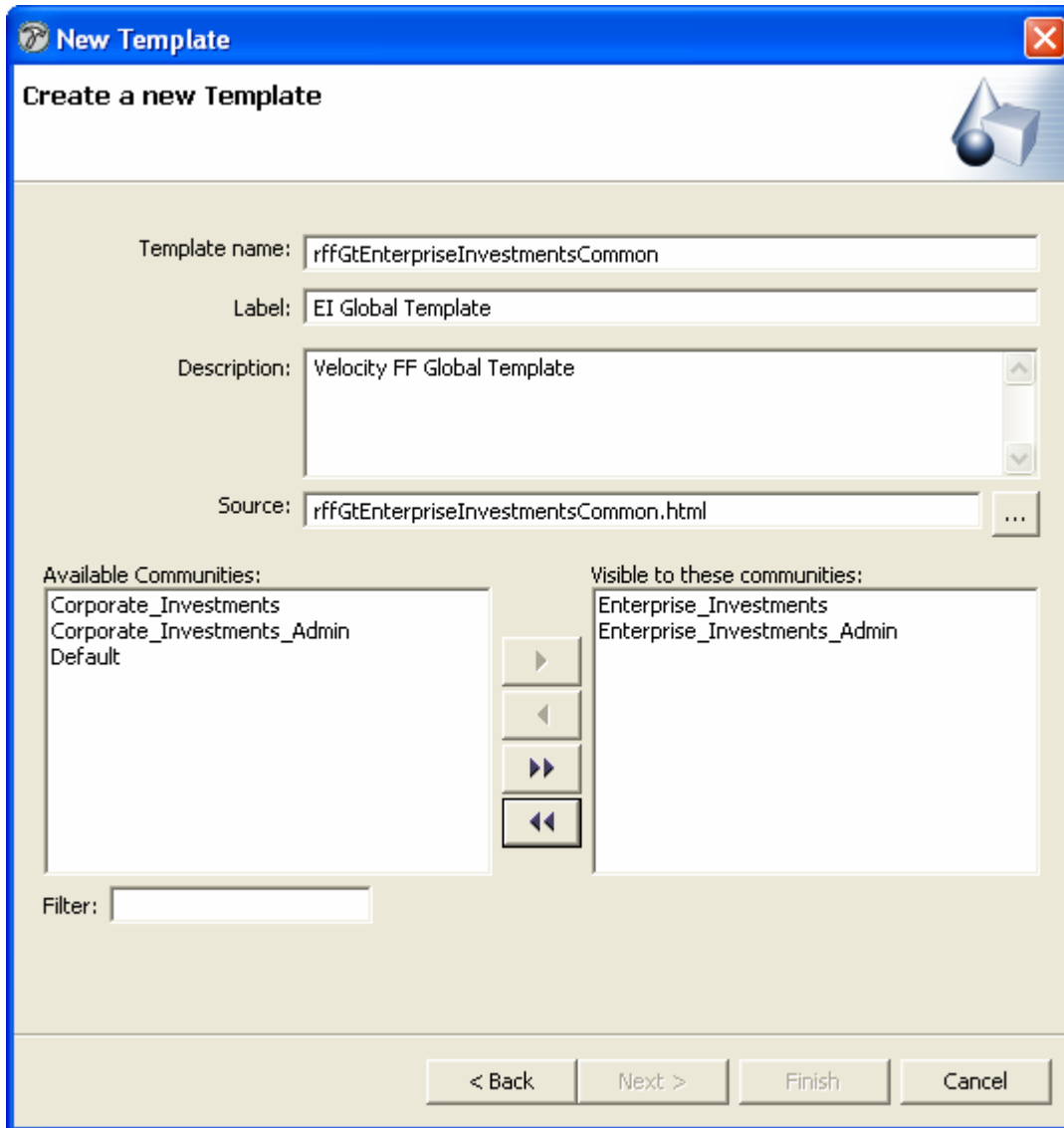


Figure 120: Template wizard with general data for the `rffGtEnterpriseInvestmentsCommon` Template

- 3 In the **Template name** field, enter *Enterprise\_Investments\_Global Template*. In the **Label** field change the underscores to spaces..
- 4 In the **Description** field, enter *Global Template for the Enterprise Investments Site*.
- 5 Click browse button next to the **Source** field, and use the browse dialog to find the file `rffGtEnterpriseInvestmentsCommon.html`, and add it to the field.
- 6 In the **Available Communities** field, select *Enterprise\_Investments* and *Enterprise\_Investments\_Admin* then click [**>**] button to make this Template available to that Community. Do not add the Corporate Investments Communities, which should not have access to the Enterprise Investments Global Template.
- 7 Click the [**Next**] button.

The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

- 8 The Enterprise Investment Global Template does not contain any Slots, so click the [Finish] button.

Rhythmyx creates the Template and displays the Template editor for the `rffGtEnterpriseInvestmentsCommon` Template.

## Adding Local Content to the Global Template HTML

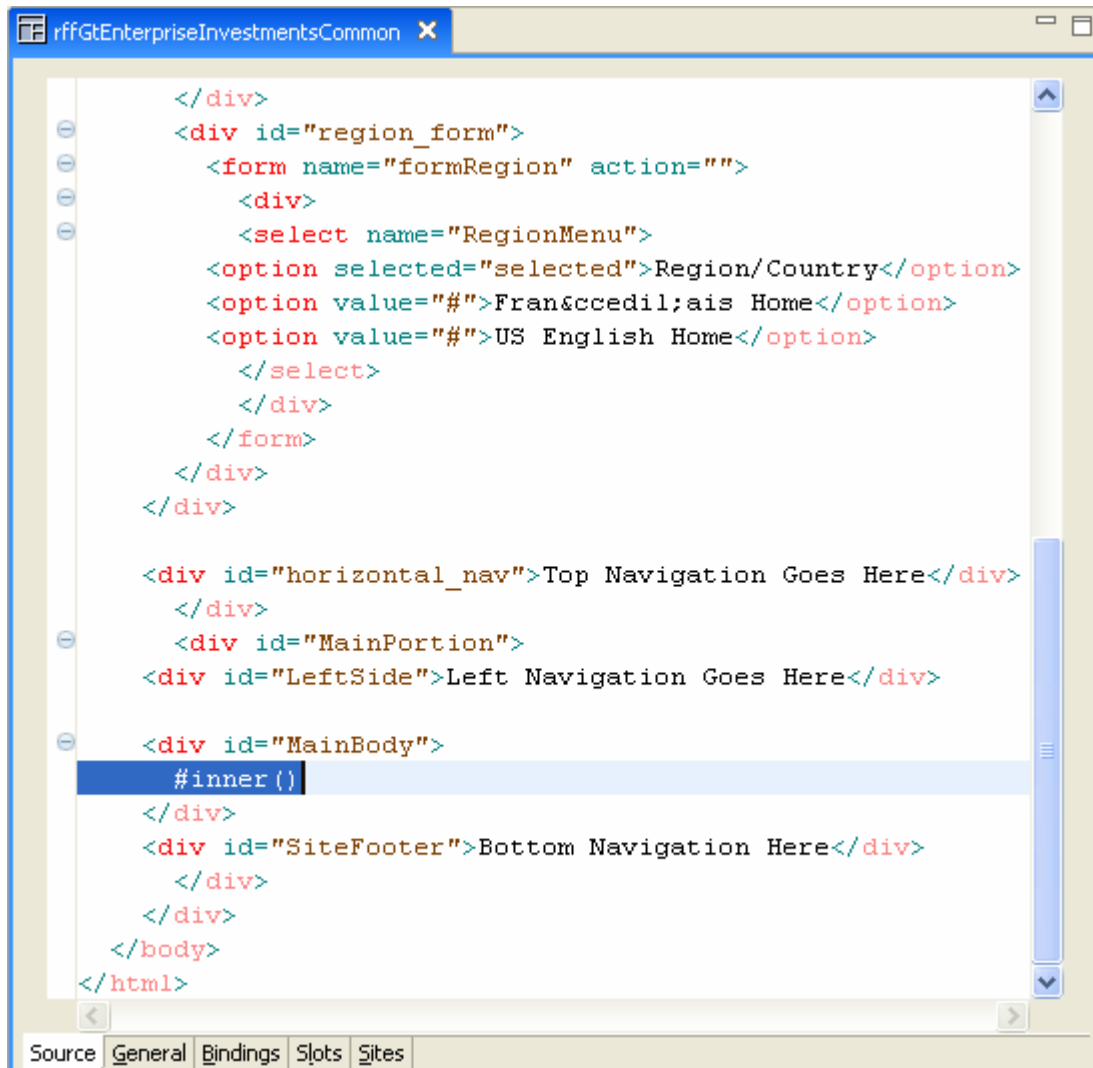
In the raw markup of the Enterprise Investments Global Template, we have inserted a note “Local Content Goes Here” to denote the location of the local content in the Global Template:



```
src="../../../web_resources/images/go_black.gif"
width="34" height="12" />
</div>
</form>
</div>
<div id="region_form">
  <form name="formRegion" action="">
    <div>
      <select name="RegionMenu">
        <option selected="selected">Region/Country</option>
        <option value="#">Fran&ccedil;ais Home</option>
        <option value="#">US English Home</option>
      </select>
    </div>
  </form>
</div>
</div>
<div id="horizontal_nav">Top Navigation Goes Here</div>
</div>
<div id="MainPortion">
<div id="LeftSide">Left Navigation Goes Here</div>
<div id="MainBody">
  Local Content Goes Here
</div>
<div id="SiteFooter">Bottom Navigation Here</div>
</div>
</div>
</body>
</html>
```

Figure 121: `rffGtEnterpriseInvestmentsCommon` Template HTML with location of local content highlighted

To include the content of a Local Template, use the `#inner` macro. This macro does not include any attributes.



```
</div>
<div id="region_form">
  <form name="formRegion" action="">
    <div>
      <select name="RegionMenu">
        <option selected="selected">Region/Country</option>
        <option value="#">Fran&ccedil;ais Home</option>
        <option value="#">US English Home</option>
      </select>
    </div>
  </form>
</div>
</div>

<div id="horizontal_nav">Top Navigation Goes Here</div>
</div>
<div id="MainPortion">
<div id="LeftSide">Left Navigation Goes Here</div>

<div id="MainBody">
  #inner()
</div>
<div id="SiteFooter">Bottom Navigation Here</div>
</div>
</div>
</body>
</html>
```

Figure 122: `rffGtEnterpriseInvestmentsCommon` Template with `#inner` macro added

Previewing a Content Item using this Template produces the following results:

Top Navigation Goes Here  
Left Navigation Goes Here  
[Enterprise Investments Home](#)

## 12 Easy Steps to preparing your estate plan

By Paul Baker

1. Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
2. If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
3. Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
4. Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
5. If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
6. Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
7. Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
8. Choose a guardian for your children.
9. Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
10. Review your estate plan every few years, even if your situation is pretty much the same. Laws change constantly, and your planning may be out of date.
11. Don't keep your insurance policies in your safe-deposit box. This delays filing for death benefits.
12. There are three kinds of joint ownership. If you die, your share does not automatically go to the other owner. Make sure you have the right kind of joint ownership for your needs.

*Figure 123: Preview of rffEnterpriseInvestmentsCommon Template. Locations for Managed Navigation are noted with text.*

## Adding Managed Navigation to the Global Template

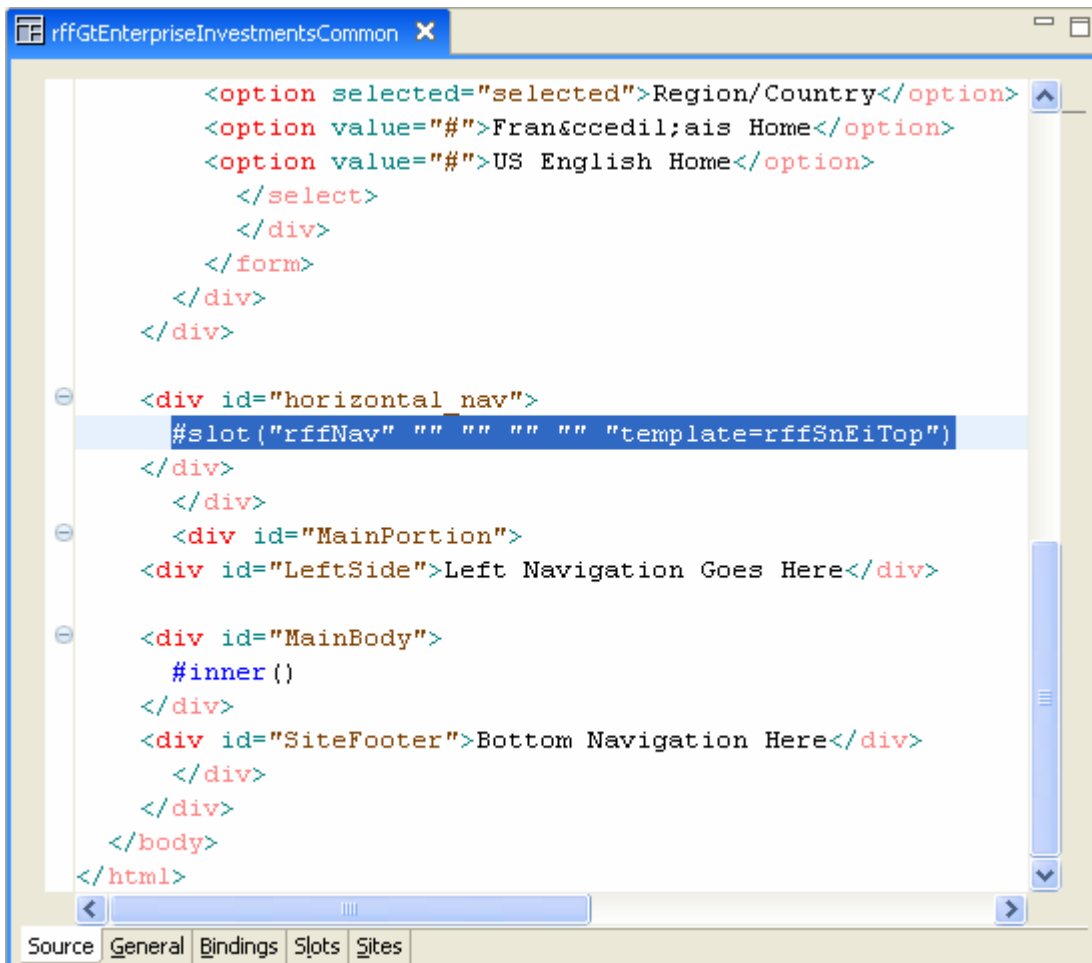
Managed Navigation is a Rhythmyx feature that allows you to create and maintain simple and effective navigation for your site automatically during publishing. The section *Managed Navigation* (see page 261) explains how to implement Managed Navigation in detail. For now, we only need to focus on how to add Managed Navigation to Global Templates.

Rhythmyx is shipped with a default Managed Navigation Slot. Adding this Slot to a Global Template differs little from adding a standard Slot to a Snippet or Page. Use the `#slot` macro to add the Slot. In Enterprise Investments, the `params` attribute is used to specify the Template used in each Slot, since different Templates are used for each Managed Navigation Slot. For example, the following Managed Navigation Templates were created for the Enterprise Investments Site in FastForward:

- `rffSnEINavTop` (provides Top Navigation for the Enterprise Investments Site)
- `rffSnEINavLeft` (provides Left Navigation for the Enterprise Investments Site)
- `rffSnEINavBottom` (provides Bottom Navigation for the Enterprise Investments Site)
- `rffSnEINavBreadcrumbs` (provides Breadcrumbs for the Enterprise Investments Site)
- `rffSnEISiteMap` (provides a Site Map for the Enterprise Investments Site)
- `rffSnNavPreload` (custom Template for FastForward)

For example, to add top navigation to the `rffEnterpriseInvestmentsCommon`, use the following code:

```
#slot("rffNav" "" "" "" "" "template=rffSnEiTop")
```



```
<option selected="selected">Region/Country</option>
<option value="#">Fran&ccedil;ais Home</option>
<option value="#">US English Home</option>
</select>
</div>
</form>
</div>
</div>
</div>
<div id="horizontal_nav">
  #slot("rffNav" "" "" "" "" "template=rffSnEiTop")
</div>
</div>
<div id="MainPortion">
<div id="LeftSide">Left Navigation Goes Here</div>

<div id="MainBody">
  #inner()
</div>
<div id="SiteFooter">Bottom Navigation Here</div>
</div>
</div>
</body>
</html>
```

Figure 124: Adding top navigation to the `rffGtEnterpriseInvestmentsCommon` Global Template



This code produces the following output:

Left Navigation Goes Here

[Enterprise Investments Home](#)

## 12 Easy Steps to preparing your estate plan

By Paul Baker

1. Prepare your Will. If you die without a Will, your estate ends up in probate court and your heirs' memories will not be as fond.
2. If you're 21 or older, make sure you not only have a Will, but also a durable power of attorney and a health care proxy.
3. Use estate planning software to make at least initial preparations. Most of the estate-planning software packages available today give you a good start on your estate plan. Completing the questions in the software program gives an attorney the necessary information and saves you time and billable dollars. If you create legal documents with a software package, make sure your attorney reviews them.
4. Get your Will notarized with the correct number of witnesses. Laws vary from state to state on this. No beneficiary should ever sign as a witness.
5. If you already have an estate plan, you should always review your plan in cases of divorce, death of a spouse, adoption, birth of each child, moving from one state to another, receiving a windfall, getting married or remarried.
6. Make a list of all of your assets and all of your liabilities. Your liabilities will have to be paid at your death. What is left over, minus administrative and probate costs, is what your beneficiaries will get. Decide who gets what, and in what proportion.
7. Name an executor who will manage your estate from the time of your death until the time that your assets are distributed. This is a big job, so make sure the person has the time and the ability to do it.
8. Choose a guardian for your children.
9. Have only one set of documents signed, witnessed and notarized. You will probably get duplicate copies. Keep the others for your files.
10. Review your estate plan every few years, even if your situation is pretty much the same. Laws change

*Figure 125: Preview of rffGtEnterpriseInvestmentsCommon Global Template with top navigation added*

Note that the page now includes a banner, and that a navigation bar is included immediately below the banner.

We can add the left (side) navigation and bottom navigation the same way.

```
#slot("rffNavSlot" "" "" "" "" "template=rffSnEINavLeft")
#slot("rffNavSlot" "" "" "" "" "template=rffSnEINavBottom")
```

## Converting References to Static Files

The header of the Enterprise Investments Global Template includes references to Cascading Stylesheet and JavaScript files:

```
<link rel="stylesheet"
href="..\web_resources\enterprise_investments\css\rxs_styles.css"
type="text/css" />
<script psx-
src="..\web_resources\enterprise_investments\js\mouseover.js"
language="javascript" type="text/javascript"></script>
```

Recall that the recommended cleanup of HTML files includes moving inline scripting and markup to supporting files. This code links to the supporting files containing this supporting code. The supporting files might not be in the same location in different output contexts, however. When previewing your pages in Rhythmyx, these files are in the following locations:

```
..\web_resources\enterprise_investments\css\rxs_styles.css
..\web_resources\enterprise_investments\js\mouseover.js
```

When the output is published, however, these files will likely be in a different location. For example, the defined locations for these files when the Enterprise Investments Site is published locally on the Rhythmyx server are:

```
\EIHome\resources\css\rxs_styles.css
\EIHome\resources\js\mouseover.js
```

To allow the flexibility to produce different paths to these files in different output contexts, Rhythmyx allows you to define a set of Context Variables that resolve to the different locations when an output is generated.

A Context Variable is a string that resolves to a particular value for each output context. When Rhythmyx is processing output for a specific context, it replaces the Context Variable with the value defined for that context.

### Defining Context Variables

In most cases, you should be able to use the standard Context Variable, `$ResourcePath`. The major reason you would create additional Context Variables is to support alternative output renderings for Managed Navigation.

To demonstrate the process of creating and using Context Variables, we will illustrate how the standard `ResourcePath` Context Variable was created for the Preview Output Context for the Enterprise Investments Site, and how to define additional values for this Context Variable.

To create the `ResourcePath` Context Variable:

- 1 Start a browser and log in to Content Explorer as a user with admin privileges.
- 2 Click on the Publishing tab.
- 3 In the left navigation, under Variables, click the **By Name** link.  
Content Explorer displays the Context Variables Editor.
- 4 Click the **New Variables** link.

Content Explorer displays the Edit Global Variables page.

- 5 In the Name field, enter `$ResourcePath`.
- 6 In the Value field, enter `../web_resources/enterprise_investments`.
- 7 In the Context drop list, choose `Preview`. (This is the default option.)
- 8 In the Site drop list, choose `Enterprise Investments`. Options in this list include all Sites defined in the system.

Edit Global Variables	
*Name	ResourcePath
*Value	../web_resources/enterprise_investments
Context	Preview
Site	Enterprise Investments
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 126: Defining the ResourcePath Context Variable

- 9 Click the [Save] button.  
Rhythmyx adds the new Context Variable.

Variables			
			New Variable
ResourcePath			Add Value
	Value	Site(id)	Context(id)
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
rxs_navbase			Add Value
	Value	Site(id)	Context(id)
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
✗	../web_resources/corporate_investments	Corporate Investments(303)	Preview(0)
✗	../web_resources	Preview Site(0)	Preview(0)
✗	/EI_Home/resources	Enterprise Investments(301)	Publish(1)

Figure 127: ResourcePath Context Variable defined

### Adding a Context Variable to the Global Template

We can now update the URL of the location of the static files in our Global Template with the Context Variable. To add the Context Variable, we must use the `$sys.variables` Binding Variable.

Thus the URL of the cascading stylesheet files

`..\web_resources\enterprise_investments\css\rxs_styles.css`  
becomes

`$sys.variables.ResourcePath\css\rxs_styles.css`

Thus, the header references become:

```
<link rel="stylesheet"
href="$sys.variables.ResourcePath\css\rxs_styles.css" type="text/css" />
<script psx-src="$sys.variables.ResourcePath\js\mouseover.js"
language="javascript" type="text/javascript"></script>
```

This produces a Preview that uses all of the correct Cascading Stylesheets and JavaScript files.

### **Adding a New Value for the Context Variable**

When creating the Context Variable, we defined a value for the variable for the Preview Context for the Enterprise Investments Site. We also need to define a value for the Publish Context for the Site. When publishing the Enterprise Investments Site locally to the Rhythmyx server, the output is published to `<Rhythmyxroot>\AppServer\server\rx\deploy\EI_Home.war`. This directory contains a resources subdirectory that contains the same static resources as the `..\web_resources\enterprise_investments` directory. All references to static files need to refer to this directory, so we need to add a new value to the ResourcePath Context Variable for the Publish output Context for the Enterprise Investments Site.

To add a new value to the ResourcePath Context Variable:

- 1 Start a browser and log in to Content Explorer as a user with admin privileges.
- 2 Click on the Publishing tab.
- 3 In the left navigation, under Variables, click the **By Name** link.  
Content Explorer displays the Context Variables Editor.  
Click the **Add Value** link next to Resource Path..  
Content Explorer displays the Edit Global Variables page.
- 4 In the **Value** field, enter `/EIHome/resources`.
- 5 In the **Context** drop list, choose *Publish*.
- 6 In the **Site** drop list, choose *Enterprise Investments*.
- 7 Click the [Save] button.

Rhythmyx adds the value to the Resource Path Context Variable

			Variables
			New Variable
ResourcePath			Add Value
	Value	Site(id)	Context(id)
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
✗	/EiHome/resources	Enterprise Investments(301)	Publish(1)
rxs_navbase			Add Value
	Value	Site(id)	Context(id)
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
✗	../web_resources/corporate_investments	Corporate Investments(303)	Preview(0)
✗	../web_resources	Preview Site(0)	Preview(0)
✗	/EI_Home/resources	Enterprise Investments(301)	Publish(1)

Figure 128: ResourcePath Context Variable with values for the Preview and Publishing Context of the Enterprise Investments Site

Use the same procedure to add values for the Corporate Investments Site. The Site needs values for both the Preview Context and the Publish Context.

## Implementing a Page Template Without a Global Template

Some individual Page Templates produce a look and feel that is different from any other page in the Site. Home pages are a typical example. For pages that have such a unique structure, there is no point to using a Global Template. You would have to use two Templates to produce the output when one Template would suffice.

The Enterprise Investments Home Page Template (rffPgEIHome) illustrates this technique.

Name: rffPgEIHome

Label: P-EI Home

Content Type: Home

Assembler: Velocity Assembler

Output: Page

Global Template: None

Publish: Default

Active Assembly Format: Normal

MIME Type: Text/HTML

Character Set: <null>

Location Prefix: <null>

Location Suffix: <null>

Bindings: None

Communities: Enterprise Investments

Contained Slots: rffHomeImage, rffHomeList, sys\_inline\_link

Sites: Enterprise Investments

Included Fields: Display Title. Body

We will assume that the HTML for this Snippet is stored in an HTML file named rffPgEIHome.html, which was created during the modeling and design process.

## Creating a Page Template Object Without a Global Template

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the rffPgEIHome PageTemplate object:

- 1 In the Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.  
The Rhythmyx Workbench displays the Type dialog of the Template wizard.
- 2 Choose the **Type-specific** radio button. In the **Content Type** field, choose *Home*. Click the [Next] button.

The Rhythmyx Workbench displays the Output format dialog of the Template wizard.

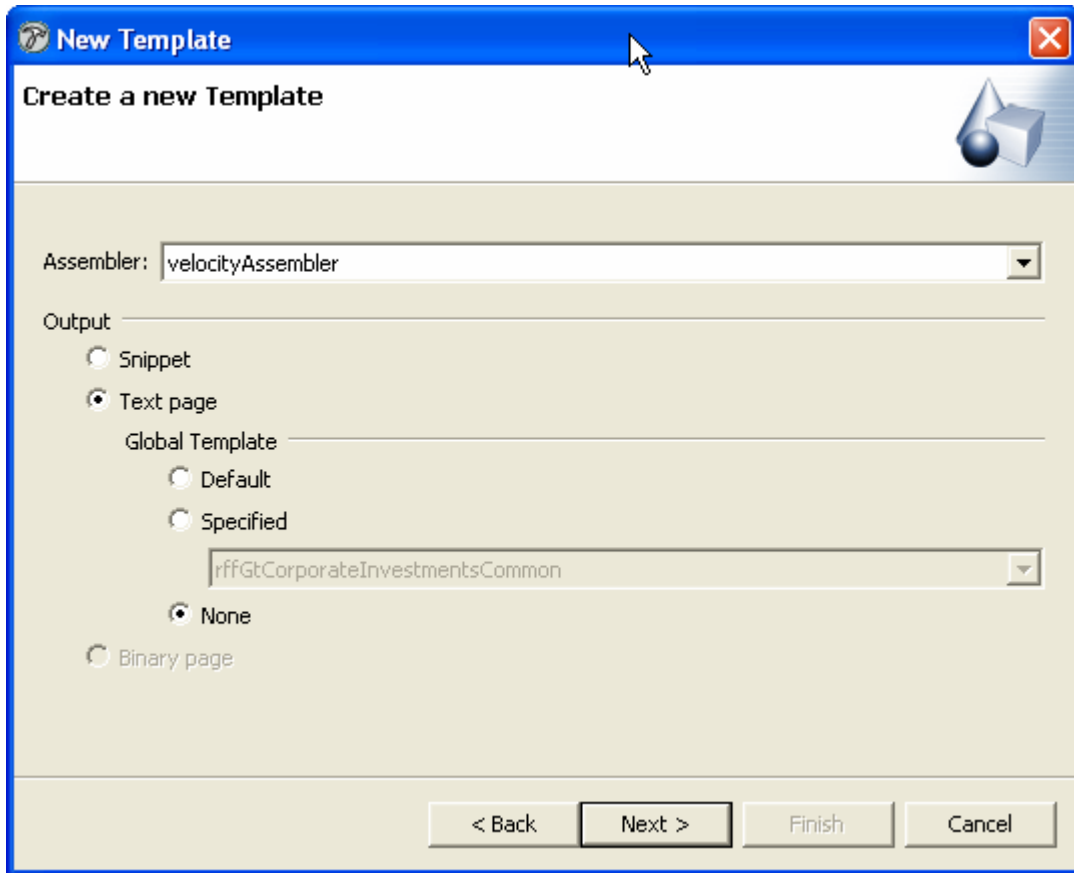


Figure 129: Output dialog for a Page Template with no Global Template specified

- 3 In the Assembler drop list, choose *Velocity Assembler* (this is the default option). In the Output section of the dialog, choose the **Page** radio button. Under Global Template, select **None**, which specifies that the Template will not use a Global Template. Click the [Next] button.

The Rhythmyx Workbench displays the General properties dialog of the Template wizard.

- 4 In the Template name field, enter *rffPgEIHome*. Modify the value in the Label field to *P - EI Home*.
- 5 In the Description field, enter *EI Home Pages*.
- 6 Click browse button next to the Source field, and use the browse dialog to find the file *rffPgEIHome.html*, and add it to the field.
- 7 In the Available Communities field, select *Enterprise Investments* and click [>] button to make this Template available to those Communities.
- 8 Click the [Next] button.

The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

- 9 Select the *rffHomeImage*, *rffHomeList*, and *sys\_inline\_link* Slots and the click [>] button to add them to the Template.

10 Click the [Finish] button.

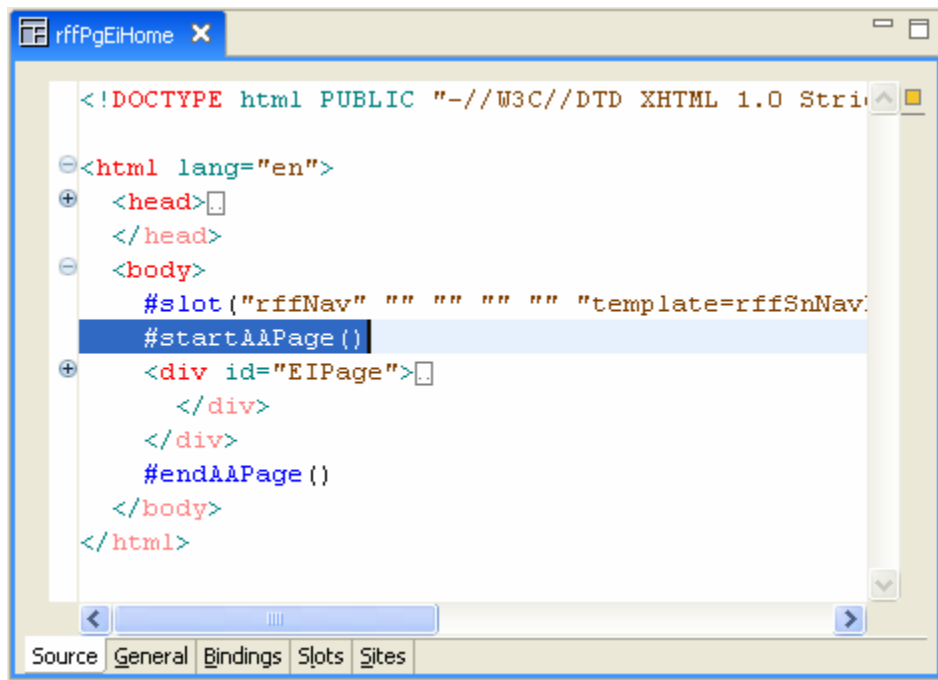
Rhythmyx creates the Template and displays the Template editor for the rffPgEIHome Template.

## Adding Velocity to the EIHome Page Template

To ensure that Active Assembly works correctly in a Page Template that does not use a Global Template, you must add the following markup:

- `#startAAPage` after the `<body>` tag in the Template and before any page content markup that you want to access in Active Assembly; and
- `#endAAPage` before the closing (`</body>`) tag after all page content markup that you want to access in Active Assembly.

So you would modify HTML markup of the rffEIHome Template as illustrated in the following screenshot:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Stri
<html lang="en">
  <head>
  </head>
  <body>
    #slot("rffNav" "" "" "" "" "template=rffSnNav
    #startAAPage()
    <div id="EIPage">
      </div>
    </div>
    #endAAPage()
  </body>
</html>
```

Figure 130: Page Template showing the `#startAAPage` and `#endAAPage` macros used when the Template does not have a Global Template

Note that the `#startAAPage` macro is highlighted in this screenshot.

If you open the rffEIHome Template in the Rhythmyx Workbench, you will notice that it uses embedded Velocity code. For details about using this code, see "Embedding Velocity Code in Templates" in the *Rhythmyx Technical Reference*.



## Dispatch Templates

A Dispatch Template is a Template that calculates a result to select the Template used to format an output. Dispatch Templates do not include any formatting themselves. The bindings of the Template are used to calculate the result.

The calculations are typically performed using the JEXL `if . . . else` function.

```
if (condition) {truevalue} else {falsevalue}
```

where

`condition` is a boolean condition you are testing

`truevalue` is the value used if the boolean expression evaluates to true

`falsevalue` is the value used if the boolean expression evaluates to false.

In the FastForward implementation, the `rffDsEIGenericSelector` Template illustrates the implementation of a Dispatch Template. This Template selects the correct Template to publish depending on whether a Content Item is specified as a Category Landing Page. If so, the `rffPgEIGenericCategoryPage` is published. Otherwise, the `rffPgEIGeneric` Template is published.

The Generic Content Type includes a field, `Usage`, that specifies whether the page is a landing page. The value of this field can be either *Landing Page* (the value "L" is stored in the Repository) or *Normal* (the value "N" is stored in the Repository).

The `rffDsEIGenericSelector` Template is only available to the Enterprise Investment Community and is only available on the Enterprise Investments Site.

## Creating the Dispatch Template Object

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the `rffDsEIGenericSelector` Template object:

- 1 In Menu bar of the Rhythmyx Workbench, choose *File > New > Template*.  
The Rhythmyx Workbench displays the Type dialog of the Template wizard.
- 2 Choose the **Shared** radio button and click the **[Next]** button.

The Rhythmyx Workbench displays the Output format dialog of the Template wizard.

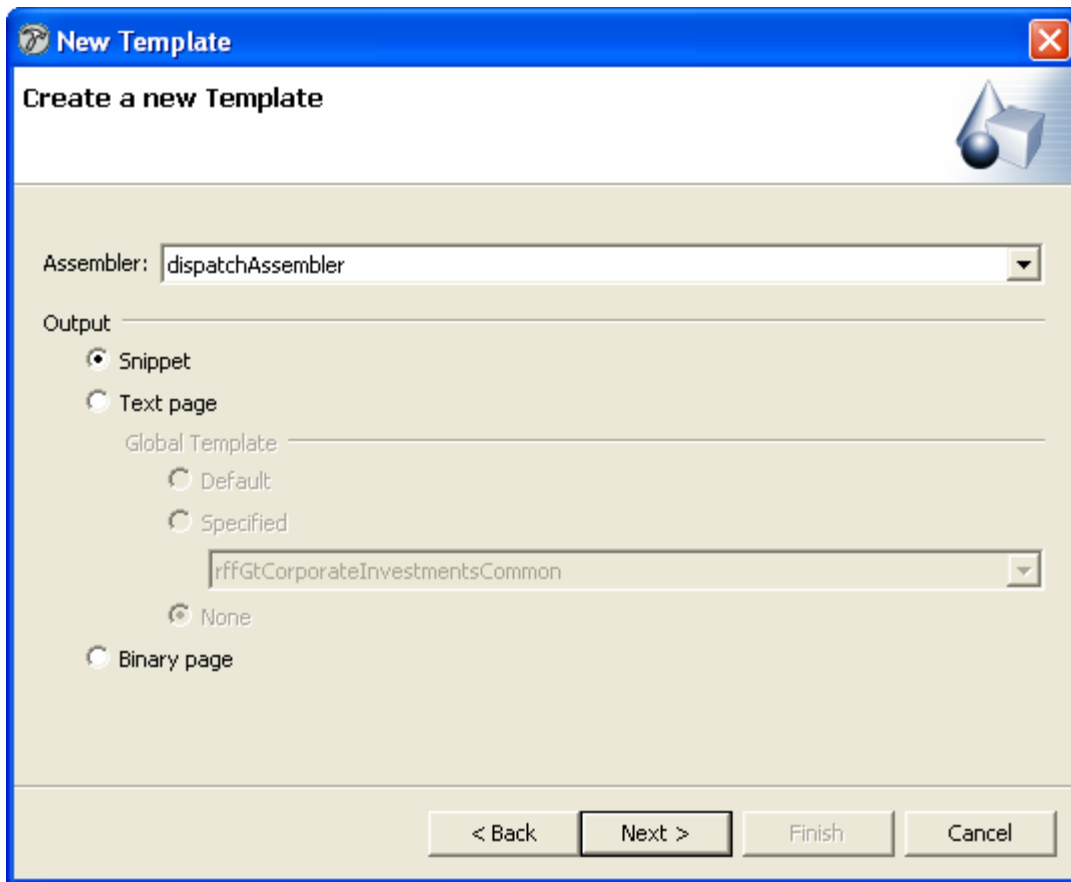


Figure 131: Specifying the Output properties of the Dispatch Template

- 3 In the Assembler drop list, choose *dispatch Assembler*. Click the [Next] button.

The Rhythmyx Workbench displays the General properties dialog of the Template wizard.

**New Template**

**Create a new Template**

Template name:

Label:

Description:

Source:

Available Communities:

- Corporate\_Investments
- Corporate\_Investments\_Admin
- Default
- Enterprise\_Investments\_Admin

Visible to these communities:

- Enterprise\_Investments

Filter:

< Back   Next >   Finish   Cancel

Figure 132: General Properties of the Dispatch Template

- 4 In the Template name field, enter *rffDsEIGenericSelector*. In the Label field, change the value to *D - EI Generic*.

- 5 In the **Description** field, enter *Dispatch to the appropriate page template for the given generic item*.
- 6 In the **Available Communities** field, select *Enterprise Investments*, then click the [**>**] button to make this Template available to the Enterprise Investment Community.
- 7 Dispatch Templates do not include any markup, so ignore the **Source** field. Click the [**Next**] button.

The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

- 8 Dispatch Templates cannot include Slots, so click the [**Next**] button.

The Rhythmyx Workbench displays the Content Types dialog of the Template wizard.

- 9 Move the Generic Content Type to the **Associated Content Types** field.
- 10 Click the [**Finish**] button.

Rhythmyx creates the Template and displays the Template editor for the rffDsEIGenericSelector Template.

## Defining the Dispatch Binding

Since we want to select a Template, we will bind the variable `$sys.template`.

The condition we want to test is the value of the usage field:

```
if usage=L, use rffPgEiGenericCategory, else use rffPgEIGeneric
```

We will need two bindings to to implement this selection. (NOTE: In the FastForward Implementation, the two bindings are combined into one script. Here, we separate the bindings for clarity.)

The first binding retrieves the the value of the usage field and assigns it to a variable; we will use `$usage`:

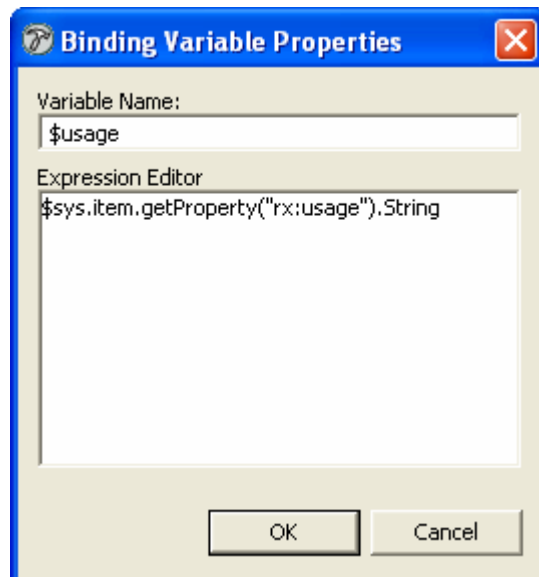


Figure 133: `$usage` binding for Dispatch Template

The second binding tests the value of \$usage to determine which Template to select.

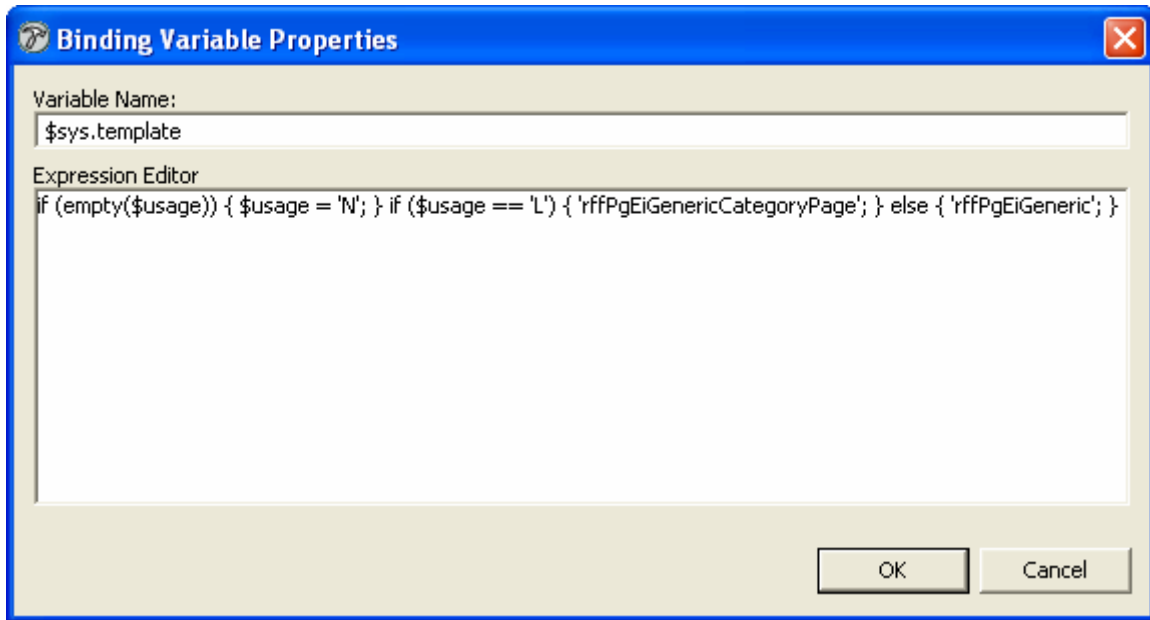


Figure 134: Condition Binding for Dispatch Template

To avoid an error in case the Usage field has a null value, the binding includes a script to assign a default value of "N" to \$usage)

The following screenshot illustrates the combined into one script:

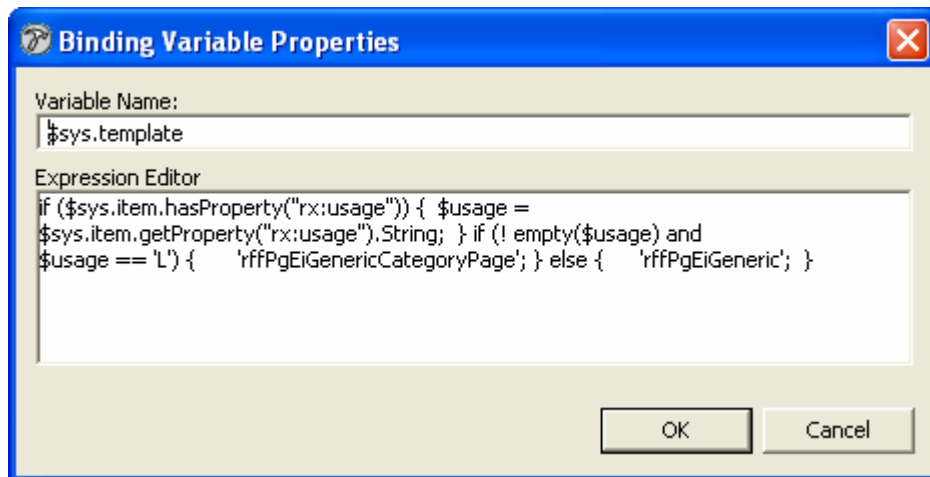


Figure 135: Dispatch binding as a script

Multiple conditions can be nested. For example, suppose a third option, "F" was available for the usage field; if the value of this field is "F", we want to use the rffPgEiGenericFund Page Template. The condition we want to test is:

```
if usage=L, use rffPgEiGenericCategory,
if usage=F, use rffPgEiGenericFund
else use rffPgEiGeneric
```

The binding expression would be:

```
if ($usage == 'L') {'rffPgEiGenericCategoryPage'; } else {if
($usage=='F') {'rffPgEiGenericFund';} else { 'rffPgEiGeneric';}; }
```

## Creating an Automated Slot

In some cases, you may want to generate a list of Content Items for a Slot automatically rather than requiring Content Contributors to assign related Content Items to the Slot manually. You may want to use this practice if the criteria for including Content Items in the Slot are fixed and easy to define and automate. For example, if you want to select all the Press Release Content Items created in a specific year, you can define an expression that would select Content Items where the value of the Created Date Field is in that year. Sometimes automation may be the only way to achieve the desired result. For example, if you want to select the last five Press Releases to go Public, it is unlikely that you can find a practical method that allows a Content Contributor to update the list, but you can easily define a query that selects the required Press Release Content Items.

The difference between an Automated Slot and a Standard Slot is that the Content Finder specified for an Automated Slot is the `sys_AutoSlotContentFinder`. One of the required parameters of this Content Finder is the query parameter, which specifies the query used to select the Content Items added to the Slot.

## Creating a Simple Automated Slot

The `rffAutoPressReleases2005` Slot in FastForward is a Simple example of an Automated Slot. This Slot has the following characteristics:

Slot Name	Description	Allowed Relationship Type	Content Finder
<code>rffSnPressReleases2005</code>	Lists all Press Release Content Items created during 2005	Active Assembly	<code>sys_AutoSlotContentFinder</code>

The Allowed content for the Slot is defined as:

Content Type	Template
Press Release	<code>rffSnDateAndTitleLink</code>
Press Release	<code>rffSnTitleLinkBullet</code>

In pseudocode, the query for this slot resembles the following:

```
select Press Release Content Items from the current Site where the
sys_contentcreatedate=2005 and order them by start date
```

The query is written in JSR-170 query language (for additional details, see *Writing Automated Slot Queries* (see page 190) ) which does not include a date function or an IN operator. We can circumvent this problem by specifying that the Content Start Date falls before January 1, 2006 and after December 31, 2004.

```
SELECT rx:sys_contentid, rx:sys_contentstartdate FROM rx:rffpressrelease
WHERE rx:sys_contentstartdate < '2006/1/1' AND rx:sys_contentstartdate >
'2004/12/31'
```

Specifying the current Site and the ordering results in the following query:

So the final query would resemble the following:

```
SELECT rx:sys_contentid, rx:sys_contentstartdate FROM rx:rffpressrelease
WHERE rx:sys_contentstartdate < '2006/1/1' AND rx:sys_contentstartdate >
'2004/12/31' AND jcr:path like :site_path ORDER BY
rx:sys_contentstartdate
```

NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Press Releases 2005 Auto Slot:

- 1 In the Rhythmyx Workbench, from the Menu bar, choose *File > New > Slot*.  
The Rhythmyx Workbench displays the New Slot wizard.
- 2 In the Slot name field, enter *rffAutoPressReleases2005*. This value is also entered in the Label field. Change the value in the Label field to *All Press Releases 2005*.
- 3 In the Description field, enter *All press releases with a start date in 2005*.
- 4 In the Content finder drop list, choose *sys\_AutoSlotContentFinder*.

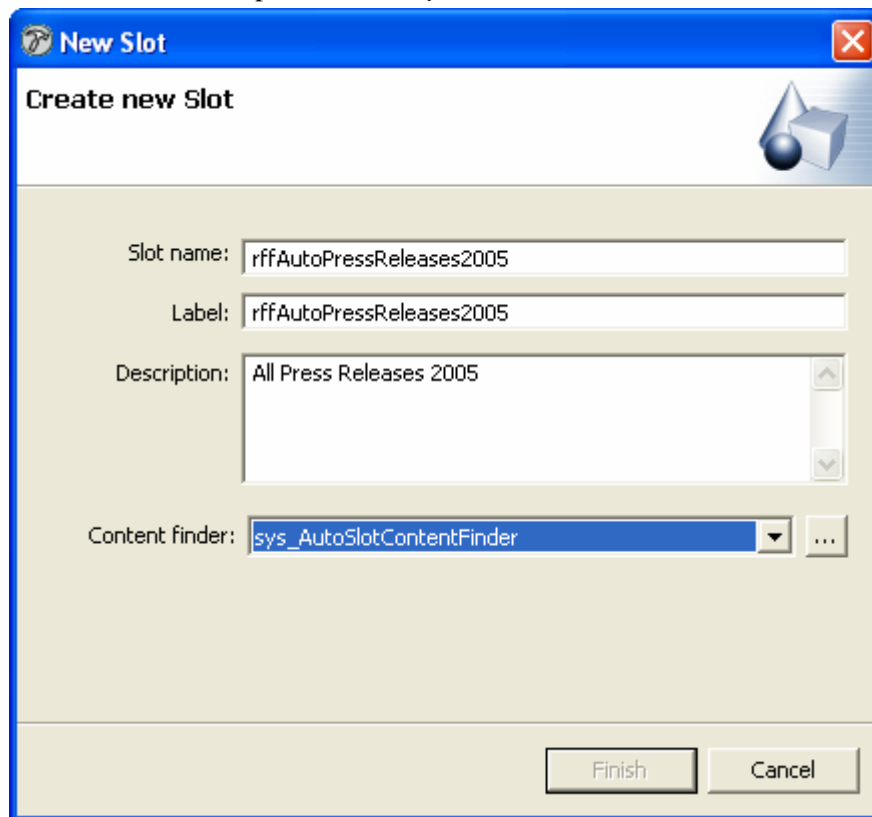


Figure 136: Creating the *rffAutoPressReleases2005* Slot

This Content Finder defines the list of Content Items for the Slot automatically. The criteria for selecting the Content Items are defined in the parameters of the *sys\_AutoSlotContentFinder* extension. To specify the criteria for selection Content Items:

- a) Click the browse button  to display the Extension Parameters dialog.



b) Enter the following values for the parameters of the extension:

Parameter	Value
query	SELECT rx:sys_contentid, rx:sys_contentstartdate FROM rx:rffpressrelease WHERE rx:sys_contentstartdate < '2006/1/1' AND rx:sys_contentstartdate > '2004/12/31' AND jcr:path like :site_path ORDER BY rx:sys_contentstartdate
type	sql (can leave unspecified; if unspecified, defaults to sql)
template	rffSnDateAndTitleLink
maxresults	(Leave null)

c) When you finish entering values for the parameters, click the [OK] button to save your edits.

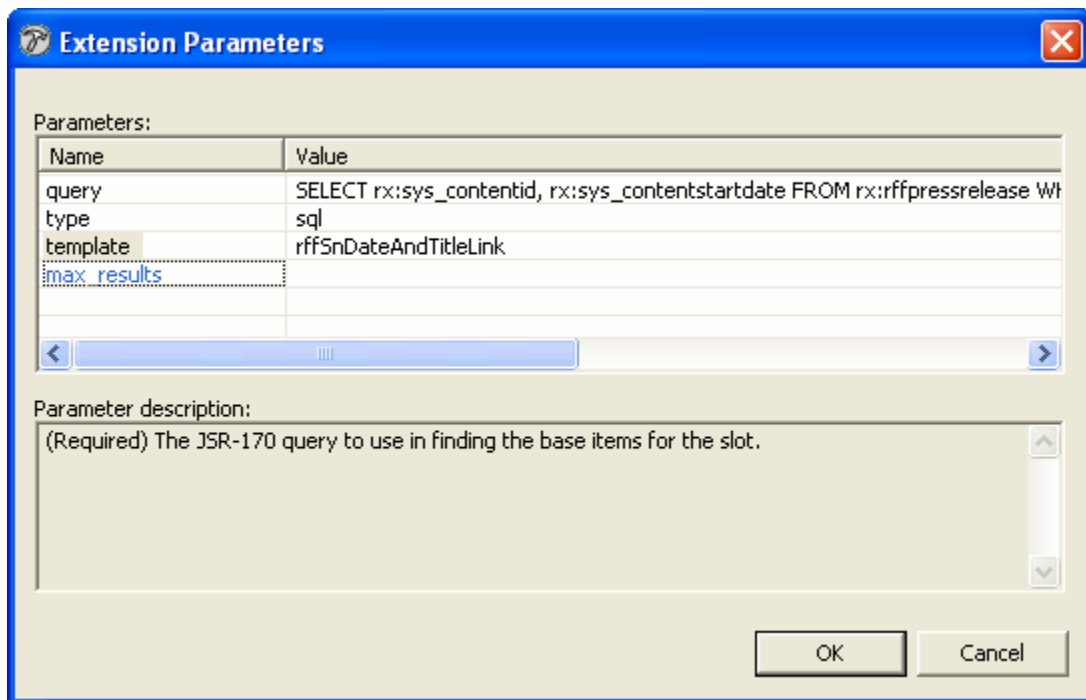


Figure 137: *rffAutoPressReleases2005 Query*

5 Click the [Finish] button.

- 6 Rhythmyx saves the Slot and displays it in the Slot editor.

The screenshot shows the Rhythmyx Slot Editor for a slot named 'rffAutoPressReleases2005'. The interface includes the following fields and controls:

- Slot name:** rffAutoPressReleases2005
- Label:** All Press Releases 2005
- Description:** All press releases with a start date in 2005
- Type:** Radio buttons for 'Regular' (selected) and 'Inline'.
- Content finder:** A dropdown menu showing 'sys\_AutoSlotContentFinder' with a search button.
- Allowed relationship types:** A dropdown menu showing 'ActiveAssembly'.
- Allowed content:** A table with two columns: 'Content Type' and 'Template'.
 

Content Type	Template
rffPressRelease	rffSnDateAndTitleLink

Figure 138: rffAutoPressReleases2005 Editor

- 7 The Slot Type will remain **Regular** and the **Allowed relationship type** will remain *ActiveAssembly*.
- 8 The Content Types and Templates are specified in the parameters of the `sys_AutoSlotContentFinder` extension, so we can leave the Allowed content table empty or specify the Press Release Content Type and the `rffSnDateAndTitleLink` Template as illustrated..
- 9 In the Button bar of the Rhythmyx Workbench, click the save button.

## Writing Automated Slot Queries

The value of the query parameter of the `sys_AutoSlotContentFinder` written using JSR-170 Query Language. JSR-170 Query Language is a language similar to Structured Query Language [SQL] used to query content Repositories.

---

NOTE: For additional details, see the JSR-170 spec at <http://www.jcp.org/en/jsr/detail?id=170>.

---

If you are familiar with the use of Structured Query Language (SQL) to interact with relational databases, the format of a JSR-170 Query Language query will look familiar:

```
select rx:sys_contentid(,rx:sys_revisionid) from rx:contenttype
[,rx:contenttype...] where conditional expression order by fieldname
```

Note that all Rhythmyx elements in the query must be prefixed by the string `rx:`. If you do not prefix a Rhythmyx element with this string, the output of the query will generate errors. Also, all Rhythmyx elements should be formatted in lowercase (for example, to select Content Items of the Press Release Content Type, you would specify `from rx:press_release`). Note that spaces are not valid and should be replaced by underscores.

The `select` clause in the expression must include the fields `rx:sys_contentid` and `rx:sys_revision`. All data for the Content Items is returned. The specific fields used are defined by the Template used to render the output.

The `from` clause specifies one or more Content Types for which to return data. Each Content Type specified must be prefixed with the `rx:` string. Use commas to separate Content Types. To return all Content Types, specify `nt:base`.

The `where` clause specifies the conditions used to select specific Content Items. The following operators can be used:

- `<` (less than)
- `>` (greater than)
- `=` (equals)
- `<=` (less than or equal to)
- `>=` (greater than or equal to)
- `<>` (does not equal)
- `LIKE`

Multiple conditions can be specified using the following operators (in order of precedence):

- `NOT`
- `AND`
- `OR`

If a condition clause includes more than two conditions, use parentheses to group conditions. Parentheses override the usual precedence order.

The `LIKE` operator matches the pattern string specified with the operator. The pattern string must be enclosed in single quotation marks and can use the wildcards “`%`” (matches 0 or more characters) and “`_`” (matches one character). When using the `LIKE` operator, use `jcr:path` to return Folder paths. For example, the statement

```
select rx:sys_contentid, rx:sys_revisionid from rx:generic where
jcr:path like '//Sites/EnterpriseInvestments/Invest%'
```

returns all Content Items of the Generic Content Type that have a path that starts with `//Sites/EnterpriseInvestments/Invest`, such as `//Sites/EnterpriseInvestments/InvestmentAdvice` and `//Sites/EnterpriseInvestments/InvestmentPlans`.

Use the `order by` clause to specify the order of the returned results, specifying the field to use when determining the order. For example, to order by Content Creation Date, you would add the clause

```
order by rx:sys_contentcreatedate
```

To return the first or last of a set of Content Items, combine the order by clause with the `maxresult` parameter of the `sys_AutoSlotContentFinder`. For example, if you wanted to publish the last five Press Release Content Items to go public, you might add the following order clause to your query:

```
Order by rx:sys_startdate
```

Then specify `maxresults = 5`.

## Automated Slots with Variable Parameters

In many cases, when defining an Automated Slot, you will want to query Slot Contents based on variable data rather than based on constants. The variables must be defined in the bindings of the Template that calls the Slot.

Variable parameters are used in the `where` clause of the Automated Slot select query. Variable parameters are formatted with a colon before the name of the parameter:

```
:variablename
```

For example, suppose we wanted a richer Funds section of the Site, with subsections categorizing funds in different ways (by type, such as REITs, Index Funds, and so forth; by fund size; by date established ). To implement this behavior, we would need two Content Types:

- A Funds Content Type that includes fields for the various categorizations we want. For the purposes of this exercise, we will assume that this Content Type contains the following fields:
  - Fund Type has the following options: REIT, S&P 500 Index, High Income, High Growth
  - Status has the options Open and Closed.
- A Funds Category Content Type that would render the index of Funds Content Items of each combination of Fund Type and Status (in other words, REIT Open, S&P 500 Index Closed, and so forth. This Content Type shares the Fund Type field with the Funds Content Type; the same set of values will be available for the field in both Content Types.

## Setting Up Bindings for an Automated Slot

The variables for an Automated Slot are defined in the Bindings of the Template that calls the Slot. You must use a compound variable to define the variables for the Automated slot query. The "parent" variable is added to the Slot definition in the Velocity markup. The "child" variables are used in the Automated Slot query.

For example, to implement the behavior we want for our Funds section, we need two variables. We will call the "parent" variable `$fundselector`. The child variables are

- `$fundselector.ftype` is used to pass the value of the Fund Type field to the Automated Slot query. The binding for this variable is  

```
$fundselector.ftype=$sys.item.getProperty("fund_type").string
```
- `fstatus` is used to pass the value of the Status field to the Automated Slot query. To ensure that only open funds are selected, we will set the value of the `theStatus` variable to `open`:  

```
$fundselect.fstatus="open"
```

When adding the Automated Slot to the Page Template, in the parameters parameter of the `#slot` macro, specify the `$fundselector` parameter.

```
#slot ("rffFundsAutomatedList", "", "", "", "", "", $fundselector)
```

This call passes the `ftype` and `fstatus` variables to the Automated Slot.

## Adding Variables to an Automated Slot Query

When defining the where clause of an Automated Content query, compare the value of the Content Type fields to the value of the variable:

```
rx:contenttypefield=:variablename
```

In our example, we want to select Funds Content Items where the value of the Fund Type field on the Fund Content Item matches the value of the Funder Type field on the Funds Category Content Item and where the value of the Status field of the Fund Content Item is *open*. The query would resemble the following code:

```
select rx:sys_contentid,rx:sys_revisionid from rx:funds where  
rx:fund_type=:ftype and rx:fund_status=:fstatus
```

## Troubleshooting Templates

When developing Templates, you may encounter one of the following common errors. This section describes these common errors, how to diagnose the cause of the error, and how to resolve it.

### Property Not Found Error

When previewing a Template, an error page is returned with the “Error reported” stating “property: <name> not found”:

#### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	2
sys_siteid	301
sys_itemfilter	preview
sys_template	rffSnNameAndAddress
sys_contentid	504
sys_folderid	
sys_context	0

#### Error reported

Problem assembling output for item: 2-101-504 with template: rffSnNameAndAddress exception: Property rx:firstnme not found see log for stack trace

*Please note: More information may be available on the console*

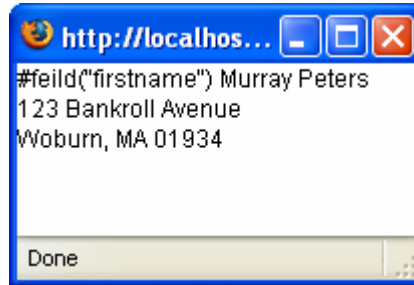
*Figure 139: Error page showing "property not found"*

This error indicates that the Content Item field was incorrect spelled (“firstnme”, which probably should have been “firstname”).

To resolve this problem, open the Template and correct the spelling of the field. To find the correct spelling of the field, open the Content Type associated with the Template and find the field you intended the add. In the Rhythmyx Workbench, you can display the Template Editor and the Content Type Editor side-by-side, as illustrated in the screenshot below, making it easy to find the field you need. To display the editors side-by-side, select one of the editors and drag it to the bar between the navigation view and the other editor.

## Macro Rendered as Plain Text

When previewing a Template, you may see a Velocity macro rendered as plain text, as in the following screenshot:



*Figure 140: Assembled Content Item showing macro rendered as plain text*

This output indicates that the macro was specified incorrectly. The following errors in specifying macros may occur:

- The macro was misspelled (as in the example)
- The macro was specified using one or more characters of the wrong case (in other words, an upper-case letter where a lower-case letter should have been used, or a lower-case letter where an upper-case letter should have been used).
- The macro was specified without the “#” character before the macro name.

You can usually determine the error in specifying the macro by looking at the output. To address this error, specify the macro correctly:

- Ensure that you included the “#” character before the macro.
- Ensure that all characters use the correct case.
- Ensure that the macro is spelled correctly.

To confirm the spelling and formatting of macros, check the .vm files where the macros are defined.

Macros shipped by Percussion Software are defined in the file

<Rhythmyxroot>/sys\_resources/vm/sys\_assembly.vm. Custom macros should be defined in the file <Rhythmyxroot>/rx\_resouces/vm/rx\_assembly.vm. (NOTE: Custom macros should only be defined in the file <Rhythmyxroot>/rx\_resouces/vm/rx\_assembly.vm. The file <Rhythmyxroot>/sys\_resources/vm/sys\_assembly.vm is overwritten during upgrade and any modifications to it will be lost.)

## Invalid Argument

When previewing a Template, an error page is returned with the “Error reported” stating “Invalid argument #<n> in VM #macroname”:

### Problem during the assembly of item

[Click here to view velocity log](#)

### Parameters passed

Name	Value
sys_revision	2
sys_siteid	301
sys_template	rffSnNameAndAddress
sys_itemfilter	preview
sys_contentid	504
sys_folderid	
sys_context	0

### Error reported

Problem assembling output for item: 2-101-504 with template: rffSnNameAndAddress exception: Invalid arg #0 in VM #field at line 11, column 7 in template rffSnNameAndAddress see log for stack trace

*Figure 141: Error page showing "invalid argument" error*

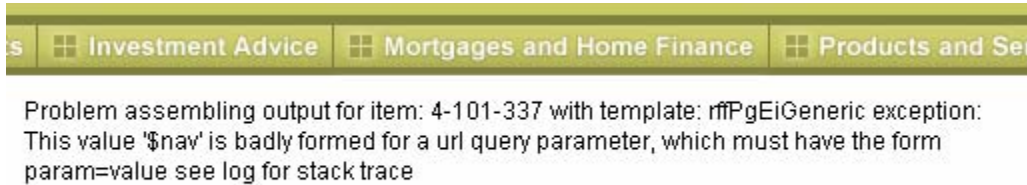
This error typically means that at least one parameter of the macro that requires a literal value was specified without quotation marks. All literal values must be specified with quotation marks (best practice is to use double quotation marks), while objects must be specified without quotation marks. In general, it is safe to assume that anything that begins with the character “\$” is an object and must not be encased in quotation marks. Any other value is a literal value that must be encased in quotation marks.

Review all instances of the specified macro in the Template and ensure that all literal value arguments are encased in quotation marks.



## Problem Assembling Output: Value is Badly Formed

When previewing a Template, an unformatted or partially formatted page is returned with an error message stating that there was a problem assembling output for a Content Item, and that “This value <name> is badly formed for a url parameter.



*Figure 142: Partially assembled page showing incorrectly formatting object Template macro*

This error indicates that the macro parameter, which is being specified as an object, was defined with quotation marks. Objects must be specified without quotation marks, while literal values must be specified with quotation marks. In general, it is safe to assume that anything that begins with the character “\$” is an object and must not be encased in quotation marks. Any other value is a literal value that must be encased in quotation marks.

Review all instances of the specified macro in the Template and ensure that all object value arguments are not encased in quotation marks.

## Parameter Not Defined

When previewing a Template, an error page is returned with the “Error reported” stating "parameter not defined":

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	3
sys_siteid	301
sys_itemfilter	preview
sys_template	TestDates
sys_contentid	497
sys_folderid	509
sys_context	0

#### Error reported

Problem assembling output for item: 3-101-497 with template:  
TestDates exception: parameter todaysdate not defined see log for  
stack trace

*Figure 143: Error page showing "parameter not defined error"*

This error typically occurs when you have defined a compound variable (such as `$circle.diameter` and `$circle.radius`) and have specified the "root" variable (in this example, `$circle`) with quotation marks (for example, `#slot ("template" "" "" "" "" "$circle")`). The leaf variable you were using would be reported as not defined (so in this case, if we were using `$circle.radius`, the message would read that "parameter radius not defined").

Binding variables are objects and should be specified without quotation marks. In general, it is safe to assume that anything that begins with the character "\$" is an object and must not be encased in quotation marks. Any other value is a literal value that must be encased in quotation marks.

To resolve this problem, check the bindings for the leaf variable reported in the error message and note the root variable. On the Source tab, find the macro where the root variable is defined and remove the quotation marks from it.

## Lexical Error

When previewing a Template, an error page is returned with the “Error reported” stating “Lexical error”:

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	2
sys_siteid	301
sys_template	rffSnNameAndAddress
sys_itemfilter	preview
sys_contentid	504
sys_folderid	
sys_context	0

#### Error reported

```
Problem assembling output for item: 2-101-504
with template: rffSnNameAndAddress exception:
Lexical error:
org.apache.velocity.runtime.parser.TokenMgrError:
Lexical error at line 11, column 102. Encountered:
"\n" (10), after : "\'firstname\')#field_if_set(\' \'
\'\'lastname\''\'")#field_if_set(\' \'
\'\'lastname\''\'")" see log for stack trace
```

*Figure 144: Error page showing "lexical error"*

The log returns a result similar to the following:

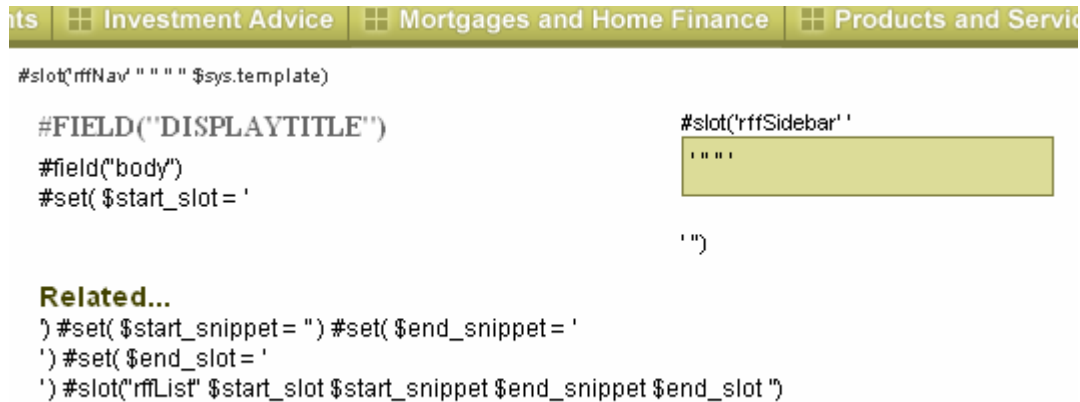
```
2006-08-29 13:35:27,901 - Method getProperty threw exception
for reference $sys in template rffSnNameAndAddress at [1,29]
2006-08-29 13:58:35,352 - Parser Error: #macro() :
rffSnNameAndAddress :
org.apache.velocity.runtime.directive.MacroParseException:
Invalid arg #0 in VM #field at line 11, column 7 in template
rffSnNameAndAddress
```

*Figure 145: Velocity log showing output for a lexical error*

Typically, this error indicates that the macros in the Template have been specified with a mix of single quotation marks and double-quotation marks. In general, best practice is to use double quotation marks for the parameters of all macros.

## Velocity Code in Output

When previewing a Template, the assembled output includes Velocity code:



```

#slot('rffNav' " " " $sys.template)

#FIELD("DISPLAYTITLE")
#field("body")
#set( $start_slot = '

#slot('rffSidebar' '
....

')

Related...
) #set( $start_snippet = " ) #set( $end_snippet = '
') #set( $end_slot = '
') #slot("rffList" $start_slot $start_snippet $end_snippet $end_slot ")

```

*Figure 146: Assembled Template showing Velocity code in output*

This output occurs if you have specified `$sys.template` as the value of a Template parameter in a macro. The value of a Template parameter of a macro should be either the name of a Template or a binding that resolves to the name of a Template. The system binding `$sys.template` should not be used as the value of a macro parameter.

## Illegal Argument Exception: Target Template May Not be Null

When previewing a Snippet Template that includes a link, an error page is returned with the “Error reported” stating “java.lang.IllegalArgumentException: targetTemplate may not be null.”:

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	4
sys_siteid	301
sys_itemfilter	preview
sys_template	rffSnTitleCalloutLink
sys_contentid	337
sys_folderid	310
sys_context	0

#### Error reported

```
Unexpected exception while assembling one or more items:
java.lang.IllegalArgumentException: targetTemplate may not be
null
```

*Please note: More information may be available on the console*

*Figure 147: Error page showing "target Template may not be null"*

This error indicates that you have included the `$rx.location.generate` function with the `targetTemplate` parameter, but have specified the Template incorrectly, usually by misspelling the name. Correct the name of the Template in the binding. To find the correct name, use the Assembly View.

## Problems Assembling Binary Outputs

When previewing a binary Content Item, such as an image file or a .pdf file, an error is reported:

- When previewing an image file, an error such as “The image <URL> cannot be displayed because it contains errors.”
- When Previewing a .pdf file, Acrobat Reader displays an error stating that "The files does not begin with '%pdf-'.
- When previewing a Microsoft Word document, the system offers to open a document called “/render”. When opened, the document only contains the text “assembly/render”.

These outputs indicate that you have specified an invalid data type for the `$sys.binary` binding in the Binary Template. The value of the `$sys.binary` binding must be a binary value. A common error is specifying the wrong field, such as the `sys_title` field, which returns a string. Correct the value of the binding to the name of a binary field.

## Could Not Find Method <Name> for Object [null]

When previewing a Template, an error page is returned with the “Error reported” stating “Could not find method <name> for object [null]”:

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	4
sys_siteid	301
sys_itemfilter	preview
sys_template	rffSnTitleCalloutAndMoreLink
sys_contentid	337
sys_folderid	310
sys_context	0

#### Error reported

```
Unexpected exception while assembling one or more items:
java.lang.RuntimeException: Could not find method generate for object [null] and
arguments [com.percussion.services.assembly.data.PSAssemblyWorkItem@c671f1[
m_id=4-101-337 m_mimeType= m_isDebug=false m_isPublish=true m_resultData=
m_status=
m_path=//Sites/EnterpriseInvestments/InvestmentAdvice/EstatePlanning/EI 12 Easy
Steps to preparing your estate plan
m_parameters={ sys_siteid=[Ljava.lang.String;@e926a0,
sys_revision=[Ljava.lang.String;@1adcafb,
sys_template=[Ljava.lang.String;@e6d5eb, sys_folderid=[Ljava.lang.String;@5b4219,
sys_contentid=[Ljava.lang.String;@12e0542,
sys_context=[Ljava.lang.String;@15088ee,
sys_itemfilter=[Ljava.lang.String;@e6526f} m_variables=
m_template=com.percussion.services.assembly.data.PSAssemblyTemplate@1e2d332[
id=537 version=18 name=rffSnTitleCalloutAndMoreLink label=S - Title Callout and
More Link locationPrefix= locationSuffix=
assembler=Java/global/percussion/assembly/velocityAssembler
assemblyUrl=./assembler/render styleSheet= aaType=0 outputFormat=2
publishWhen=n templateType=0 description=Renders an Image, Title, callout and a
more Link template=
```

```
#slot_simple("rffImageLink")
```

```
#field("displaytitle")
```

```
#field("callout") more >>
```

Figure 148: Error page showing "could not finde method <name> for object [null]"

This error usually indicates that the name of a binding function has been specified incorrectly. The method has generally been specified correctly. To address this problem, review your bindings and find the ones that use the specified method, and correct the spelling of the function. If you are using a binding function shipped by Percussion Software, check the Binding Variables section of the Workbench Help or the Javadoc for the correct spellings. If you are using a custom binding function, check your code or your own Javadoc.



## Java.lang.RuntimeException: Could not find method <name> for object <bindingfunction>

When previewing a Template, an error page is returned with the “Error reported” stating “java.lang.RuntimeException: Could not find method <name> for object [bindingfunctionclass]”:

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	4
sys_siteid	301
sys_itemfilter	preview
sys_template	rffSnTitleCalloutAndMoreLink
sys_contentid	337
sys_folderid	310
sys_context	0

#### Error reported

```
Unexpected exception while assembling one or more items:
java.lang.RuntimeException: Could not find method generate for object
[com.percussion.services.assembly.jexl.PSLocationUtils@14bb38f] and arguments
[com.percussion.services.assembly.data.PSAssemblyWorkItem@f98b98[
m_id=4-101-337 m_mimeType= m_isDebug=false m_isPublish=true m_resultData=
m_status=
m_path=//Sites/EnterpriseInvestments/InvestmentAdvice/EstatePlanning/EI 12 Easy
Steps to preparing your estate plan
m_parameters={ sys_siteid=[Ljava.lang.String;@1caaaa61,
sys_itemfilter=[Ljava.lang.String;@17f8b87, sys_context=[Ljava.lang.String;@29d4f6,
sys_revision=[Ljava.lang.String;@13bc040,
sys_folderid=[Ljava.lang.String;@1ad9946,
sys_contentid=[Ljava.lang.String;@a15c38,
sys_template=[Ljava.lang.String;@7804be} m_variables=
m_template=com.percussion.services.assembly.data.PSAssemblyTemplate@13f01f0[
id=537 version=20 name=rffSnTitleCalloutAndMoreLink label=S - Title Callout and
More Link locationPrefix= locationSuffix=
assembler=Java/global/percussion/assembly/velocityAssembler
assemblyUrl=./assembler/render styleSheet= aaType=0 outputFormat=2
publishWhen=n templateType=0 description=Renders an Image, Title, callout and a
more Link template=

#slot_simple("rffImageLink")

#field("displaytitle")

#field("callout") more >>
```

Figure 149: Error page showing “Java.lang.RuntimeException: Could not find method <name> for object <bindingfunction>”

This message may occur for two reasons:

- The name of the specific binding function method was specified incorrectly. The class of the binding function is listed, and the incorrect method name is also indicated. To resolve this problem, review your bindings for the ones that use the specified function. The method is typically misspelled, so you can probably determine which method you intended based on the information in the error message.
- The binding function was specified with the wrong number of parameters (either too many or too few). The binding function and the parameters passed are listed. Look up the Javadoc for the function to determine the correct number of parameters to pass to the function and correct the specification of the function in the bindings.

You may also see the `Java.lang.runtimeexception` specifying that the method does not exist for object `[null]`. This error indicates that you have specified both the function and the method incorrectly. Isolate the incorrect method first; the incorrect function is in the same binding.

## Problem Parsing Expression

When previewing a Template, an error page is returned with the “error reported” stating “Problem parsing expression” `<function>`”.

### Problem during the assembly of item

[Click here to view velocity log](#)

#### Parameters passed

Name	Value
sys_revision	3
sys_siteid	301
sys_itemfilter	preview
sys_template	rffPgCalendarMonth
sys_contentid	497
sys_folderid	
sys_context	0

#### Error reported

```
Unexpected exception while assembling one or more items:  
java.lang.Exception: Problem parsing expression:  
$rx.cond.choose($sys.site.path != null, $sys.site.path,) + "%" at  
character 55
```

*Figure 150: Error page showing "problems parsing expression"*

This error indicates that you have specified the parameters of the function incorrectly. A common error is incorrect separators between parameters. Parameters should be separated by commas with no spaces. Spaces, dots, or other separators will result in a parsing error. Another common error is including a stray comma after the last parameter.

## Java.lang.NullPointerException

When previewing a Template, an error page is returned with the “Error reported” stating “java.lang.NullPointerException”

### Problem during the assembly of item

[Click here to view velocity log](#)

### Parameters passed

Name	Value
sys_revision	4
sys_siteid	301
sys_template	rffSnTitleCalloutAndMoreLink
sys_itemfilter	preview
sys_contentid	337
sys_folderid	310
sys_context	0

### Error reported

Unexpected exception while assembling one or more items:  
java.lang.NullPointerException

*Figure 151: Error page showing "null pointer exception"*

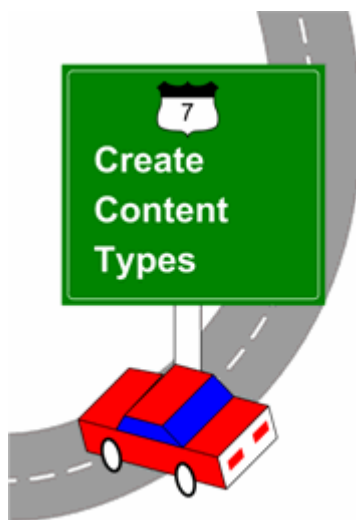
Null pointer exceptions occur whenever a null value is passed to the Assembly engine. Null values could occur for a variety of reasons. Common causes of null pointer exceptions include:

- Specifying a non-existent object as the value of a binding;
- Specifying a null as the value of a binding function parameter where nulls are not valid;

The error message does not give any details regarding the cause of the null pointer exception. To debug, carefully examine all binding functions and macros to assess which is causing the exception.



# Creating Content Types



A Content Type defines a specific group of Content Items. A Content Type's definition consists of the fields that make up the Content Type and their properties, and the Workflows and Communities associated with the Content Type. The Content Type's definition also includes any validation, transform, and pre- and post-processing extensions assigned to it. A Content Type can include local fields that are specific to its definition as well as shared fields that are common to multiple Content Types, and system fields that the CMS defines. Most Content Types have a specific function; for example, the FastForward Image Content Type stores image files and the Calendar Content Type includes data for creating a calendar. A Content Type includes the Content Editor that displays its fields to users for creating or editing a Content Item.

In this chapter, we will demonstrate how to create some of the Content Types that you specified in the Modelling and Design section of this document:

- First we will create the Generic Content Type. We will begin with this Content Type because it is basic: it includes fields that already exist (except for a required dummy local field) and includes no special features. The Generic Content Type is a good example for demonstrating the basic procedure for creating a Content Type.
- Then we will create the Image Content Type. We include this Content Type because most systems require one or more Content Types that upload images. Furthermore, it includes a single local field, which allows us to introduce the concept of creating local fields in a Content Type.
- Finally, we will create a modified version of the Events Content Type that includes a child field set. A child field set is a field that stores a table of data. We have included the modified Events Content Type because implementers of Content Types that require child field sets must know the procedure for configuring them.

---

Note that you most likely have the FastForward Generic, Image, and Event Content Types on your system as part of Rhythmyx so we are using them in this chapter for demonstration purposes only. You would use the information in your implementation plan as substitute for the data used in the instructions in this chapter (or duplicate the Content Types we are creating but give them different names).

---

In most cases, we will only discuss Content Type fields when the information has not already been covered in *Creating Shared Fields* (on page 75). Some of the information that we will discuss includes how to create child field sets and how to override shared fields. We will also review some of the fields used to upload an image file since their functions are integral to the Image Content Type.

---

## Summary of Content Types

The three topics in this section outline the specifications for the Content Types that we will create in this chapter.

- *Generic Content Type* (see page 211)
- *Image Content Type* (see page 211)
- *Event Content Type* (see page 212)

### Generic Content Type

The *Generic Content Type specification* (see page 460) shows the fields in the FastForward Generic Content Type and their properties. Review this table now to see the fields included in the Content Type. Note that internally the FastForward name for this Content Type is rffGeneric.

Notice that the system fields `sys_title`, `sys_communityid`, `sys_lang`, `sys_currentview`, `sys_workflowid`, and `sys_hibernateVersion` are listed. By default, they are included in every Content Type because Rhythmyx uses them for internal processing of Content Items.

The Generic Content Type is intended to serve a variety of purposes, so its other fields serve common functions. They are all system or shared fields and are included in many Content Types. The `displaytitle` (Title) field holds the Content Item title that is visible to users. Three date fields, `sys_contentstartdate`, `sys_contentexpirydate`, and `sys_reminderdate` hold the dates for publishing and removing the content from a Web site, and a date for sending notifications (for any purpose). Keywords and description fields hold search words and phrases for locating the Content Item (in general, words and phrases that are not included in the text content of the item). Callout and body fields hold a summary of the body content and the body content, respectively. A filename field stores the filename of the Content Item, and the `sys_suffix` field stores the suffix portion of the filename. These fields are used to publish the Content Item to the correct location.

Since one local field is required in a Content Type, a local field named `placeholder`, a dummy field, is included.

Below the table, the Allowed Workflows, Default Workflow, and Communities that can view the Content Type are listed. We will refer to this table when we create the Generic Content Type in the section *Basic Content Type Creation* (on page 214).

### Image Content Type

The *Image Content Type specification* (see page 465) shows the fields in the FastForward Image Content Type and their properties. We will refer to this table when we create the Image Content Type in the section *Image Content Type Creation* (see page 229). Internally in FastForward, this is referred to as the `rffImage` Content Type.

Review this table now to see the fields included in the Content Type. Below the table, the Allowed Workflow, Default Workflow, and Communities that can view the Content Type are listed.

As in the Generic Content Type, the system fields `sys_title`, `sys_communityid`, `sys_lang`, `sys_currentview`, `sys_workflowid`, and `sys_hibernateVersion` are include by default for internal processing of Content Items.

Notice that some of the other fields used in the Generic Content Type for common functions are used in the Image Content Type for the same functions. The `displaytitle` (Title) field holds the Content Item title that is visible to users. Three date fields, `sys_contentstartdate`, `sys_contentexpirydate`, and `sys_reminderdate` hold the dates for publishing and removing the content from a Web site and a date for sending notifications (for any purpose). The description field holds search phrases for locating the Content Item (in general, phrases that are not included in the text content of the item). A `filename` field stores the filename of the Content Item, and the `sys_suffix` field stores the suffix portion of the filename. These fields are used to publish the Content Item to the correct location.

Most of the remaining fields in the Image Content Type are taken from the `sharedimage` field set and are used to upload images. Two versions of the same fields are included, one for uploading full size images (the full size image fields are prefixed with `img1`) and one for uploading a thumbnail graphic of the same image (the thumbnail image fields are prefixed with `img2`). Since many systems do not require the thumbnail image, the `img2` fields are hidden by default. The fields that are used to store the uploaded image are `img1` and `img2`. The other fields that begin with the `img1` and `img2` prefixes are used to store metadata associated with the image: `img1_filename` and `img2_filename` store the filename; `img1_ext` and `img2_ext` store the extension portion of the filename; `img1_type` and `img2_type` store the MIME type; `img1_height`, `_width`, and `_size` and `img2_height`, `_width`, and `_size` store the height, width, and size of the images; `img_alt` stores text to display if image display fails for `img1` or `img2`.

The `img_category` field is local to the Image Content Type and is used to assign a category to the image. The category has various functions, including finding the image in a search and determining whether to display the image on a Web page.

The `webdavowner` field stores the user who has a lock on the Content Item when content is uploaded through Rhythmyx's WebDAV feature. This document does not cover WebDAV. See the document *Implementing WebDAV in Rhythmyx* for information about WebDAV.

The `shared filename` and `webdavowner` fields and the `sharedimage img1_size` and `img1_ext` fields are hidden because they are used for Rhythmyx's internal processing.

In FastForward the Image Content Type is visible to the Enterprise Investments, Enterprise Investments Admin, Corporate Investments, and Corporate Investments Admin Communities. In our example, we will assume that content contributors only enter text and reserve the creation of Image Content Types for administrators. Therefore we will change the visible Communities to Enterprise Investments Admin and Corporate Investments Admin only. The purpose of this change is to demonstrate why you might choose to make a Content Type visible to certain Communities only.

## Event Content Type

The *Event Content Type specification* (see page 457) shows the fields in the FastForward Event Content Type and their properties. We will refer to this table when we create the Event Content Type in the section *Creating a Content Type with a Child Field Set* (see page 240). Internally the FastForward name for this Content Type is `rffEvent`.

Review this table now to see the fields included in the Content Type. Below the table, the Allowed Workflow, Default Workflow, and Communities that can view the Content Type are listed.

As in all Content Types, the system fields `sys_title`, `sys_communityid`, `sys_lang`, `sys_currentview`, `sys_workflowid`, and `sys_hibernateVersion` are include by default for internal processing of Content Items.



Notice that some of the other fields used in our other Content Types for common functions are used in the Event Content Type for the same functions. At this point, the practicality of using system and shared fields should be evident; all three of our Content Types have largely reused existing fields. The `displaytitle` (Title) field holds the Content Item title that is visible to users. Three date fields, `sys_contentstartdate`, `sys_contentexpirydate`, and `sys_reminderdate` hold the dates for publishing and removing the content from a Web site and a date for sending notifications (for any purpose). The keywords and description fields hold search words and phrases for locating the Content Item (in general, words phrases that are not included in the text content of the item). `Callout` and `body` fields hold a summary of the body content and the body content, respectively. A `filename` field stores the filename of the Content Item, and the `sys_suffix` field stores the suffix portion of the filename. These fields are used to publish the Content Item to the correct location.

The Event Content Type uses four local fields that hold event information. `event_start` and `event_end` hold the start and end dates for an event. `Event_type` lets the content contributor choose a type of event; the contents of this field can be used for searching Event Content Items or determining which should be included on a Web page. In FastForward, the `event_location` field uses a `sys_EditBox` control that lets the content contributor enter a simple location for the event. We have changed `event_location` to use a `sys_Table` control. When the implementer chooses the `sys_Table` control the entry is no longer referred to as field but is called a child field set. We have done this to demonstrate how to create and use a child field set. Our `event_location` child field set has four entries as shown in the following table:

`event_location` child field set:

Name	Label	Control Name	Occur	Data Type	Format
<code>event_city</code>	Event City:	<code>sys_EditBox</code>	optional	text	50
<code>event_state</code>	Event State:	<code>sys_EditBox</code>	optional	text	50
<code>event_address</code>	Event Address:	<code>sys_TextArea</code>	optional	text	255
<code>event_contact</code>	Event Contact:	<code>sys_TextArea</code>	optional	text	255

We have also modified the *shared/callout* field to use `sys_EditBox` control instead of the default `sys_EditLive` control so that we can demonstrate how and why to override the properties of a shared field.

## Basic Content Type Creation

You create Content Types using the Rhythmyx Workbench's New Content Type Wizard and Content Type Editor. Once Content Type objects are created, you can access them in the Content Design view of the Rhythmyx Workbench to edit or view them in the Content Type Editor. See the *Rhythmyx Workbench Online Help* for information about the New Content Type Wizard and Editor.

You create and access Content Types from the Rhythmyx Workbench's Content Design view. You can create any number of user-defined sub-folders for storing your Content Types, however, you cannot add or modify Content Types in the Navigation folder; its Content Types are defined in the navigation.properties file. See the chapter *Managed Navigation* (see page 261) for more information about managed navigation.

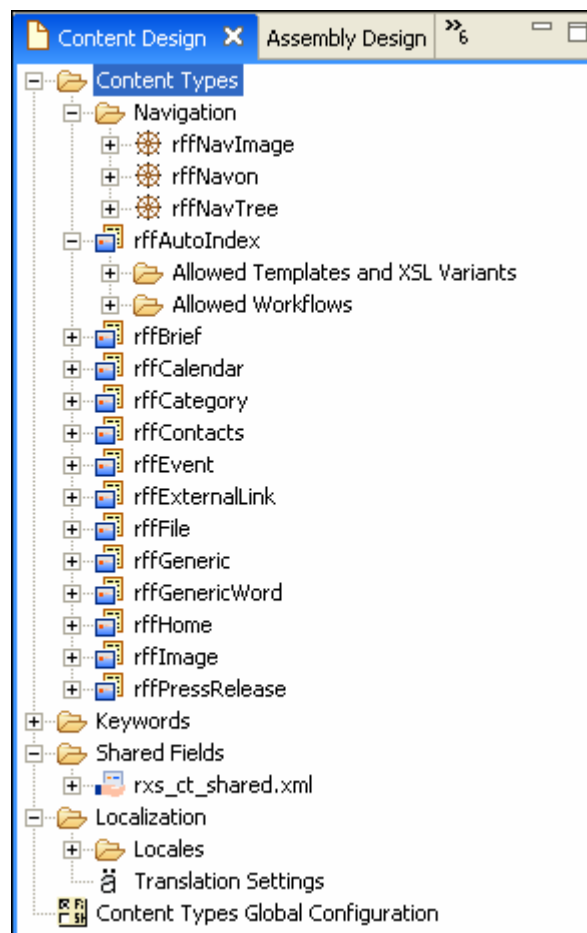


Figure 152: Content Design View

This section will show you how to create the FastForward Generic (*rffGeneric*) Content Type, which provides a good starting point because it is composed of previously created shared and system fields (except for the one required local field, Placeholder) and includes no special features. Its main fields are the Display Title, Body, and Callout (summary).

The *Generic* Content Type can be used for a range of purposes because many varieties of content simply require a display title, a body, and a summary.

---

*Note: You cannot create a Content Type named Generic, since it already exists in FastForward. Instead, create a similar Content Type included in your implementation plan or copy our steps but give your Content Type a different name.*

---

This section includes the following steps for creating and viewing the Generic Content Type.

- 1 **Creating the Generic Content Type object** (see page 215).
- 2 **Including fields** (see page 219).
- 3 **The Generic Content Editor** (see page 225)
- 4 **Viewing Generic Content Items** (see page 226).

## Creating the Generic Content Type Object

In this topic we will initially create your version of the Generic Content Type object using the New Content Type wizard. We will assume that this is the first of your modelled Content Types that you are creating, and we will begin by creating a subfolder below the Content Types node for holding the modelled Content Types, which will all be used on your customer site. We will call the folder *CustomerSite*.

To create your Generic Content Type:

- 1 In the Rhythmyx Workbench, make Content Design the visible view.
- 2 Right-click the Content Types folder and choose *New > Folder*.  
A sub-folder named *New Folder* appears under the Content Types folder.
- 3 Right-click on *New Folder* and choose *Rename*.  
The folder name is highlighted.
- 4 Type *CustomerSite* and press ENTER.  
The folder is now named *CustomerSite*.
- 5 Click the *CustomerSite* folder and in the menu bar choose *File > New > Content Type*.  
The Rhythmyx Workbench displays the New Content Type wizard.
- 6 In **Content Type name**, enter the name *Generic* for the *Generic* Content Type.  
The wizard automatically enters the name in **Label**. Leave the value in **Label**.
- 7 In **Description**, optionally enter a description of the Generic Content Type.



- 8 To make the Generic Content Type visible to all Communities except Default, click Default under **Visible to these communities** and click  to move it to the **Available communities** list box.

Figure 153: New Content Type wizard, first screen

*Note: The Default Community is included with Rhythmyx and is not part of the FastForward sample; therefore it is not usually given access to FastForward components.*

- 9 Click [Next].

The Workflow dialog of the wizard opens.

- 10 To enable the arrow buttons, click one of the Workflows in the **Available workflows** list box. Since you want to Allow users to assign either the *Simple* or the *Standard* Workflow to the *Generic* Content Type, click  to move them both from the **Available workflows** list box to the **Allowed workflows** list box.

11 Under Default, choose *Standard Workflow*.

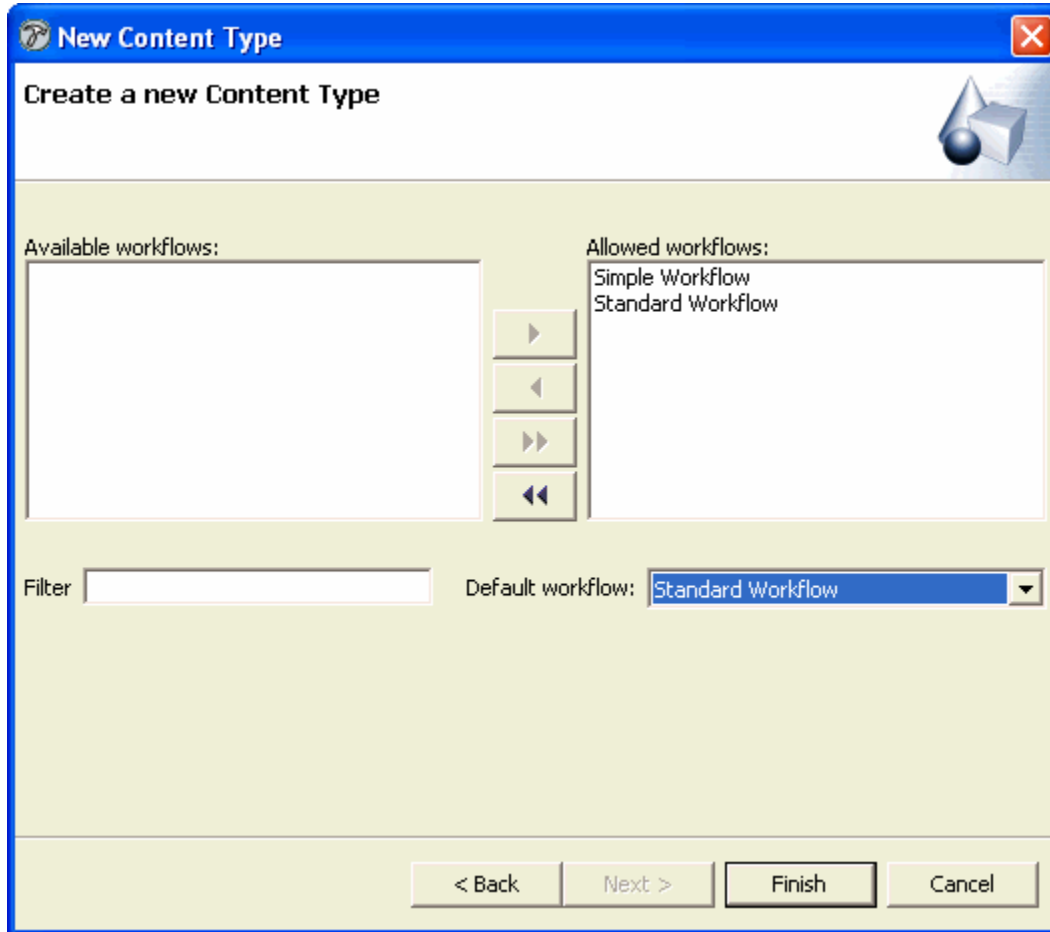


Figure 154: New Content Wizard, second dialog

12 Click [**Finish**].

The wizard closes. The Content Type editor opens in a window in the Workbench so that you can add fields to the Content Type.

Your *Generic* Content Type object is created and appears under the *CustomerSite* folder in Content Design view. Since you have established the Workflows available to the Content Type, they are listed below it under the Allowed Workflows folder. At this point, you have not created any templates that are local to the Content Type so they are not listed below the Allowed Templates and XSL Variants folder.

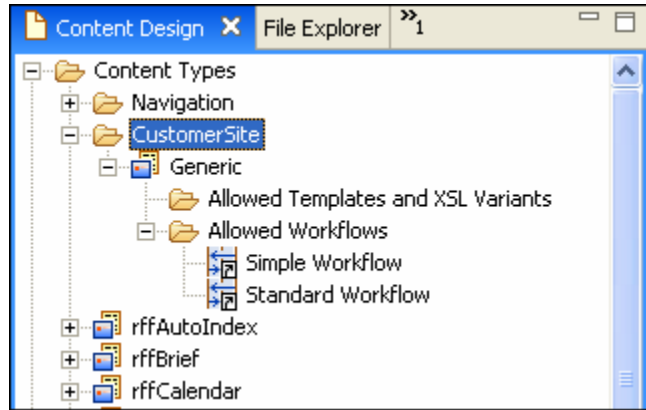


Figure 155: New Content Type in Content Design View

## Including Shared and System Fields

After you complete the New Content Type wizard and click **[Finish]**, the Content Type editor automatically opens to the Content Type tab in a Workbench window. The Content Type already displays the following mandatory system fields in the Fields and Field Sets table:

- `sys_title` - Rhythmyx's internal title for Content Type
- `sys_communityid` - Community assigned to Content Type; by default, the Community of the login user.
- `sys_lang` - Locale assigned to Content Type; by default, the Locale of the login user.
- `sys_currentview` - (used internally by system)
- `sys_workflowid` - Default Workflow assigned to Content Type.
- `sys_hibernateVersion` - (used internally by system)

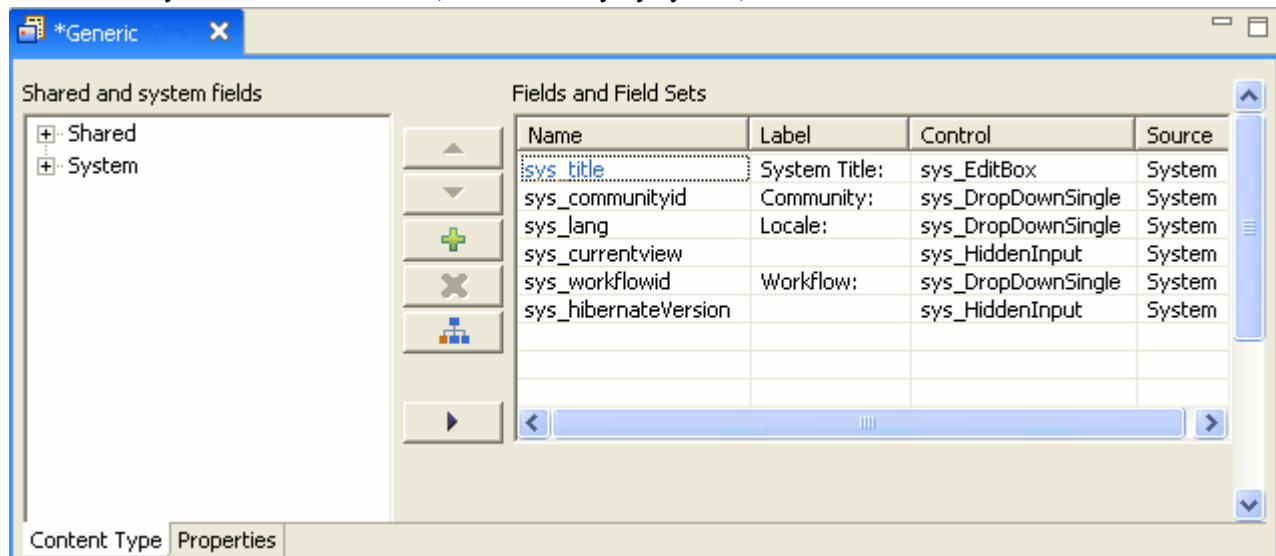


Figure 156: Content Type Editor

You can move the position of these fields in the table to insert other fields above or below them.

For the Generic Content Type, you simply include additional existing shared and/or system fields. Since one local field is required, we will also add a dummy local field.

To include fields in the Generic Content Type:

- 1 Refer to the *rffGeneric Content Type specification* (see page 460) to identify the first field to insert into the Generic Content Type. The first field in the table is `sys_title`. Since `sys_title` is included by default, identify the next field, `shared/displaytitle`. Since you will be using shared fields that you have created, they will have slightly different names than the fields in the specification.
- 2 On the Content Type editor's Content Type tab in the **Shared and system fields** box, expand *Shared*. The two shared field sets that you have added, *shared* and *sharedimage* appear.

- 3 Expand the field set *shared*.

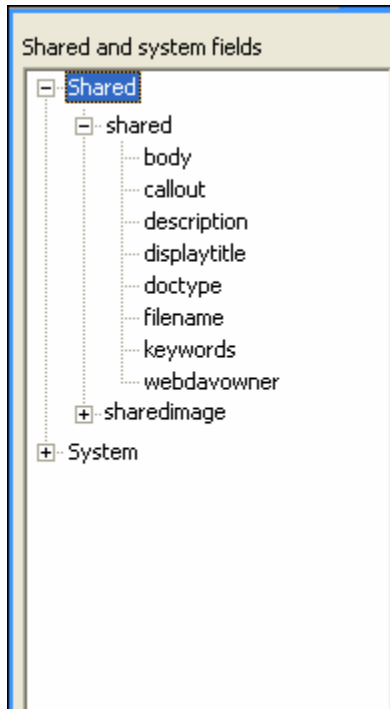

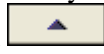


Figure 157: Expanded Shared Field Set in Content Type tab

- 4 Select *displaytitle* and click  to move it to the first empty row in the Fields and field sets table.
- 5 Since you want *displaytitle* to appear directly under *sys\_title*, select the row for *displaytitle* in the Fields and field sets table and click  until *displaytitle* appears directly under *sys\_title*.
- 6 Leave the default values for *displaytitle*.
- 7 Repeat steps 1 through 6 for each of the fields listed in the table in the *rffGeneric Content Type specification* (see page 460). Since some of the fields are system fields, expand *System* instead of *Shared* in the **Shared and system fields** box to locate the field. For each field, leave the default values and settings.

If you need help adding the local placeholder field, see *Including a Local Field* (see page 231).

When you are done, the Content Type tab should appear as follows. Note that not all of the fields are visible in the portion of the Fields and Field Sets table shown.



Since the Generic Content Type does not include any unique properties, you do not have to access the Properties tab of the Content Type editor.

- 8 To save the changes you have made in the editor, click the X on the Content Type tab:





The Save Resource dialog opens:

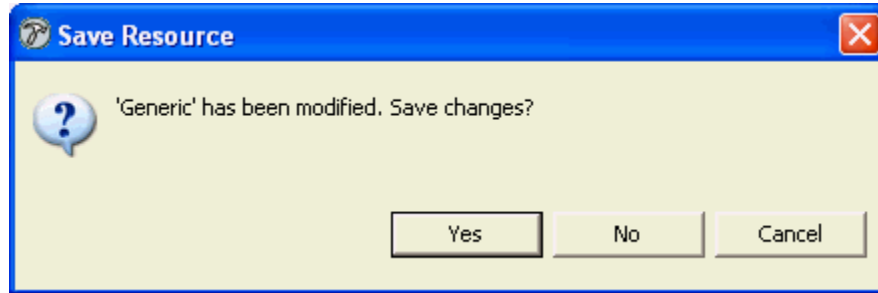


Figure 158: Save Resource dialog

**9** Click [Yes].

The Save Resource dialog closes and the Content Type editor closes.

Creation of your Generic Content Type is now complete.

In Community Visibility view, your Generic Content Type is listed under Communities that it is visible to.

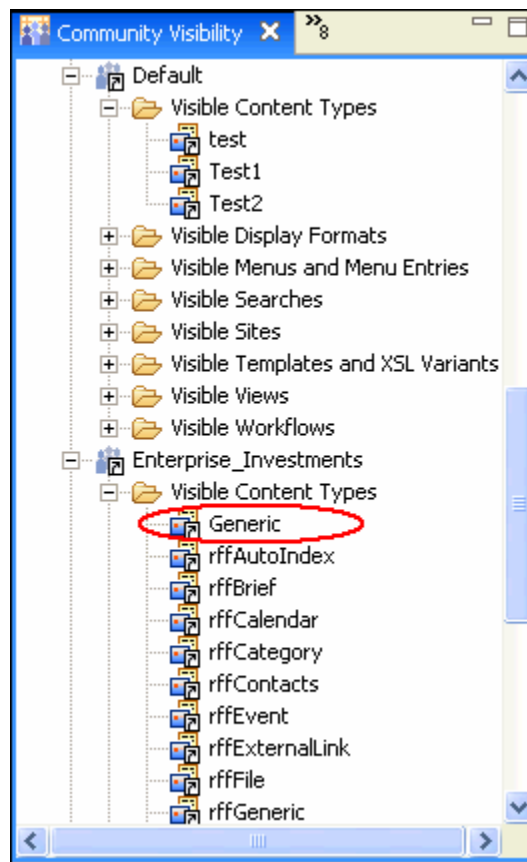


Figure 159: Community Visibility view

If you want to make the Generic Content Type visible to another Community, for example, *Default*, add the Default Community to the Object's ACL.

To add the Default Community to the Generic Content Type's ACL:

- 1 In Content Design View, right-click on Generic and choose *Security*.  
The Object ACL dialog opens for Generic.

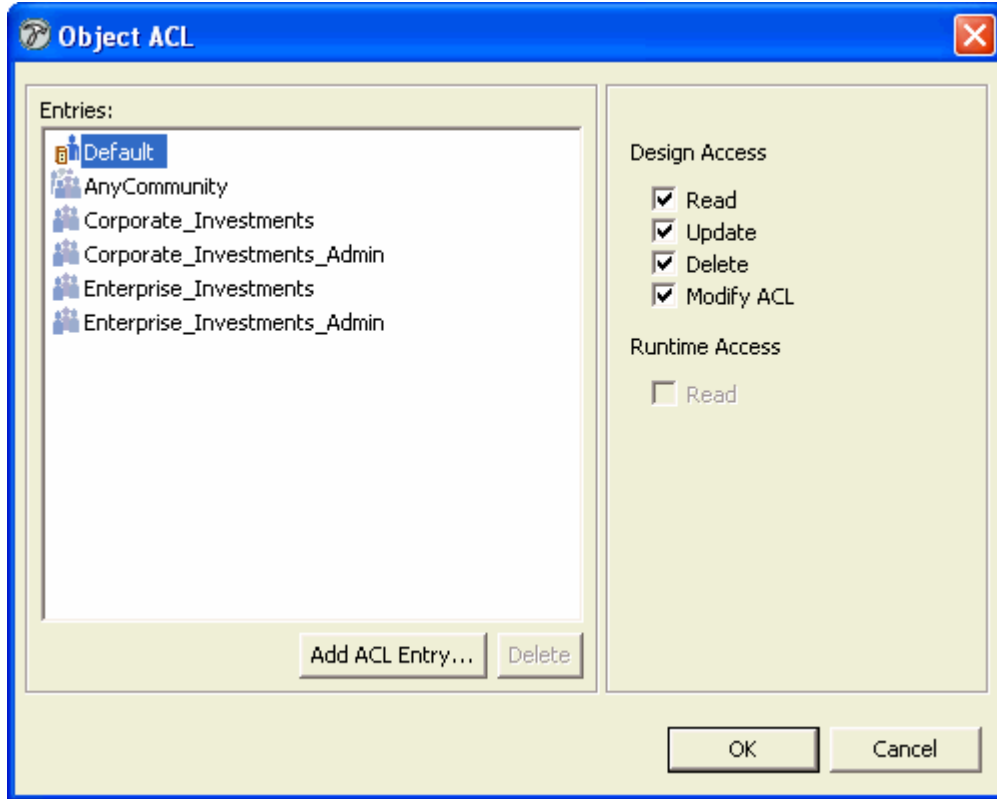


Figure 160: Object ACL dialog

Although a Default user has access to the object, the Default Community is not listed.

- 2 Click [**Add ACL Entry**].

The Add ACL Entry dialog opens.

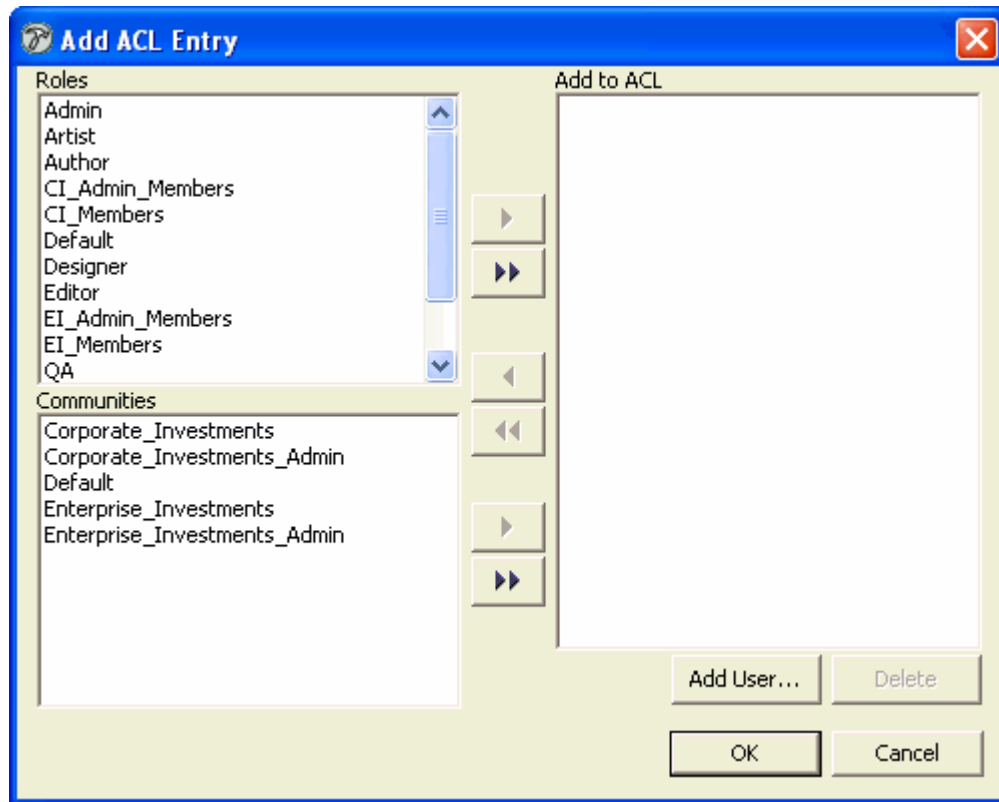


Figure 161: Add ACL Entry dialog

- 3 In the **Communities** table select *Default* and click the right arrow.
- 4 *Default* moves to the **Add to ACL** table.
- 5 Click [**OK**].

The Add ACL Entry dialog closes and the *Default* Community is added to the Entries table in the Object ACL dialog.

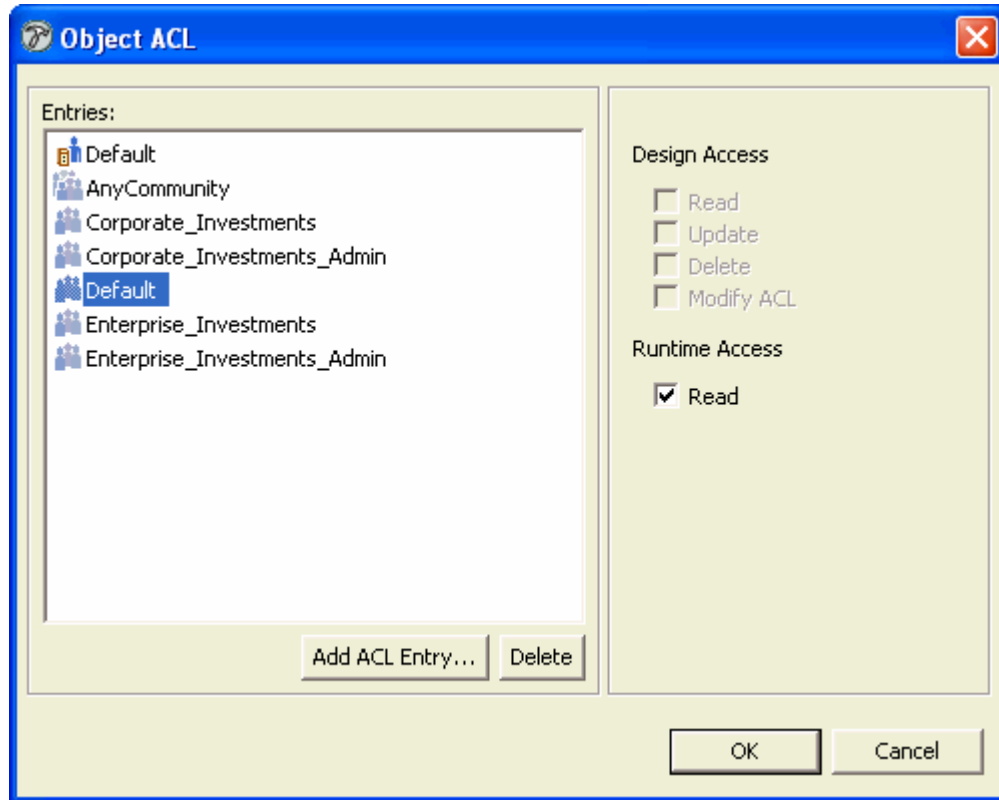


Figure 162: Object ACL dialog

**6** Click [OK].

The Default Community can now access the Generic Content Type.

---

NOTE: You could also have dragged and dropped the Generic Content Type onto the Default Community in Community Visibility view to give the Default Community access to the Generic Content Type. This method is especially useful for giving a Community access to multiple objects; instead of accessing the ACL for each object, you can multi-select the objects and drag and drop them onto the Community.

---

## The Generic Content Editor

At least one Template must be associated with a Content Type to view the Content Editor correctly. From Assembly Design view, drag the shared rffPgEiGeneric Template on top of the Generic Content Type's Allowed Templates and XSL Variants folder.

Since the Generic Content Type is complete, a user in Content Explorer can open the Generic Content Editor if the user is a member of a Community associated with the Content Editor. You do not have to restart the Rhythmyx Server to make the new Content Editor available in Content Explorer.

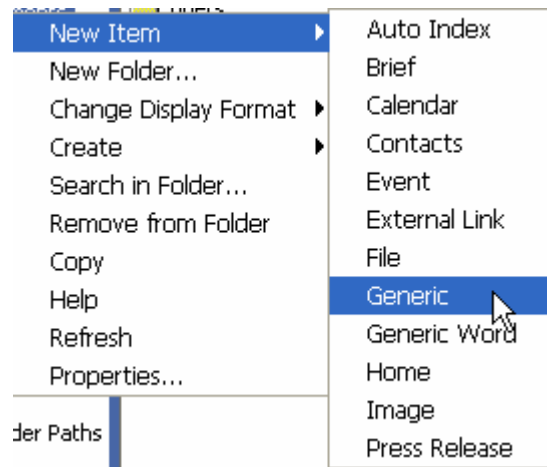


Figure 163: New Content Type in drop menu

The following graphic shows a Content Item entered in your new Content Editor.

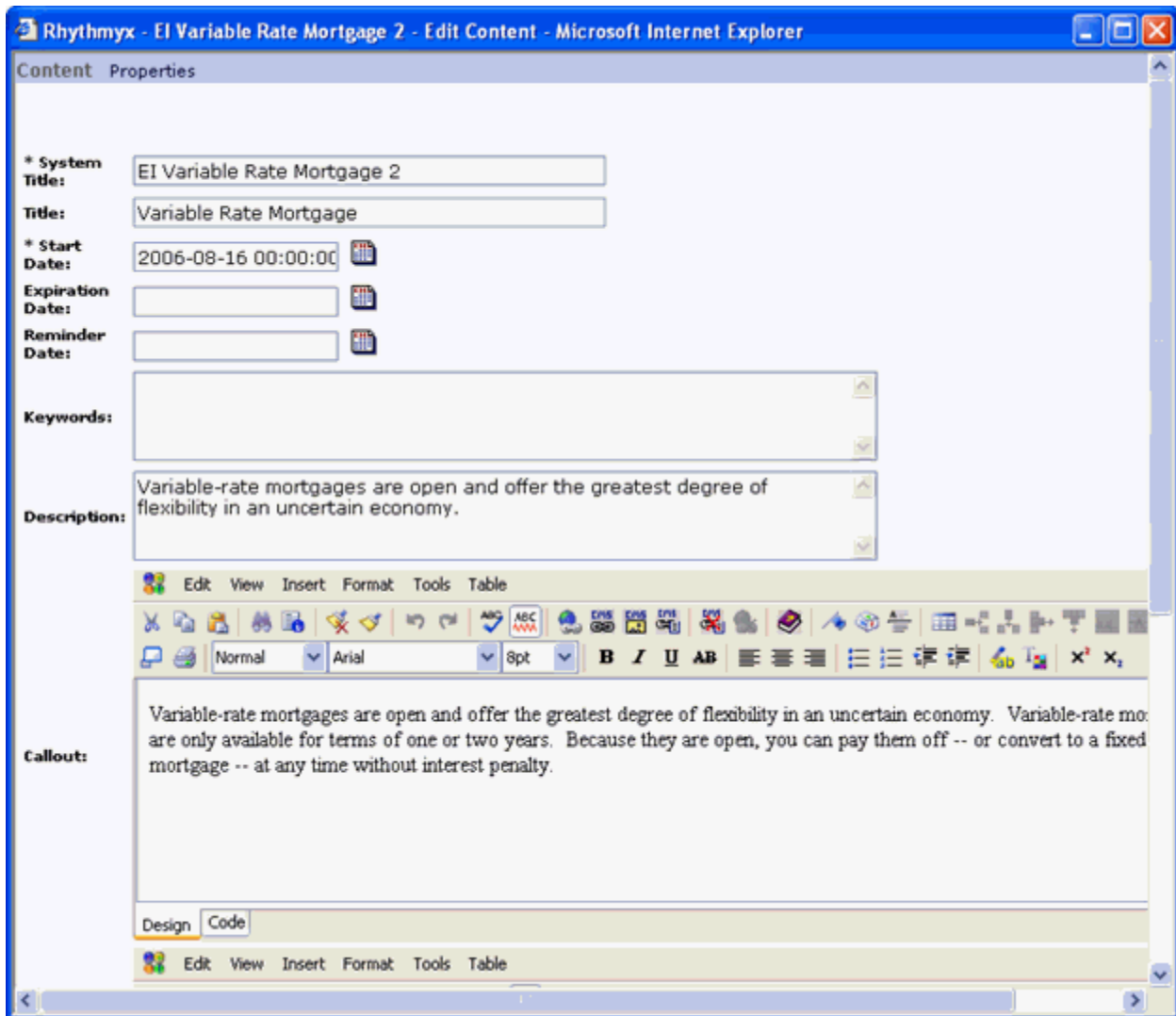


Figure 164: Generic Content Editor

If you refer back to the table in the *Generic Content Type specification* (see page 460), you can see that the fields appear in the order in which you entered them. The Content Editor displays the Labels and controls specified for each field. Note that the required fields have an asterisk next to them.

Since the filename, sys\_contentview, and sys\_hibernateVersion fields are hidden, they do not appear.

## Viewing Generic Content Items

In the recommended implementation roadmap, we create local templates after creating Content Types. However, in this document, the chapter that explains creating local templates precedes this chapter. Therefore, to understand how to create local templates for your new Generic Content Type, refer back to the chapter *Creating Slots and Templates* (see page 107).

When FastForward is installed, the Generic Content Type's references to Templates that the Generic Content Type can use appear under the Content Type's Allowed Templates and XSL Variants subfolder in Content Design view.

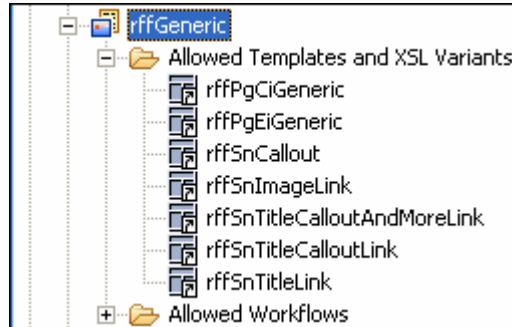


Figure 165: Allowed Templates for Generic Content Type

Once the templates are assigned to the Content Type, its Content Items can be published or previewed in the format of the template.

Here, we will preview a Generic Content Item using three different templates assigned to it. The templates shown below are rffPgEiGeneric, rffSnCallout, and rffSnTitleLink. rffPgEiGeneric and rffSnTitleLink are explained in detail in the chapter *Creating Slots and Templates* (see page 120).

The rffPgEiGeneric template displays the Generic Content Type in the following page format. Note that the graphics and navigation links on the top and left side are part of the global template. The local template only shows the Generic Content Item's *displaytitle* field (*Variable Rate Mortgage*) and the *body* field (the text below *displaytitle*).

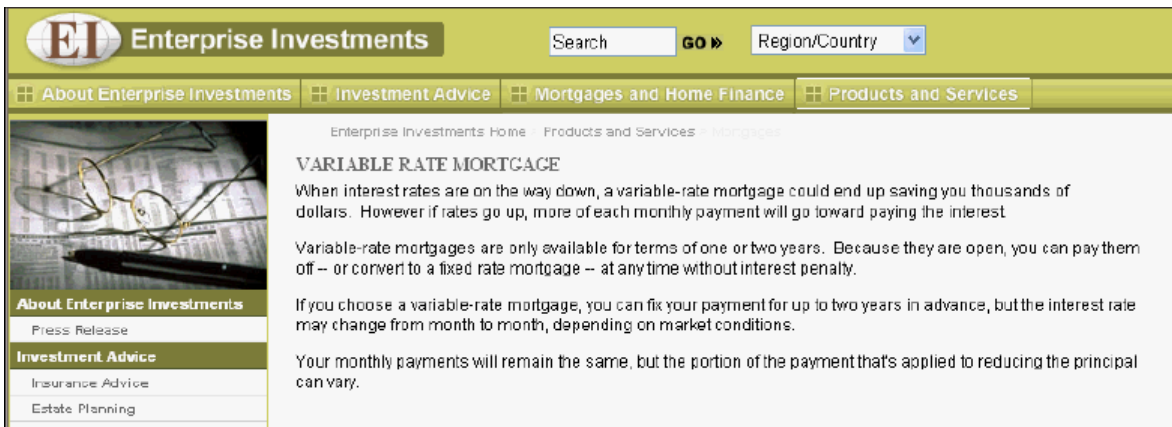
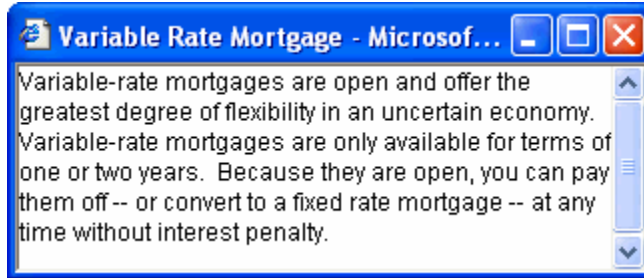


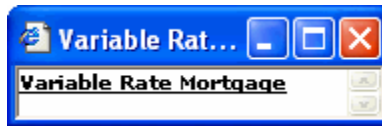
Figure 166: Generic Content Item formatted by page template

The rffSnCallout template displays the Generic Content Type in the following snippet format. This template only shows the callout (summary) field. To display this snippet on your Web site, you would have to include it on a Page template.



*Figure 167: Generic Content Item formatted with s-Callout template*

The rffSnTitleLink template displays the Generic Content Type in the following snippet format. This template only shows the displaytitle field as a link to the item in the rffPgEIGeneric format. You might include this snippet in a list of links to related topics on a page in your Web site.



*Figure 168: Content assembled in s-TitleLink template*



---

# Image Content Type Creation

This section shows you how to create a typical Content Type for uploading an image that you can use on pages in your Web site. Since FastForward includes the Image Content Type for this purpose, we will demonstrate how to create this Content Type.

We demonstrate the Image Content Type because most systems require one or more Content Types that upload images.

The Image Content Type consists of shared and system fields, like the Generic Content Type. In addition to including shared fields from the *shared* field set, the Image Content Type includes fields from the *sharedimage* field set, which defines fields that are used to upload an image. The Image Content Type also includes a local field, which we will add in the topic ***Including a Local Field*** (see page 231).

In the chapter ***Creating Shared Fields*** (on page 75), the topic ***sharedimage Field Set*** (see page 78) explains how image fields upload images. Let us review the process again here:

The field that uploads the file is assigned the `sys_File` control. In our Image Content Type, the field will be *img1*. The `sys_file` control lets the user select a file and uploads it into the *img1* field. When the `sys_File` control is used, a Java extension (Java plugin) that extracts and inserts metadata from the uploaded file into other fields in the Content Type is included. The extension looks for fields that are prefixed with the name of the upload field (in this case, *img1*) and suffixed with specific metadata labels. For example, it looks for the field *img1\_filename* and, if it finds it, inserts the uploaded file's file name into *img1\_filename*. In FastForward's Image Content Type, both the `sys_FileInfo` and `sys_imageInfoExtractor` extensions are used to perform this extraction of metadata; in our example, we only have to add the `sys_imageInfoExtractor` since `sys_FileInfo` is automatically added when we specify the `sys_File` control. For a full list of the fields that `sys_fileInfo` and `sys_imageInfoExtractor` look for and the content that they insert into them, see the topics *sys\_fileInfo* and *sys\_imageInfoExtractor* in the *Rhythmyx Technical Reference*. Note that the *sharedimage* Field Set does not include all of the fields that these extensions look for, but only the ones that implementers most commonly use in Content Types. You may include any fields with *sys\_fileInfo* and *sys\_imageInfoExtractor* metadata suffixes that are not included in the *sharedimage* Field Set as local fields. We will demonstrate how to add the `sys_imageInfoExtractor` java extension in ***Adding Pre-processing Extensions*** (see page 251) instead of in this section.





This section includes the following steps for creating and viewing the Image Content Type:

- 1 ***Creating the Image Content Type object*** (see page 230).
- 2 ***Including a local field*** (see page 231).
- 3 ***Entering Properties.*** (see page 233)
- 4 ***The Image Content Editor*** (see page 236).
- 5 ***Image Content Items*** (see page 238).

## Creating the Image Content Type Object

In this topic we will initially create your version of the Image Content Type object using the New Content Type wizard. We will place the object in the CustomerSite folder along with the Generic Content Type.

To create your Image Content Type:

- 1 In the Rhythmyx Workbench, make Content Design the visible view.
- 2 Click the *CustomerSite* folder and in the menu bar choose *File > New > Content Type*.  
The New Content Type wizard opens.
- 3 In **Content Type name**, enter a name for your *Image* Content Type, such as *Image*.  
The wizard automatically enters *Image* in **Label**. Leave the value in **Label**.
- 4 In **Description**, optionally enter a description of the Image Content Type.
- 5 In this example, you only want to make the Image Content Type visible to Communities assigned to administrators, since your other users are only responsible for entering text. Under **Visible to these communities**, click **Corporate Investments** and click  to move it to the **Available Communities** list box. Then, under **Visible to these communities**, click **Default** and click  to move it to the **Available communities** list box. Repeat the procedure for **Enterprise Investments**. (You could also use CTRL-click to select all three Communities and then click  to move them all at once to the **Available communities** list box.)
- 6 Click [**Next**].  
The Workflow dialog of the wizard opens.
- 7 Since you want to allow users to assign either the *Simple* or *Standard* Workflow to the Image Content Type, click  to move them both from the **Available workflows** to the **Allowed workflows** list box.
- 8 Under **Default**, choose *Standard Workflow*.
- 9 Click [**Finish**].

The wizard closes. The Content Type editor opens in a window in the Workbench so that you can add fields to the Content Type. Your Image Content Type object is created and appears under the *CustomerSite* folder in Content Design view. Since you have established the Workflows available to the Content Type, they are listed below it under the Allowed Workflows folder. At this point, you have not created any templates that are local to the Content Type so they are not listed below the Allowed Templates and XSL Variants folder.

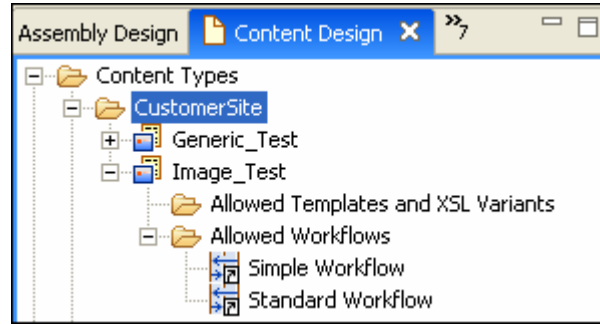


Figure 169: Image\_Test Content Type

- 10 The Content Type editor opens so that you can add fields to the Content Type.

## Including a Local Field

After you complete the New Content Type wizard and click **[Finish]**, the Content Type editor automatically opens to the Content Type tab. The Fields and Field Sets table on the Content Type tab includes the mandatory system fields. See *Including Shared and System Fields* (see page 219) for a list of these mandatory fields.

When we created the Generic Content Type, we added shared and system fields that already existed, so we will not cover how to add these types of fields in this topic. Here, we will show how to add a local field.

To include fields in the Image Content Type:

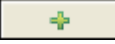
- 1 Refer to the topic *rffImage Content Type specification* to identify the first field to insert into the Image Content Type. The first field in the table is `sys_title`. Since `sys_title` is included by default, identify the next field, `shared/displaytitle`. Remember to use the `shared` and `sharedimage` shared field sets and their fields that correspond to those in the `rffImageContentType` specification.
- 2 Include the `shared/displaytitle` field in the Image Content Type as described in steps 1 through 6 in the topic *Including Shared and System fields* (see page 219).
- 3 Continue to add shared and system fields to the Image Content Type as described in the topic *Including Shared and System fields* (see page 219) until you reach the local field `img_category`.

**4** To add the `img_category` field complete the following steps.


---

NOTE: Local fields are entered in the same manner that shared fields are entered into shared field sets. For in depth discussions about field properties and the values that you can enter into them, in the chapter *Creating Shared Fields*, see the topics **Implementing the "shared" Field Set** (see page 81) and **Implementing the "sharedimage" Field Set** (see page 93).

---

- a) In the first empty row in the Fields table, under Name, click in the cell and enter `img_category`. You may have to click  if the next available column is hidden from your view. `img_category` is the internal name Rhythmyx uses for the field. It is best practice to enter all field names in lower case. The editor automatically enters `Img_category`: under Label.
- b) Change the entry in Label to *Image category*:
- c) Click the cell under Control to access a drop list of control options. At this point, all of controls (except `sys_table`) are available, including array controls (`sys_checkBoxGroup`, `sys_checkBoxTree`, and `sysDropDownMultiple`) as well as non-array controls (all others). However, once you save the Content Type, if you want to edit the control associated with the field, you will only see array or non-array control options depending on your initial choice. In other words, if you initially choose an array control, you can only change the field to use another array control; but if you initially choose a non-array control, you can only change the field to use another non-array control.

Choose `sys_DropDownSingle`. Populate the control with the choices from the Keyword *FF Image Category*. For information about the `sys_DropDownSingle` control and how to populate it with keyword choices, see the topic **Implementing a List Control** (see page 90) in the chapter *Creating Shared Fields*. For instructions on creating a Keyword and Keyword Choices, see the topic **Creating and Using Keywords** (see page 273) later in this chapter.

- d) Under Field Properties, choose *G* in Mnemonic, and leave all of the other properties at their default values.
  - e) Use  to move the `img_category` field under the `img_alt` field.
- 5** Add the remainder of the shared and system fields. Move them into the order specified in the ***rffImage Content Type specification*** (see page 465). You can use ALT + click to choose a group of fields and move them up or down together.

When you are done, the Content Type tab should appear as:

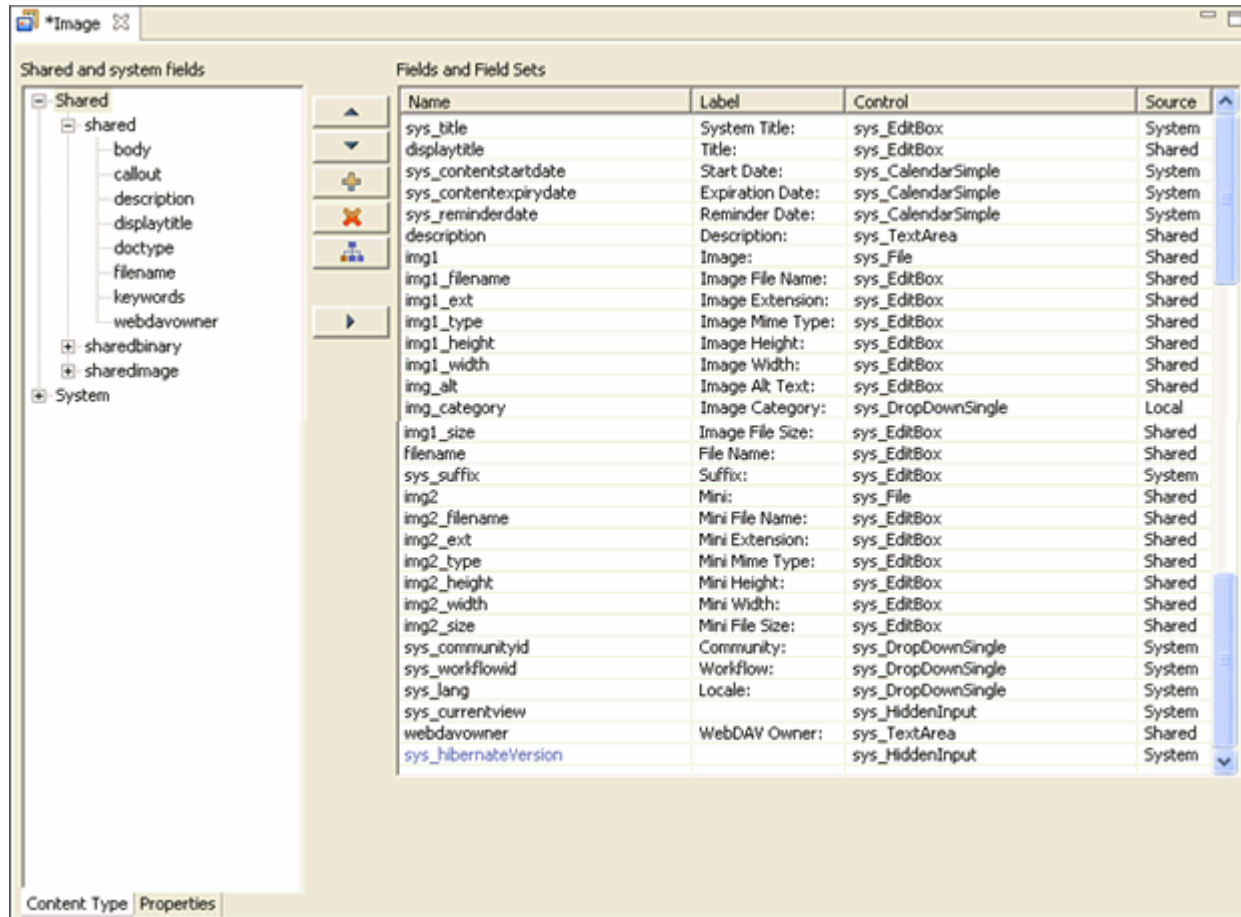


Figure 170: Content Type tab, Image Content Type

- Now, to *enter additional properties* (see page 233), click the Properties tab of the Content Type editor.

## Entering Content Editor Properties

Most of the fields on the Properties tab of the Content Type editor are filled in with values entered in the New Content Type wizard. These fields (Label, Description, Allowed workflows and Default workflow) are duplicated on the Properties tab so that implementers can edit them after the Content Type is created. We will not discuss these fields because you have already given them values when you entered your Image Content Type in the New Content Type wizard.

By default, Enable Searching for this Content Type is checked. Since only administrators use our version of the Image Content Type, we do not want it available for searching. Our administrators plan to keep the few Image Content Items that they create in a single folder where they can always find them. Therefore, uncheck Enable Searching for this Content Type.

The Properties tab includes a tabbed box for entering item transforms or validations and pre-processing and post-processing extensions (java plugins)

Since the Image Content Type requires several pre-processing extensions, we will add them to the Pre-Processing tab. You can move ahead to the section for adding the pre-processing extensions now, or proceed to close and save the Content Editor, and reopen it and add them later.

- See *Item Transformation, Validation, and Pre- and Post-processing* (see page 250) for definitions of item transforms, validations, and pre and post-processing extensions.
- See *Adding Pre-processing Extensions* (see page 251) for instructions on adding the pre-processing extensions that the Image Content Type requires.

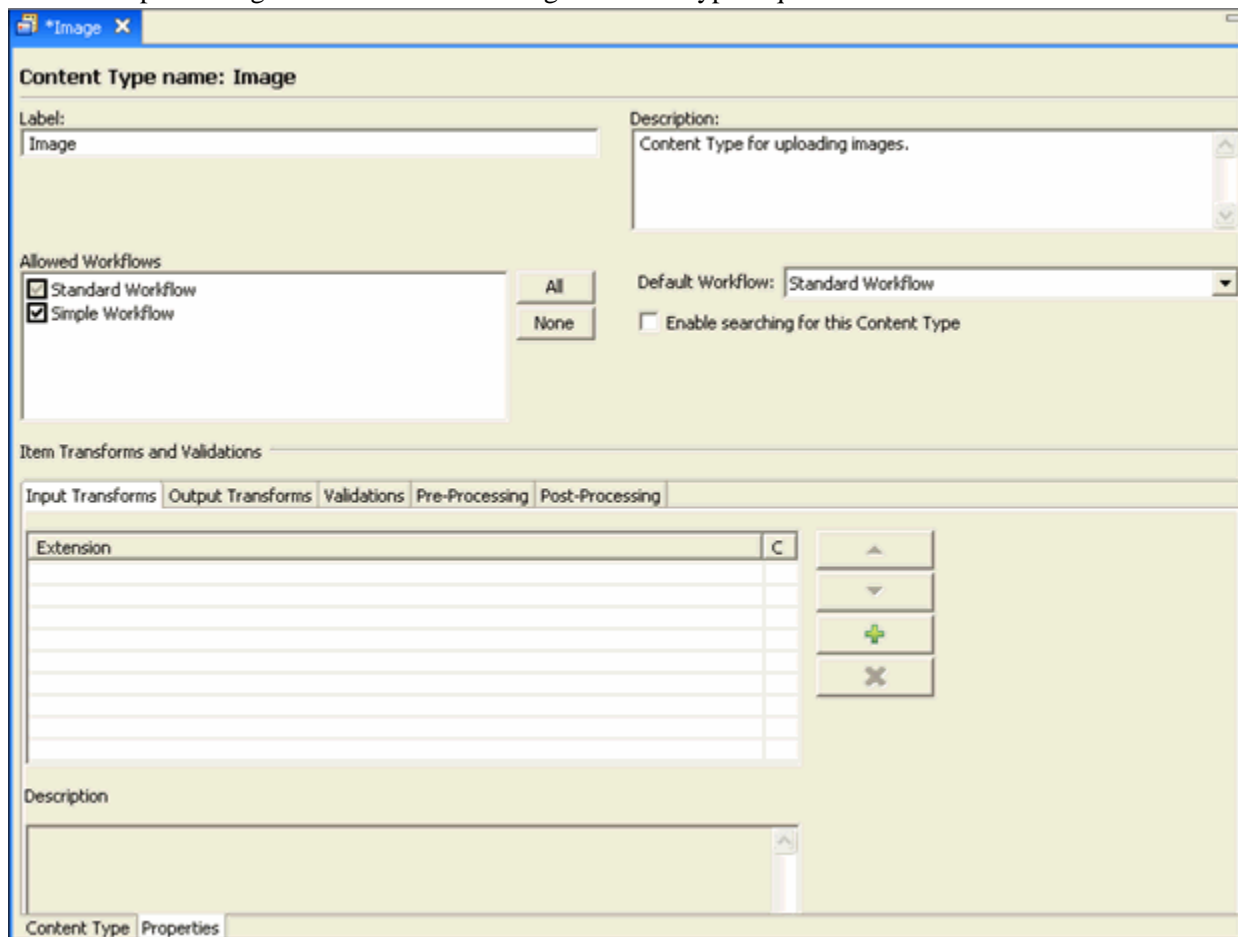


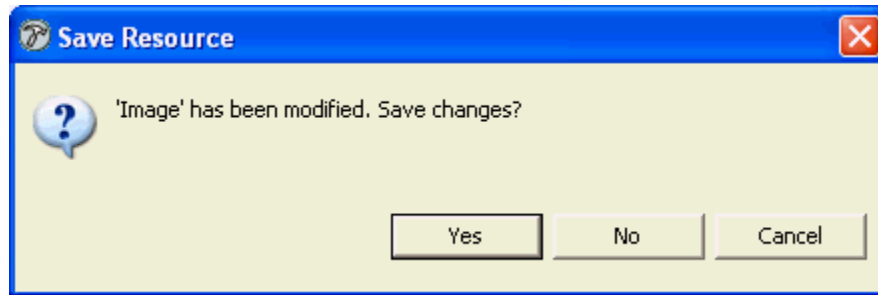
Figure 171: Image Content Type Properties tab

Once you have completed the Properties tab, creation of the Image Content Type is complete.

To save changes you have made and close the Content Type editor:

- 1 In the Menu bar, choose *File > Close*.

The Save Resource dialog opens:



*Figure 172: Save Resource Dialog*

**2** Click [Yes].

The Save Resource dialog closes and the Content Type editor closes.

The Image Content Type appears in the Community Visibility view under the Communities that you made it visible to, *Enterprise Investments Admin* and *Corporate Investments Admin*:

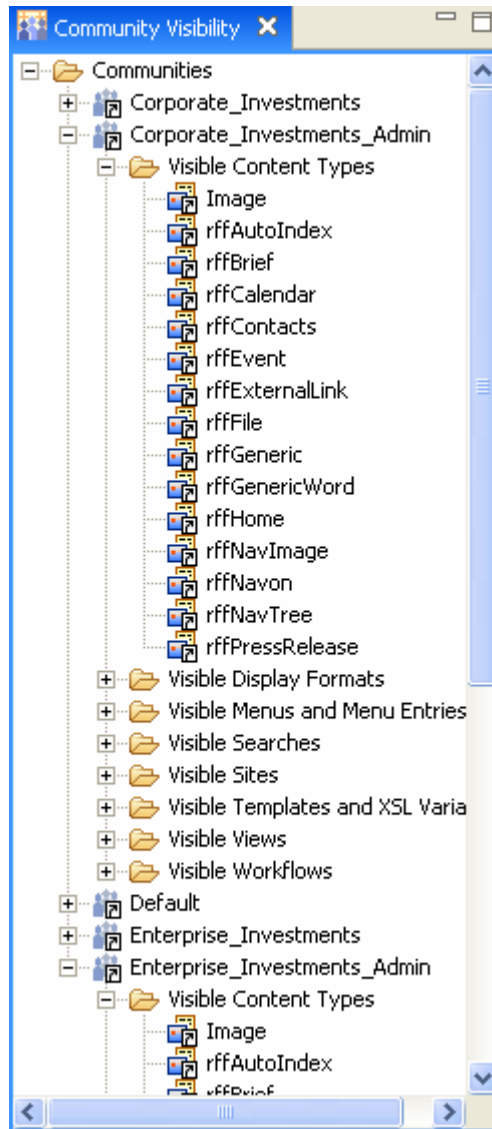


Figure 173: Communities with Image Content Type

Now, an administrator in Content Explorer can open your Image Content Editor.

## The Image Content Editor

At least one Template must be associated with the Image Content Type to view the Image Content Editor correctly. From Assembly Design view, drag the shared rffSnTitleLink Template on top of the Image Content Type's Allowed Templates and XSL Variants folder.

The Image Content Type is still not complete, because the pre-processing extension that uploads the image file had not yet been added. If you want to be able to view the Image Content Type as it is shown below, perform the steps in *Adding Pre-processing Extensions* (see page 251) now. Otherwise, you can open the Content Editor in Content Explorer, but you cannot upload an image.



In the following graphic, an image is already uploaded into the Content Editor and pre-processing extensions have filled in metadata. To initially upload a file, a user clicks **[Browse]** and chooses an image. The user must click **[Insert]** to enter the metadata properties in the Image File Name, Image Mime Type, Image Height, and Image Width fields.

The screenshot shows a web browser window titled "Rhythmyx - Banner2 - Edit Content - Microsoft Internet Explorer". The main content area is titled "Content Properties" and contains the following form fields:

- \* System Title: Banner2
- Title: Banner
- \* Start Date: 2006-08-16 00:00:00
- Expiration Date: (empty)
- Reminder Date: (empty)
- Description: (empty text area)
- Image: (empty text box) [Browse...] Preview File  Clear
- Image Mime Type: image/gif
- Image Height: 75
- Image Width: 205
- \* Image Alt Text: Banner
- Image Category: Photo (dropdown menu)
- Related Content: S - Title Link [Edit All]

At the bottom of the form are "Update" and "Close" buttons.

Figure 174: Image Content Editor

Notice that the fields appear in the order that you entered them, with asterisks next to required fields. Since the `sys_file` control has already uploaded the image, the text box beside the image field no longer holds the file name. The `sys_imageInfoExtractor` extension has located the Image Mime Type, Image Height, and Image Width fields and filled them with the appropriate values.

**Field visibility rules** (see page 99) associated with the Image Extension, Image File Size, Filename, and `webDAVOwner` fields as well as all of the thumbnail fields prevent them from appearing in the Content Editor. The internal `sys_currentView` field does not appear because it is assigned a `sys_hiddenInput` control.

If a user clicks Preview File beside Image, a preview of the image pops up.

To change the uploaded file, the user clicks **[Browse]** and chooses a new image. The user must click **[Update]** to enter the new metadata properties in the Image Mime Type, Image Height, and Image Width fields.

## Viewing Image Content Items

When FastForward is installed and the Image Content Type's local templates are included, the Content Type appears as follows in the Content Design view:

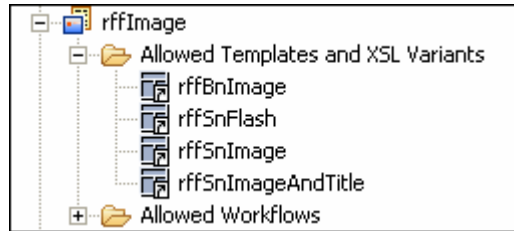


Figure 175: Allowed Templates for Image Content Type

Here we will preview an Image Content Item through the `rffSnImage` snippet template. The `rffSnImage` template simply displays the uploaded image. You would most likely include this image on a page template that includes related text content.

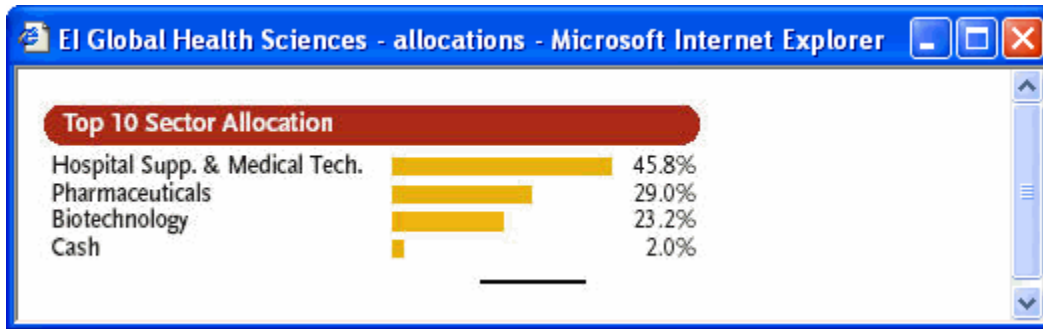


Figure 176: Image Content Item formatted with S-Image template

## WebImageFX

If you use the `sys_File` control to upload your images, you cannot modify them in Rhythmyx. If you want to modify the images, you must use a third-party application and upload the modified image to Rhythmyx, overwriting the original image file.

You can provide users with a limited capability of editing images if you use the `sys_WebImageFX` control instead of the `sys_File` control. The `sys_WebImageFX` control provides access to Ektron's WebImageFX graphics editor. For technical details about the WebImageFX editor and the `sys_WebImageFX` control, including information about customizing and upgrading the WebImageFX editor, see the *Rhythmyx Technical Reference*.

The following limitations apply to Content Editors that use the `sys_WebImageFX` control:

- The field containing the `sys_WebImageFX` control must be named *uploadfilephoto*.
- Due to the restriction on the name of the field, a Content Type cannot have more than one field that uses the `sys_WebImageFX` control. Only the first field that uses the control will be able to upload and edit graphics. The other controls will not be able to upload any file.

- A Content Type that uses the `sys_WebImageFX` control cannot use the `sys_File` control. If any field has the `sys_File` control, it will not be able to upload any file.
- When you add the `sys_WebImageFX` control to a Content Type, the `sys_FileInfo` pre-processor extension is automatically added to the Content Type. The fields that store the data returned by this extension must be prefixed with the string *uploadfilephoto*.

NOTE: The first time a user opens a Content Editor that uses the `sys_WebImageFX` control, the browser will prompt them to install WebImageFX. Users should follow the instructions in the WebImageFX installation wizard.

## Creating a Content Type with a Child Field Set

This section shows you how to create a Content Type that contains a child field set. A child field set is a table with any number of sub-fields. The following graphic shows the Content Editor with the example child field set we will create in this section. The child field set appears as a table with entries in the Content Editor.

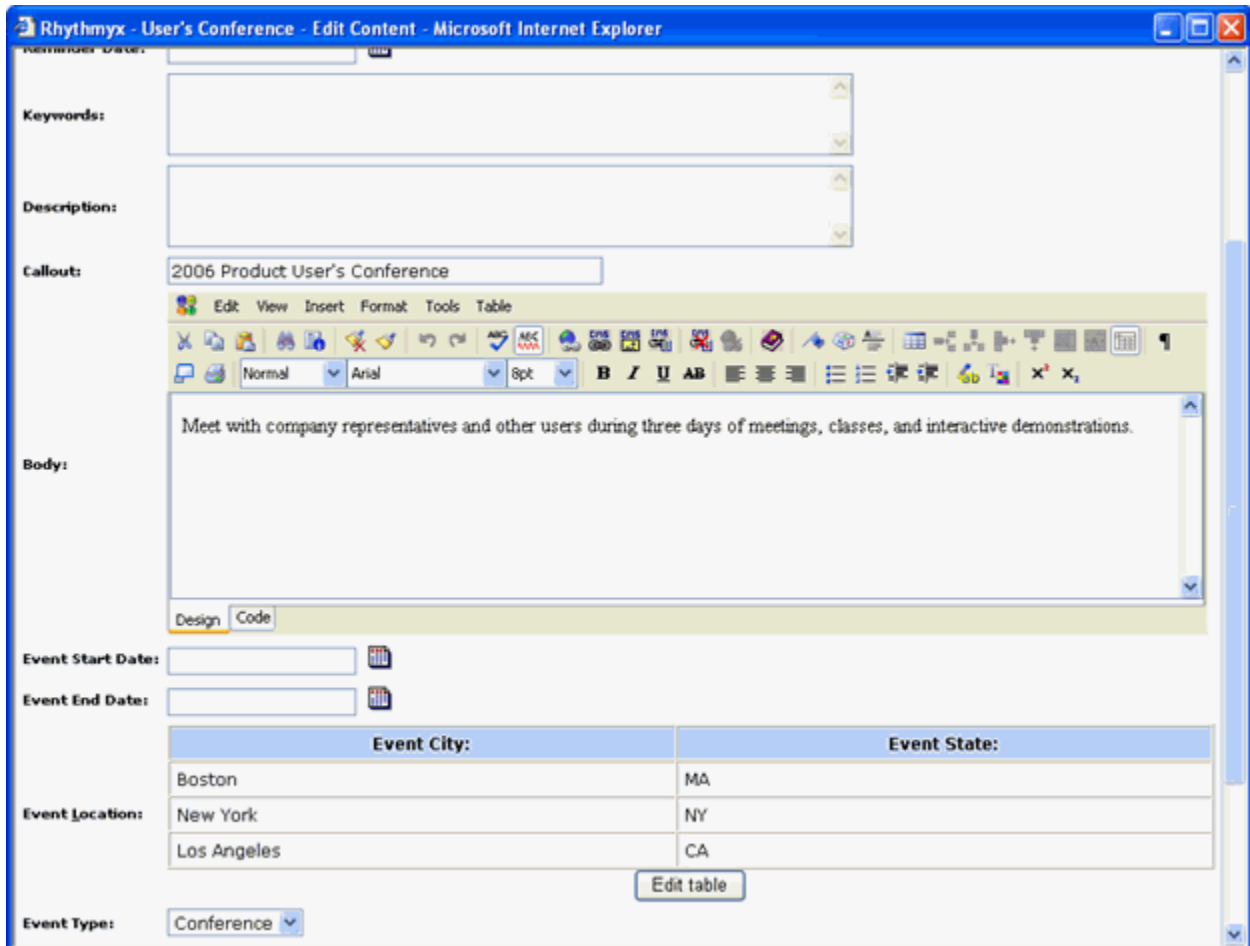


Figure 177: Content Editor with child field set

We demonstrate how to create a Content Type that uses a child field set because you may have a Content Type that requires this type of table field. In our example, we use a child field set that includes four fields for event locations. The fields are *city*, *state*, *address*, and *contact*. Since FastForward does not include a Content Type with a child field set, we have changed the *event\_location* field in the Event Content Type to a child field set.

The new specification for the event\_location child field set is the following:

event\_location child field set:

Name	Label	Control Name	Occur	Data Type	Format
event_city	Event City:	sys_EditBox	optional	text	50
event_state	Event State:	sys_EditBox	optional	text	50
event_address	Event Address:	sys_TextArea	optional	text	255
event_contact	Event Contact:	sys_TextArea	optional	text	255

To demonstrate how to override a shared field, we have also changed the shared/callout field to use a different control. In the FastForward version of the Event Content Type, the shared/callout field uses the sys\_EditLive control, but in our version, we assign it the sys\_EditBox control since we do not want to give users the ability to format its content.

See the *Event Content Type specification* (see page 457) for a table listing all of the FastForward fields in the Event Content Type and their values.

Since you have already created two Content Type objects using the New Content Type wizard, we will not repeat the process here. See *Creating the Generic Content Type Object* (see page 215) and follow the same instructions using the specifications in the *Event Content Type* (see page 212) topic instead. Once you have created the Content Type object, work through the other topics in this section.

---

*Note: You cannot create a Content Type named Event, since it already exists in FastForward. Instead, create a similar Content Type included in your implementation plan or copy our steps but give your Content Type a different name.*

---

This section includes the following steps for completing and viewing the Event Content Type:

- 1 **Overriding a shared field** (see page 241)
- 2 **Including a child field set** (see page 242)
- 3 **Viewing the Event Content Editor** (see page 246).
- 4 **Viewing Event Content Items** (see page 249).

## Overriding a Shared Field

After you create the initial Event Content Type object by completing the New Content Type wizard and clicking [**Finish**], the Content Type editor automatically opens to the Content Type tab. The Content Type already includes the mandatory system fields. See *Including Shared and System Fields* (see page 219) for a list of these fields.

In this topic, we will demonstrate that you can override the properties of a shared or system field in the local definition of a Content Type. You do this because you want the field to have a different property in this Content Type only. If you want a shared field to have a different property in all Content Types, you change the value directly in the shared field. If you want a system field to have a different value in all Content Types, you create a shared field with the new property and use it in all Content Types instead of the system field. *Note: Changing a system field directly is not recommended even if you want the field's properties to be permanently different because your changes will be overwritten when you upgrade your system. Changes to shared fields are not overwritten when you upgrade your system.*

In the FastForward version of the Event Content Type, the shared/callout field uses the `sys_EditLive` control, but in our version, we want to assign it the `sys_EditBox` control since we do not want to give users the ability to format its content.

To include the other fields in the Event Content Type follow the same procedures you used to *add shared and system fields to the Generic Content Type* (see page 219) and *add a local field to the Image Content Type* (see page 231). When you reach the shared/callout field, follow the steps below.

To override the properties of the shared/callout field:

- 1 Enter the shared/callout field as you would enter any other shared field.
- 2 Click in the Control column and change the control from `sys_EditLive` to `sys_EditBox`.  
You can make a change like this to the properties of any system or shared field that you include in your Content Type. Your change only affects the field locally in the Content Editor; it does not change the default properties of the actual shared or system field.
- 3 Continue to add the next few fields in the table in the *Event Content Type specification* (see page 457). Stop when you reach the field `event_location` and proceed to the next topic, *Including a Child Field Set* (see page 242).

## Including a Child Field Set


At this point, we will demonstrate how to add the `event_location` child field set to our Event Content Type.

We will repeat the definition of our `event_location` field set here:

`event_location` child field set:

Name	Label	Control Name	Occur	Data Type	Format
<code>event_city</code>	Event City:	<code>sys_EditBox</code>	optional	text	50
<code>event_state</code>	Event State:	<code>sys_EditBox</code>	optional	text	50
<code>event_address</code>	Event Address:	<code>sys_TextArea</code>	optional	text	255
<code>event_contact</code>	Event Contact:	<code>sys_Edit Box</code>	optional	text	50

To add the `event_location` child field set:

- 1 Click in the first available row in the Fields table, and click .

The editor enters a field set with the Name `Child1` and the Control `sys_table` in the row. It adds a tab for `Child1` at the bottom of the editor.

If you select the row, the lower portion of the dialog says Field Set Properties instead of Field Properties and a different set of properties is displayed. Whenever the row for the field set is selected the lower portion of the dialog displays these properties.

2 Change the Name to *event\_location*. Change the Label to *Event Location*:

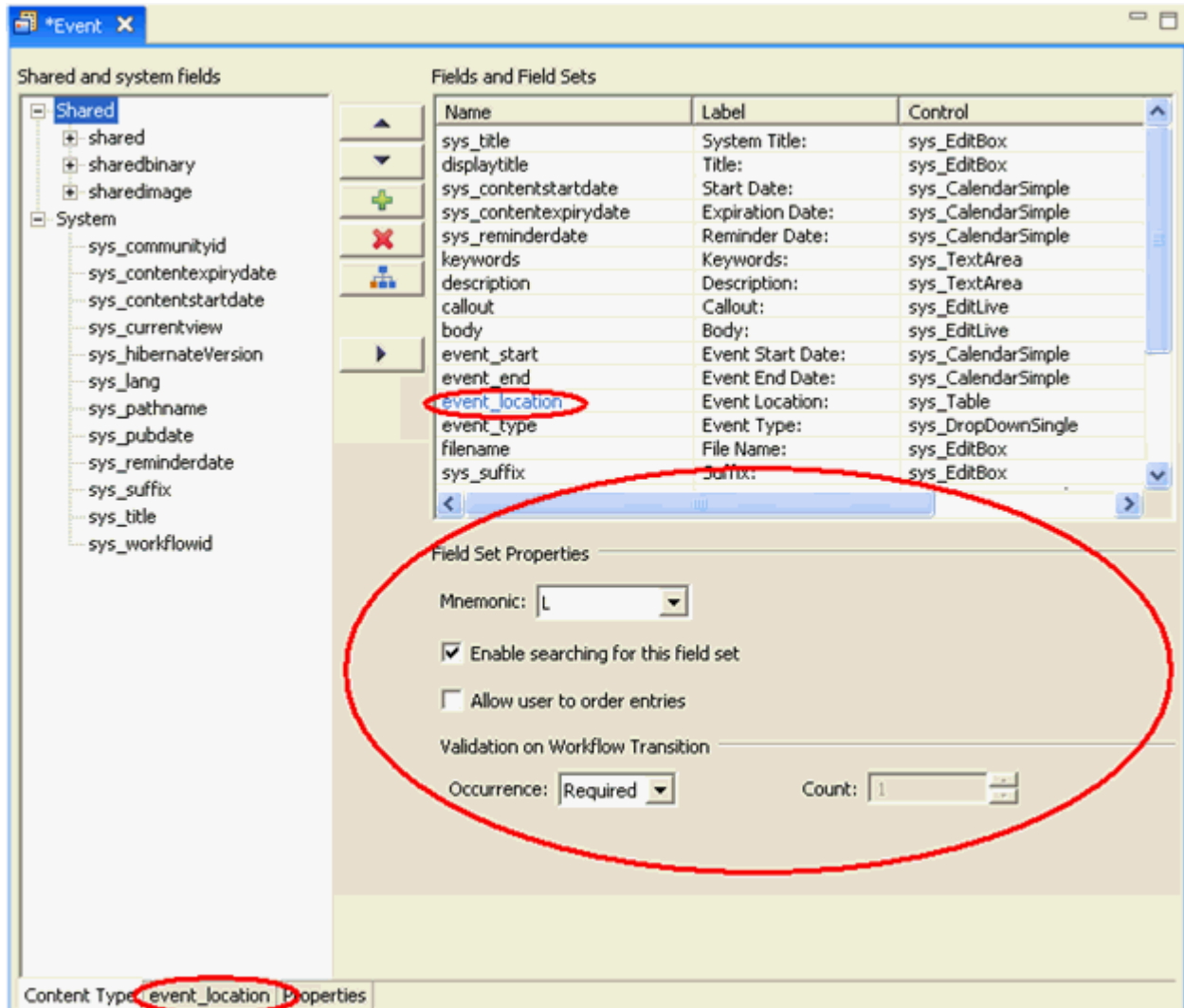


Figure 178: Event Content Type with Child Field Set Added

- 3 Under Field Set Properties:
- In Mnemonic, choose *L* since the field stores location information.
  - Leave **Enable searching for this field set** checked, since users may search for the Content Item by its location.
  - Uncheck **Allow user to order entries**, since you want the location fields to be ordered consistently in all Event Content Items.
  - Since you want users to enter at least one *event\_location* entries, choose *Required* in Occurrence.

Count is not enabled; it is only enabled if you choose *Count* in Occurrence.

- 4 Now click the *event\_location* tab to enter the fields in the child field set.

The tab is nearly identical to the Content Type tab except that it does not include the list box of shared and system fields to add to the table. Once you add an entry and click on the row, the Field Properties section appears below the Fields in event\_location table.

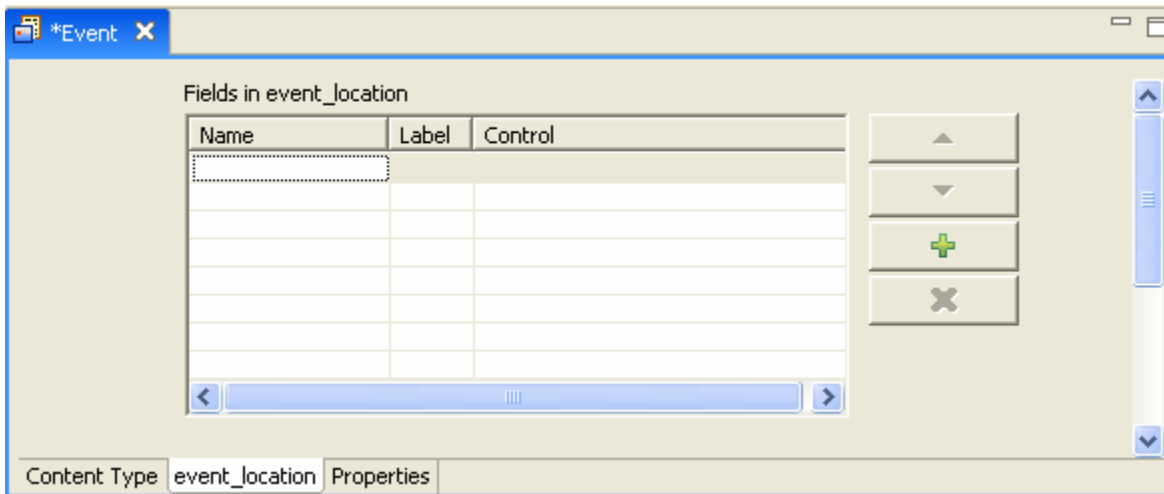


Figure 179: Child Field Set Editor

- 5 Enter the child field set entries and their properties in the table above into the Fields in event\_location table. For most properties, enter values exactly as you would enter values for local fields. For help, see *Including a Local Field* (see page 231).
- 6 For event\_address and event\_contact click [All Properties] and uncheck Show in summary. Then, click [OK] to return to the event\_location tab.

The Show in summary field is specific to entries in child field sets. If you want an entry name and its values in a child field set to display in the main Content Editor when users open it, leave Show in summary checked. If you uncheck Show in summary, users must click [Edit Table] in the Content Editor to view the entry (and all other entries in the field set) in a table. See *Event Content Editor* (see page 246) for a more detailed explanation.

- 7 Clicking the the [Validation] button take users to the same screen as for other fields. For information about filling in this screen, in the chapter *Creating Shared Field Sets*, see *Field Visibility, Validation, and Transform Rules* (see page 99). Your fields do not require you to enter any information in this screen.



You have now finished entering the child field set onto the event\_location tab. The tab should appear as:

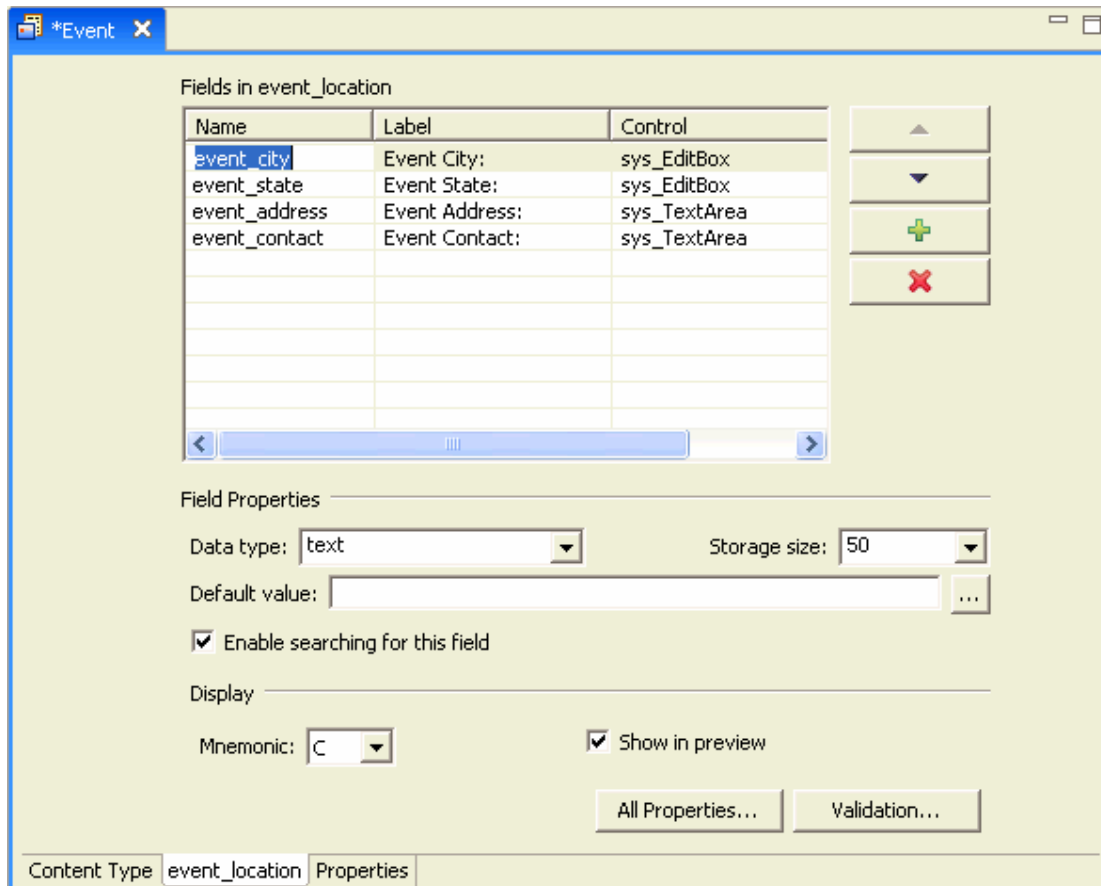


Figure 180: Entries in child field set

- 8 To finish entering the other fields in the Content Type, click the Content Type tab.
- 9 The next field that you enter is the local field event\_type. Enter this as you would enter any local field. Note that the control is sys\_DropDownSingle. You must click [...] beside the control to open the Control Properties editor to enter Choices. In the Control Properties editor, click the Use a Keyword radio button and choose FF\_Event\_Types. For information about creating the FF\_Event\_Types Keyword and adding its Keyword Choices, see *Creating and Using Keywords* (see page 273).
- 10 Add the remaining fields in the table in the *rffEvent Content Type specification* (see page 457) to the Content Type tab.

When you are done, the Content Type tab should appear similar to the graphic of the Content Type tab shown above.

- 11 Since you want to leave Enable Searching for this Content Type enabled, and the Content Type does not include any transform, validation, or pre- or post-processing extensions, you do not have to access the Properties tab of the Content Type editor. Save and close the Content Type editor.

Now, a user in Content Explorer can open the *Event Content Editor* (see page 246).

## The Event Content Editor

At least one Template must be associated with the Event Content Type to view the Event Content Editor correctly. From Assembly Design view, drag the shared rffSnTitleLink Template on top of the Event Content Type's Allowed Templates and XSL Variants folder.

Since the Event Content Type is complete, a user in Content Explorer can open the Event Content Editor. Note that the Event Location field does not display a control for entering data until you enter any required fields in the Content Editor and click **[Insert]**.

In the following graphic of the Content Editor, some fields have been entered, and the user has clicked **[Insert]** so the **[Add new item]** button is visible for the Event Location field. Since **Show in Summary** is checked for the fields *event\_city* and *event\_state* but not for the fields *event\_address* and *event\_contact*, the Content Editor displays the *event\_city* and *event\_state* fields, but the event address and event contact fields cannot be viewed until the user clicks **[Edit Table]**.

Note that the Callout field appears with the `sys_EditBox` control which you specified should override the `sys_EditLive` control.

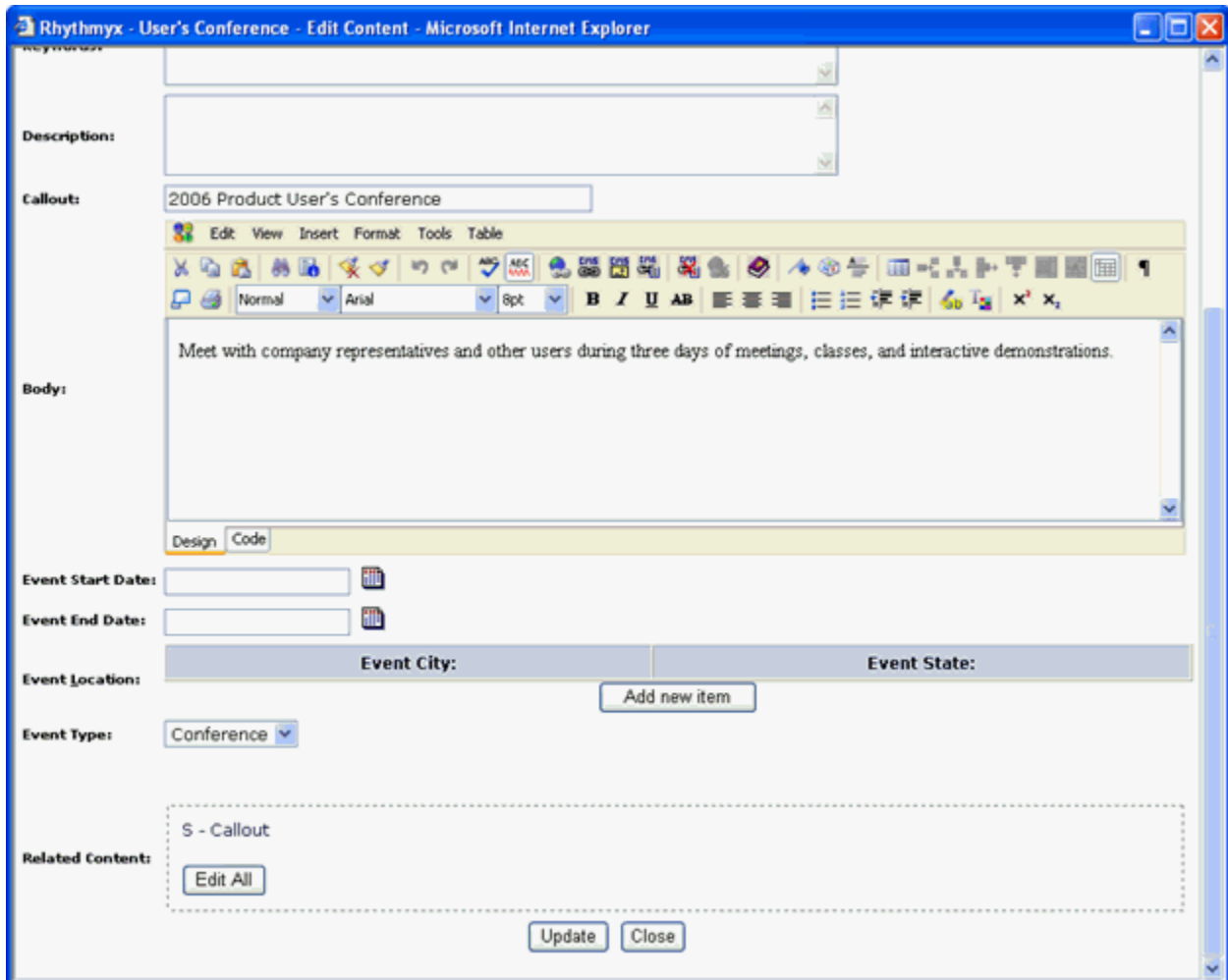


Figure 181: Event\_Test Content Editor

The user clicks [**Add new item**] to view the entire table for *Event Location*:

Figure 182: Child Table editor

After the user inserts one new location, the Content Editor provides a page for adding and editing additional ones:






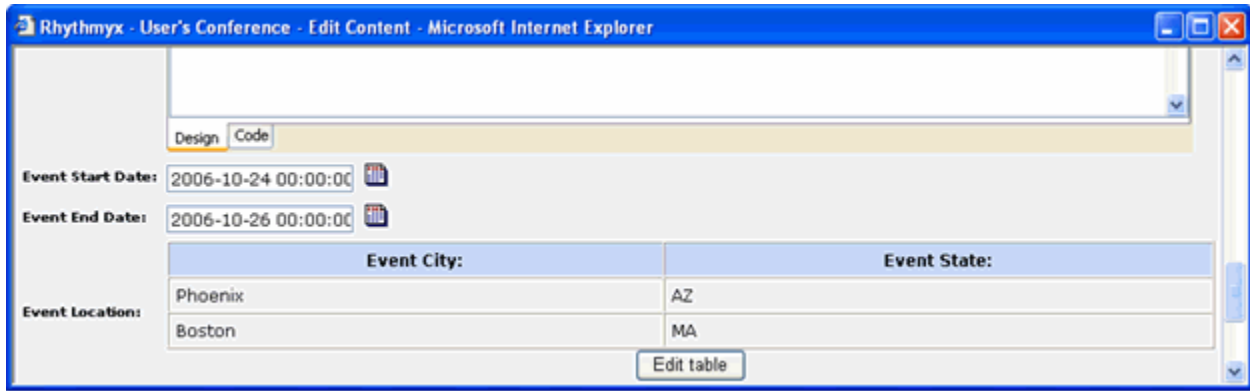
Event City:	Event State:	Action
Phoenix	AZ	  
Boston	MA	  

Figure 183: Child Table editor

The user clicks Return to parent to save the information entered and return to the parent editor, or [Close] to save the information and close the Content Editor.

In the parent Content Editor, the table appears as:



The screenshot shows a web browser window titled "Rhythmyx - User's Conference - Edit Content - Microsoft Internet Explorer". The interface includes a "Design" tab and a "Code" tab. Below these are two date pickers: "Event Start Date" set to "2006-10-24 00:00:00" and "Event End Date" set to "2006-10-26 00:00:00". The "Event Location" section contains a table with two columns: "Event City:" and "Event State:". The table has two rows of data: Phoenix, AZ and Boston, MA. An "Edit table" button is located below the table.

Event City:	Event State:
Phoenix	AZ
Boston	MA

Figure 184: Event Location table

## Viewing Event Content Items

In the chapter *Creating Slots and Templates*, in the topic *Adding Child Data to a Page Template* (see page 160) you created a Page template for our version of the Event Content Type, that displays the child field set.

The following graphic shows a preview of an Event Content Item through this template.

Enterprise Investments Home Products and Services Funds

**ENTERPRISE INVESTMENTS TO OFFER FREE "INVESTING 101: INTRODUCTION TO INVESTING" SEMINARS TO BENEFIT THE JUMPSTART COALITION**

NEW YORK, November 12, 2004 - Enterprise Investments Group, Inc. (aElou), a leading online financial services firm and the world's second-largest discount broker, today announced it will conduct a free series of "Investing 101: Introduction To Investing" seminars at customer retail outlets across the U.S. For every person attending one of these seminars, Enterprise Investments will make a donation to the JumpStart Coalition - a charitable organization that promotes financial literacy among students.

The seminars, which will cover the basics of investing and begin in mid-December, will be presented at all of the more than over 170 Enterprise Investments customer retail outlets nationwide, as well as other select locations. They are open to all adults and children over the age of 14.

To kick-off this effort, Frank J. Peter, President and Chief Operating Officer of Enterprise Investments, will make the first presentation of the "Investing 101: Introduction to Investing" seminar on December 23, 2004, at 6:00 PM EST, in New York City at the Waldorf Astoria Hotel. A total of twenty seminars will be held that week in major markets around the country.

"Enterprise Investments is presenting a summer school session that everyone will want to attend," says Mr. Peter. "You're never too young or too old to take control of your financial future. At times, getting started is the most difficult step. With these educational seminars, we want to empower individuals to make their own choices and acquaint them with the basics of investing."

"Additionally, Enterprise Investments will make a contribution to the JumpStart Coalition for every student that attends a seminar. This is a tremendous opportunity for us to invest in the future of America's young men and women."

These presentations will include basic investment principles and how to assess which types of investment instruments may be appropriate for individual investors and their particular situation. Furthermore, investment services available through Enterprise Investments will be examined.

In conjunction with the seminars, Enterprise Investments Associates are now listed as speaker resources to appear at schools nationwide through JumpStart's Educational Clearinghouse. You can gain access to this resource database at [www.jumpstart.org](http://www.jumpstart.org).

To reserve a seat at an "Investing 101: Introduction to Investing" seminar, please visit our website, [www.EnterpriseInvestmentsBankingOnline.com](http://www.EnterpriseInvestmentsBankingOnline.com), and click on Introduction to Investing Seminar to locate the seminar nearest you, or call 800-934-4448.

Enterprise Investments, (aElou), provides investors with a broad range of brokerage, mutual fund, banking and other consumer financial products on an integrated basis. In the United States, Enterprise Investments is the only online discount broker to have a retail outlet in all 50 states. Worldwide, Enterprise Investments currently services 3.8 million customer accounts in the United States, Canada, the United Kingdom, Australia, Hong Kong, Japan and India. Enterprise Investments can be found on the Internet at [www.EnterpriseInvestmentsBankingOnline.com](http://www.EnterpriseInvestmentsBankingOnline.com) and on America Online at Keyword: Enterprise Investments.

The JumpStart Coalition for Financial Literacy's purpose is to evaluate the financial literacy of young people; develop, disseminate, and encourage the use of guidelines for grades K-12; and promote the teaching of personal finance.

The JumpStart Coalition believes that all young people need to have the financial literacy necessary to make informed financial decisions.

**Details**

<b>Start:</b>	12.23.2004
<b>End:</b>	12.23.2004
<b>Type:</b>	Conference

**Locations**

216 Granite Avenue , Bangor , Maine	contact: Lisa Kerr
24816 Camino Real Way , Monterey , California	contact: Ed Wong
4873 Brazos Road , Bryan , Texas	contact: Rita Perez
1587 Wasburn Road , Topeka , Kansas	contact: Kent Hoyt

Figure 185: Preview of Event Content Item showing child data table

---

## Item Transformation, Validation, and Pre- and Post-Processing

An item transform or validation is an extension that operates on multiple fields in a Content Item.

- Item input transformers - Modifies or reformats data in a Content Editor field or fields before it is uploaded to the Rhythmyx repository. For example, if City and State fields are filled in, an item input transform could use them as keys for reading an external file and then filling in a Zip Code field. Item input transformers run after field transformers but before generic pre-processing extensions.
- Item output transformer - Modifies or reformats data in the Rhythmyx repository after it is retrieved from the Repository and before it is displayed or assembled. For example, if the Content Editor includes an expiration date, an item output transform could compare the expiration date to the current date and enter a value in a Days Remaining field. Item output transformers run after field output transformers but before generic post-processing extensions.
- Item validation - Runs during Transitions (including check in and check out) to confirm that data entered into one or more Content Editor fields meets specified criteria. If any validations fail, Rhythmyx displays an error message in the Content Editor and prevents further processing of the Content Item until the error is corrected. For example, an item validation could require that a start date precedes an end date. Note that in many cases you can use field validations to achieve the same goal. If you have to run multiple field validations, running an Item validation instead may result in improved system performance.

Pre-processing and Post-processing extensions perform more generic item processing.

- A pre-processing extension performs processing on a Content Item after item input transformers but before the Content Item (or data lookup document) is created. For example, the `sys_imageInfoExtractor` extension is a pre-processing extension that extracts and inserts data into fields before the system updates a new Content Item to the Repository.
- A post-processing extension performs processing on a Content Item after a data is retrieved from the Repository. For example, the `sys_ceDependencyTree` extension adds child and parent items of the Content Item to the result XML document so that users can view the Content Item's relationships using the Impact Analysis option.

The tabs for adding item input and output transforms, validations, and pre- and post-processing extensions on the Properties tab of the Content Type editor are nearly identical. We could demonstrate how to add any of these, and the process would be nearly the same. For details about adding the extensions that we do not demonstrate here, see the corresponding topic in the *Rhythmyx Workbench Online Help*.

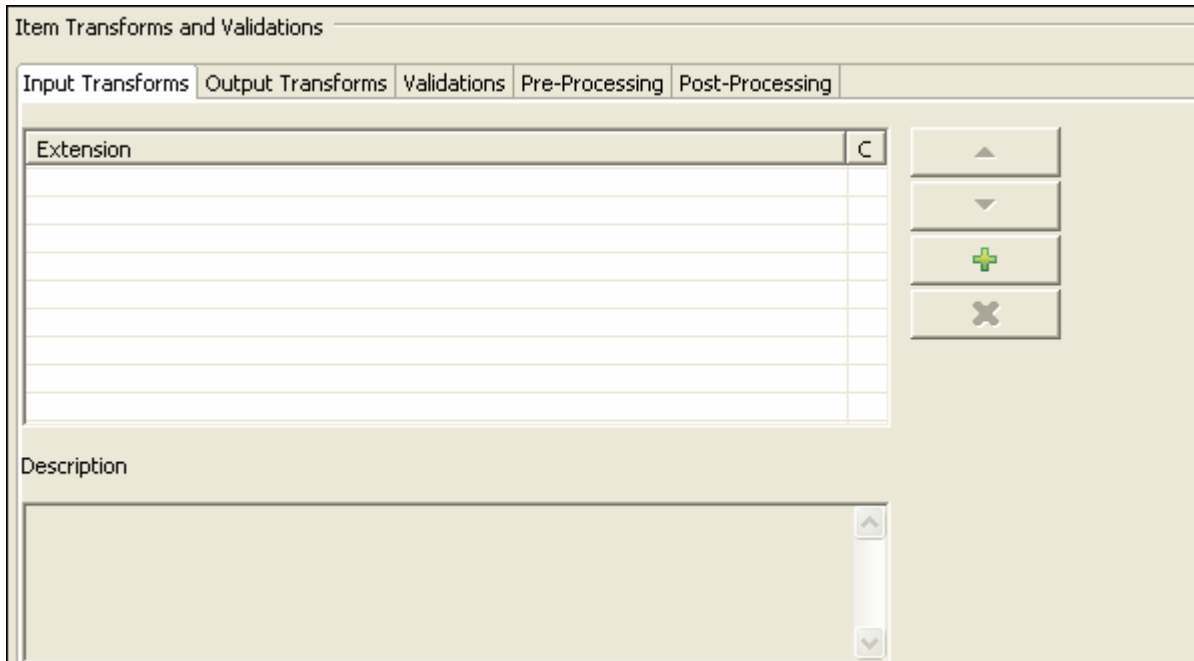


Figure 186: Item Transforms and Validations section

**Adding Pre-processing Extensions** (see page 251), shows you how to add the pre-processing extensions that are typically added to Image Content Types.

## Adding Pre-processing Extensions

Here, you will add the following three item pre-processing extensions to your Image Content Type. These extensions are commonly included in image Content Types:

- `sys_imageInfoExtractor` - Extracts an uploaded image's metadata and inserts it into fields in the Content Editor
- `sys_CopyParameter` - Copies the value of a source parameter into the destination parameter. This is used twice.
  - In the first instance, the source parameter is the value that `sys_imageInfoExtractor` stores in `img1_filename` (or in our case, `img1_filename`). It is copied into the content editor's `shared/filename` field.
  - In the second instance, the source parameter is the value that `sys_imageInfoExtractor` stores in `img1_ext` (or in our case, `img1_ext`). It is copied into the content editor's `sys_suffix` field.

---

`sys_CopyParameter` copies values into fields that are used in FastForward's default location scheme. Location Schemes are discussed in the chapter *Configuring Publishing* (see page 291).

---

To add pre-processing extensions to your Image Content Type:

- 1 Open the Image Content Type.
- 2 In the Content Type editor's Properties tab, click the Pre-processing tab.
- 3 In the **Extension** table, click in the first row to activate a drop list of pre-processing extensions.
- 4 Choose `sys_imageInfoExtractor`. This extension does not require any parameters.
- 5 In the **Extension** table, click in the next row to activate the drop list.
- 6 Choose `sys_CopyParameter`.
- 7 Click [...] beside the extension name to open the Parameters dialog.  
The parameters *source* and *destination* are listed in the first two rows of the **Name** column.
- 8 Beside *source* parameter, enter *img1\_filename* under **Value**.
- 9 Beside *destination* parameter, enter *filename* under **Value**.

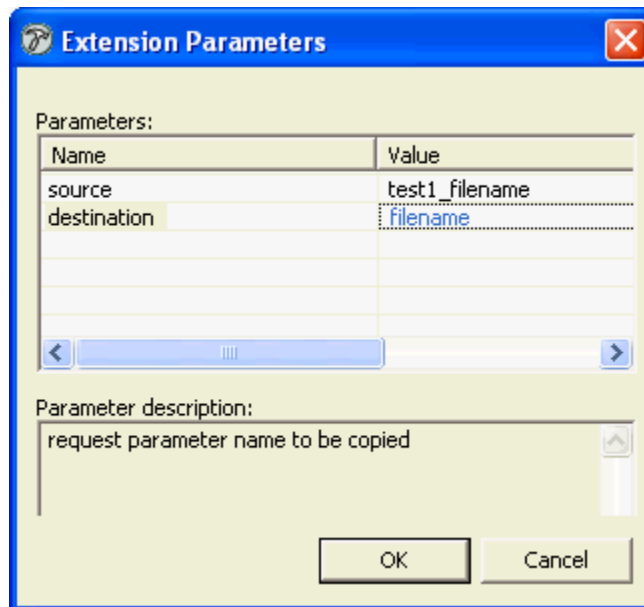


Figure 187: Extension Parameters dialog

- 10 Click **[OK]**.
- 11 In the **Extension** table, click in the next row to activate the drop list.
- 12 Choose `sys_CopyParameter` again.
- 13 Click [...] beside the extension name to open the Parameters dialog.  
The parameters *source* and *destination* are listed in the first two rows of the **Name** column.
- 14 Beside the *source* parameter, enter *img1\_ext* under **Value**.
- 15 Beside the *destination* parameter, enter *sys\_suffix* under **Value**.
- 16 Click **[OK]**.
- 17 The Conditional Property dialog closes.



The pre-processing extensions are now entered. The Pre-processing tab should appear as:

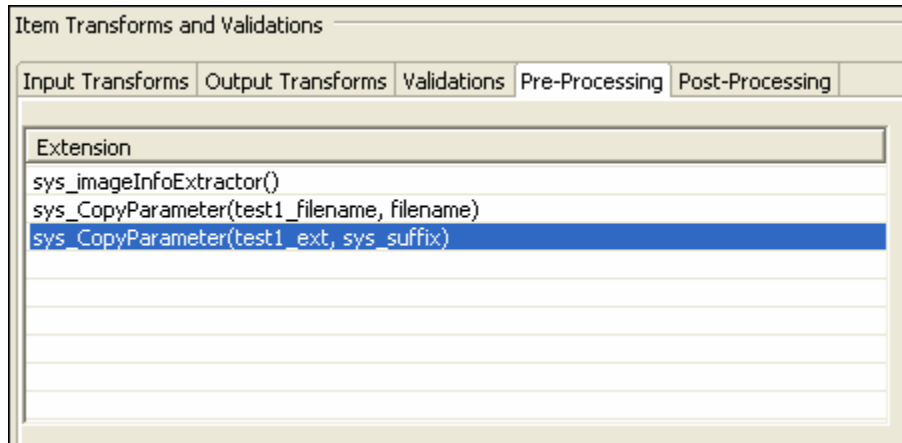


Figure 188: Preprocessing Tab

## Implementing Text Extraction

Text extraction is a Rhythmyx feature that provides the capability of pulling text from binary files created in third-party applications (such as Microsoft Word documents or .pdf files) and inserting it into a Rhythmyx Content Editor. A pre-processing extension, `sys_textExtraction`, extracts the text as either simple text with no formatting or as HTML, and inserts the text into a field in the Rhythmyx Content Editor. You can develop and add more extensions to transform the text or parse it into different Rhythmyx fields.

The text extraction feature uses the functionality of the Rhythmyx full-text search engine. The search engine must be installed before you can use text extraction, but it need not be enabled. For additional information about the full-text search engine, see:

You can use text extraction to upload files individually or you can perform bulk conversions by defining a WebDAV-enabled folder where users can add files. For additional information about WebDAV, see [Implementing WebDAV in Rhythmyx](#). Bulk conversion can be used either to facilitate migration of content into Rhythmyx or to allow continuous update of extracted content by uploading modified files.

The `sys_textExtraction` extension does not limit the size of text extracted from the binary files, but settings in the Repository database, JDBC driver, Content Editor field, or users browser may limit the amount of text that can be uploaded. If an error occurs during text extraction because the amount of text exceeds some limit, these settings should be checked.

## Creating a Text Extraction Content Type

To illustrate the creation of a Content Type that uses text extraction, we will create a Bio Content Type to store the biographies of executives at Enterprise Investments and Corporate Investments. The Bio Content Type includes the following fields:

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	sharedbinary/item_file_attachment	File:	sys_file	required	binary	max
shared	sharedbinary/item_file_attachment_filename	Binary File Name:	sys_EditBox	required	text	512
shared	sharedbinary/item_file_attachment_size	File Size:	sys_EditBox	optional	integer	none

Source	Name	Label	Control Name	Occur	Data Type	Format
shared	sharedbinary/item_file_attachment_type	File Type:	sys_EditBox	required	text	256
shared	sharedbinary/item_file_attachment_ext	Extension	sys_EditBox	required	text	50
shared	shared/body	Body:	sys_EditLive	optional	text	max
shared	shared/webdavowner	WebDAV Owner:	sys_EditBox	optional	text	256
local	extraction_error	Extraction Error	sys_EditBox	optional	text	256

For details about adding fields, see "*Basic Content Type Creation* (see page 229)" and "*Image Content Type Creation* (see page 229)".

Once you define the fields, you must add the `sys_textExtraction` post-processing extension:

- 1 On the Content Type editor, choose the Properties tab.
- 2 Under Item Transforms and Validations, choose the Pre-Processing tab.
- 3 Double-click in the first empty row and from the drop list, choose the `sys_textExtraction` extension.

Rhythmyx displays the Extension Parameters dialog with a list of parameters for the `sys_textExtraction` extension.

- 4 Enter values for the extensions. (NOTE: The values in the following table assume that the Content Type uses the fields defined above. Substitute the correct values for your own implementation.)

Name	Description	Value
Source	<p>Required.</p> <p>Specifies the file containing the data to extract. If the value of this parameter is a File object, text is extracted from that file. Otherwise, the value is used to construct a path to a file. In general, the value of this parameter is populated by a file upload control on the Content Editor.</p> <p>NOTE: Binary fields are not available as parameter values as Content Item Fields (PSXContentItemData). You must specify the name of the field as a PSXSingleHTMLParameter.</p>	PSXSingleHTMLParameter/item_file_attachment
OutputParam	<p>Required.</p> <p>The name of the Content Type field in which the extracted content will be stored.</p> <p>NOTE: Large text fields that are treated as binary are not available as parameter values as Content Item fields (PSXContentItemData). You must specify the name of the field as a PSXSingleHTMLParameter.</p>	body
FileTypeParam	<p>Optional.</p> <p>The name of the Content Type field in which the file type identified for use in the extraction will be stored as text. This parameter allows the Content Editor store the file time in a Content Item field.</p>	item_file_attachment_type

Name	Description	Value
ErrorMessageParam	<p>Optional.</p> <p>The name of the Content Type field in which any error messages are to be stored as text. If this parameter does not have a value, the extension will display any error messages to the user in the Content Editor. If the parameter has a value, any errors encountered will be written to the specified field and the exit will return silently.</p> <p>Typically, if the files will be uploaded manually, no value will be specified for this parameter so the user can see and respond to any errors that occur. If the files will be uploaded in bulk, this field typically has a value so processing will not be interrupted when a processing error occurs.</p>	extraction_error
OutputFormat	<p>Optional.</p> <p>Specifies the output format of the text extracted. Valid values are:</p> <ul style="list-style-type: none"><li>▪ HTML: text is using HTML to produce formatting that matches as closely as possible to the input data.</li><li>▪ TEXT: text is output as plain text, with some whitespace formatting.</li></ul> <p>Defaults to TEXT if not specified.</p>	HTML

<b>Name</b>	<b>Description</b>	<b>Value</b>
UseLinefeed	<p>Optional.</p> <p>If the value of this parameter is <i>Y</i>, line endings are converted to line feeds in the output text. If the value is null or any value other than <i>Y</i>, line endings are converted to carriage returns in the output text.</p> <p>This parameter is ignored if the value of the <code>OutputFormat</code> parameter is <i>HTML</i>.</p>	
HTMLTemplate	<p>Optional.</p> <p>Specifies the path to the template to use when extracting text to HTML. The template file must be stored under the Rhythmyx installation root directory; the path must be specified relative to that directory.</p> <p>If no value is specified for this parameter, the default system template is used.</p> <p>This parameter is ignored if the value of the <code>OutputFormat</code> parameter is <i>TEXT</i>.</p>	

Name	Description	Value
OutputEncoding	<p>Optional.</p> <p>Specifies the character encoding to use for the text output.</p> <p>If no value is specified for this parameter, the text is output in WINDOWS-1252 if the value of the OutputFormat parameter is TEXT; it is output in UTF-8 if the value of the OutputFormat parameter is HTML.</p> <p>The following encodings are also valid:</p> <ul style="list-style-type: none"><li>▪ Shift_JIS (Japanese)</li><li>▪ EUC_KR (Korean)</li><li>▪ GB2312 (Simple Chinese)</li><li>▪ Big5 (Traditional Chinese)</li></ul> <p>If the input text includes characters outside of the specified character set, that text may be lost.</p>	

Name	Description	Value
PDFConversion	<p>Optional.</p> <p>Specifies how to perform the extraction if the input file type is .pdf. (If the input file is not .pdf, this parameter is ignored.)</p> <p>If the value of this parameter is SINGLE (or if no value is specified):</p> <ul style="list-style-type: none"> <li>▪ Multi-byte characters in the input data are not processed.</li> <li>▪ Text is output using the system default character set.</li> <li>▪ Multiple columns in source data are processed.</li> <li>▪ The value in the OutputFormat parameter is ignored; text is always output as if the value of the OutputFormat parameter were <i>TEXT</i>.</li> </ul> <p>If the value of this parameter is MULTI:</p> <ul style="list-style-type: none"> <li>▪ Multi-byte characters in input data are processed.</li> <li>▪ If the OutputEncoding parameter has a value, text is output using that encoding.</li> <li>▪ Multiple columns in source data are not processed.</li> <li>▪ The text is output as specified by the OutputFormat parameter.</li> </ul>	MULTI

**5** Click the [OK] button to save the parameter specifications.

**6** Save the Content Editor.



## CHAPTER 8

# Managed Navigation



Seamless and intuitive navigation through a web site promotes a successful interaction for site visitors. Site navigation is typically comprised of a combination of the following navigation elements:

- top navigation bar
- side navigation
- bottom navigation
- breadcrumbs
- a Site map

The screenshot shows the Enterprise Investments website with several navigation elements highlighted:

- Top Navigation:** A horizontal menu at the top with links for 'About Enterprise Investments', 'Investment Advice', 'Mortgages & Home Finance', and 'Products & Services'.
- Breadcrumbs:** A path of links at the top of the main content area: 'Enterprise Investments > Home > Enterprise Investments'.
- Left Navigation:** A vertical menu on the left side with categories like 'About Enterprise Investments', 'Investment Advice', 'Mortgages and Home Finance', and 'Products and Services'.
- Bottom Navigation:** A horizontal menu at the bottom of the page, identical to the top navigation bar.

Figure 189: Press Release with Side Navigation Menu and Breadcrumbs

Effective employment of these elements adds to a site's ease of use.

The About Corporate Investments Generic page Content Item, for example, contains a top navigation menu, a left navigation menu, breadcrumbs, and a bottom navigation menu. The Rhythmyx Managed Navigation system automatically generates these elements during assembly.

Note that Managed Navigation relies on Site Folder Publishing to deliver the Items organized with a Site's set of folders. Managed Navigation will not function if you have not used a Site Folder to structure your content.

---

# How Managed Navigation Works

Managed Navigation is based on three specially-designed navigation Content Types:

- Navon

Navons are the basic unit of Managed Navigation, and are used to create navigation menus. Each Navon should be linked to a Landing Page, which is a Content Item in its Folder not used for Navigation (such as a Generic Content Item or a Category Content Item) where users will land when they click on the Navon in the navigation.. A Navon may also be associated with a NavImage (see below). A Navon has three essential attributes:

- Label

The Label is the source of the "clickable" text for the output.

- Landing Page

The Content Item that generates the HTML page the user jumps to when clicking on the Managed Navigation. The Landing Page can also be an external URL.

- Image Link

The Image Link specifies an image that can be used to represent the Navon in the output.

- NavTree

Similar to a Navon, NavTree Content Items reside at the root of a Site and define the root of the navigation structure for the Site. NavTree Content Items propagate Navon's to each Subfolder created in the Site Tree. The NavTree Item is generally linked to the Site's Home Page Item.

NavTrees also have the Label, Landing Page, and Image Link attributes of a Navon.

- NavImage

Images used by Navons to replace text links for navigation elements.

Most Folders in a Site will contain at least one navigation Content Item, either a NavTree Content Item if it is the root Folder, otherwise a Navon). A Folder may also contain one or more NavImage Content Items.

During assembly, Rhythmyx uses NavTrees and Navons to build an XML document that represents the Folder hierarchy of the Site. The XML document defines

- the Site's hierarchy;
- the location of each Navigation Content Item within that hierarchy;
- The relationship between an individual Managed Navigation Content Item to the other Managed Navigation Content Items in the Site; and
- whether an image is associated with the Navigation Content Item.

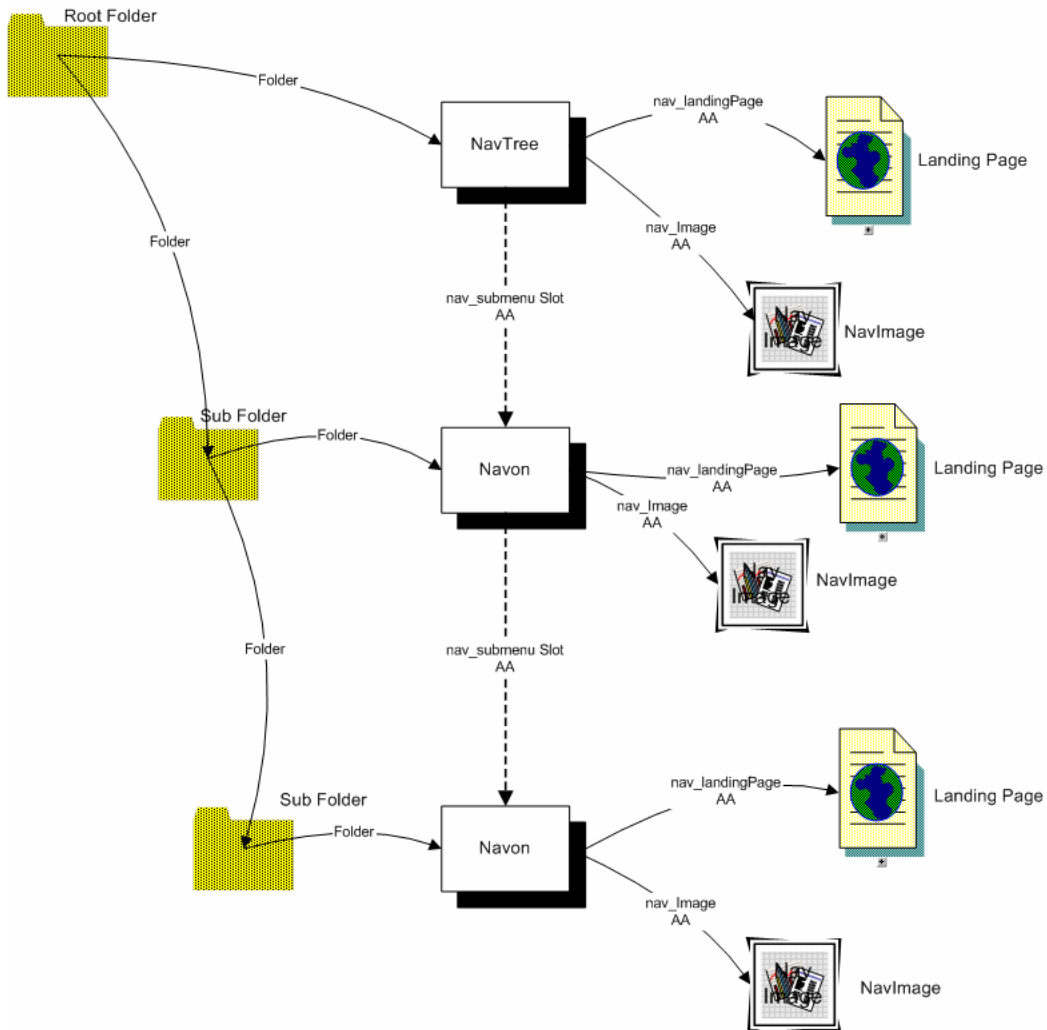


Figure 190: Relationships between Navigational Elements

Every node in the XML document is categorized by its relationship to the node where the assembly starts, which is called its *axis*. The axis can take one of the following values:

- Root
- Ancestor
- Ancestor-Sibling
- Sibling
- Self

- Descendent
- Other

The following diagram illustrates an example tree showing the axis of each node in relation to a selected node:

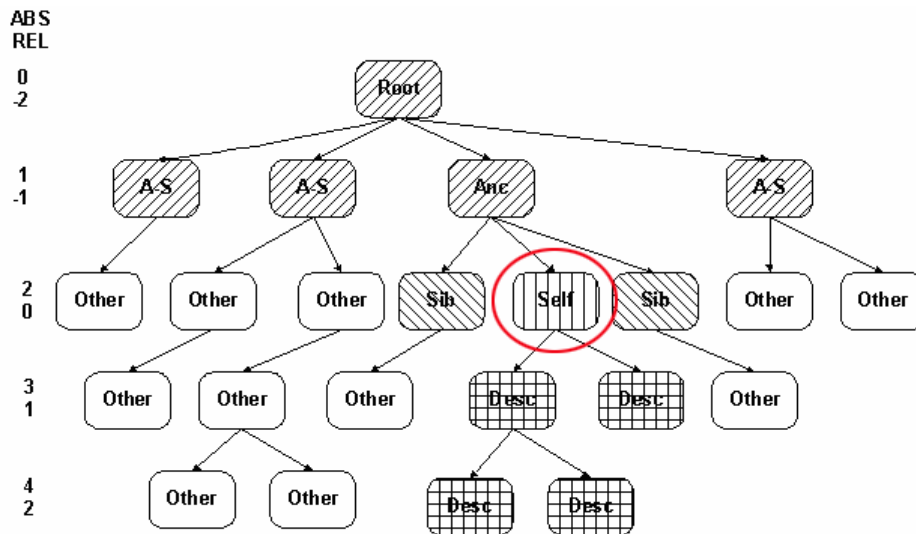


Figure 191: Navigation Relationships

The node where the assembly of the Item starts is the SELF node. All predecessors of the SELF node in the path to the root are known as ANCESTOR nodes. (The immediate predecessor in this path is referred to as the PARENT) Other nodes that share the same PARENT as the SELF node are SIBLING nodes. Nodes that share the same PARENT as an ANCESTOR are ANCESTOR-SIBLING nodes. The top node of the tree is the ROOT node. All nodes in the path branching from the SELF node are DESCENDANT nodes.

Typically, only the nodes discussed are represented in a navigation bar. Any node that does not share one of the associations described is designated an OTHER node and is not used in the navigation bar.

A special case occurs when the SELF node is the same as the ROOT node (as in a Site Map Template, for example). In this situation, only two axes are used: SELF and DESCENDENT.

The different levels in the tree are accounted for in two ways: Absolute level and Relative level. The Absolute level starts at the ROOT (0) and increases as a positive integer as the tree grows downward. The Relative level starts at the SELF node (0). This level is counted as a negative integer (-1, -2, and so on) along the path of ancestors to the ROOT and as a positive integer along the path of DESCENDANTS from the SELF node. These levels are used by rendering Templates to control the number of levels displayed and the styles used to render the different levels in the navigation bars.

Managed Navigation Templates process the Managed Navigation XML document to produce an HTML output that is merged with a Cascading Stylesheet to produce the final formatted navigation content for the page. The Breadcrumb Template (rffNavBreadcrumb), for example, renders a relatively format-free HTML document when previewed separately.

[Home](#) > [News & Events](#) > [Press Releases](#)

Figure 192: Breadcrumbs Without Formatting

When integrated into a Page, this Template inherits the look and feel of the page that includes it:



*Figure 193: Page with Breadcrumbs*

---

# Maintaining Managed Navigation Content Items

In the chapter *Setting Up the Publishing Site and Basic Navigation* (on page 61), we described how to add a NavTree to a Site root Folder and how Navons propagate. To ensure that Managed Navigation works properly, however, Navigation Content Items require some maintenance. We also need to address how to create NavImage Content Items and how to associate them with Navons. Finally, we need to address how to maintain Managed Navigation when your Site structure changes.

## Navigation Communities

Typically, Managed Navigation Content Items are maintained by the users responsible for administering Rhythmyx content, such as a Web Master or site producer, rather than by content contributors. To isolate Managed Navigation content, an administrative Community is typically implemented for each Site, in addition to the Site's content contributor Community. Managed Navigation Content Types are assigned to the admin Community, while non-navigation Content Types are assigned to the Site's content contributor Community.

For example, the FastForward implementation includes the following administrative Communities:

- Enterprise Investments Admin
- Corporate Investments Admin

The Managed Navigation Content Types are associated with these Communities. The EI\_Admin Role is assigned to the Enterprise Investments Admin Community and the CI\_Admin Role is assigned to the Corporate Investments Admin Community. A user that is a Member of the EI\_Admin Role can log in to the Enterprise Investments Admin Community and maintain the navigation content for that Site.

## Assigning a Landing Page to a Navon

Navons represent the Folder in which they reside in any piece of navigation (breadcrumb, top navigation, site map, etc). When a site visitor selects a link in a piece of navigation, they are directed to a particular page, referred to as the "landing page". You must manually associate a landing page with each Navon.

To assign a landing page to a Navon:

- 1 Navigate to the Navon to which you want to add the landing page.
- 2 Right click the Item and from the popup menu choose *Active Assembly Table Editor*. (If the Navon is already Public, you may have to Transition it to the Quick Edit State before this option is available.)

Rhythmyx displays the Active Assembly Table Editor for the Navon.

- 3 Click on the nav\_landing\_page link.

Rhythmyx displays the Active Assembly Search dialog.

- 4 Find the Content Item you want to assign as the landing page for the Navon, check the box for that Content Item, and click the **[Link to Slot]** button.
- 5 Close the Active Assembly Table Editor
- 6 If the Navon was edited in a Public State, Transition it back to the Public State.

## Creating a NavImage

NavImage Content Items support the use of images in navigation. If you want to represent a section of your site with an image, you must create a NavImage Content Item and include it as related content in the `nav_image` Slot on the NavTree or Navon Content Item. It is usually easier to find a NavImage if it resides in the same Site Folder as the Navon or NavTree that uses it, but it is not required to reside there.

A NavImage Content Item requires an Image file. You must assign this file to the NavImage.

If any given navigation element uses text links instead of images, you do not need to create a NavImage for it.

To create a NavImage

- 1 Log into the Content Explorer.
- 2 Locate the Site Folder where you want to create the NavImage.
- 3 Right click the Site Folder and from the popup menu, choose *New Item > NavImage*.  
Rhythmyx displays the NavImage Content Editor.
- 4 Fill in the fields for the new NavImage Content Item.
- 5 Insert the Item and close the Content Editor.

## Assigning a NavImage to a Navon

Once you have created a NavImage, you can assign it to a Navon. NavImages are assigned to the `nav_image` Slot on the Navon. In the default FastForward installation, NavImages are used in the `top_nav` Template to represent the Navon's Folder, and are a hotspot link to the Navon's landing page.

The `top_nav` Template is the only Navigation Template in the default FastForward implementation that supports images, but you can modify other Templates to support images as well. You could also implement new Templates to support combinations of images, text, and Flash.

Note that you can use essentially the same procedure to assign a NavImage to a NavTree.

To assign a NavImage to a Navon:

- 1 Log into the Content Explorer.
- 2 Locate the Site Folder containing the Navon to which you want to assign the NavImage.
- 3 Right-click the Navon and from the popup menu choose *Active Assembly Table Editor*. (If the Navon is already Public, you may have to Transition it to the Quick Edit State before this option is available.)  
Rhythmyx displays the Active Assembly Table Editor for the Navon.
- 4 Click on the `nav_image` link.  
Rhythmyx displays the Active Assembly Search dialog.



- 5 Find the NavImage Content Item you want to assign to the Navon. check the box for that NavImage, and click the [Link to Slot] button.
- 6 Close the Active Assembly Table Editor
- 7 If the Navon was edited in a Public State, Workflow the Item back to the Public State.

## Splitting Navigation Sections

As you add content to your Site, you may find that some Folders contain so many Content Items that they become unwieldy. In that situation, you may want to subdivide one Folder into several Subfolders. You may also want to subdivide a Folder to accommodate marketing needs, such as spinning off a new product line, or simply to make it easier for Content Contributors to manage the content assigned to them.

When you split a Folder, you can represent it as one or more subsections in the Site:

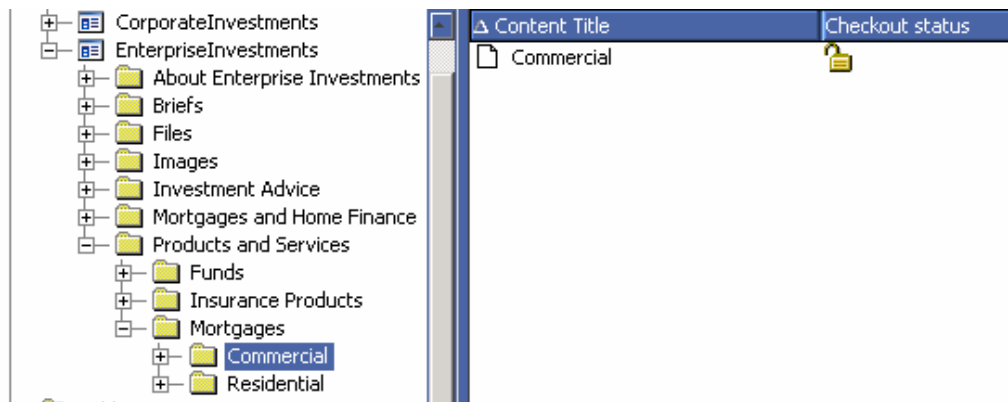


Figure 194: Splitting the Navigation Section

You can also choose to use a single Navigation Item to represent Content Items in multiple sub folders.

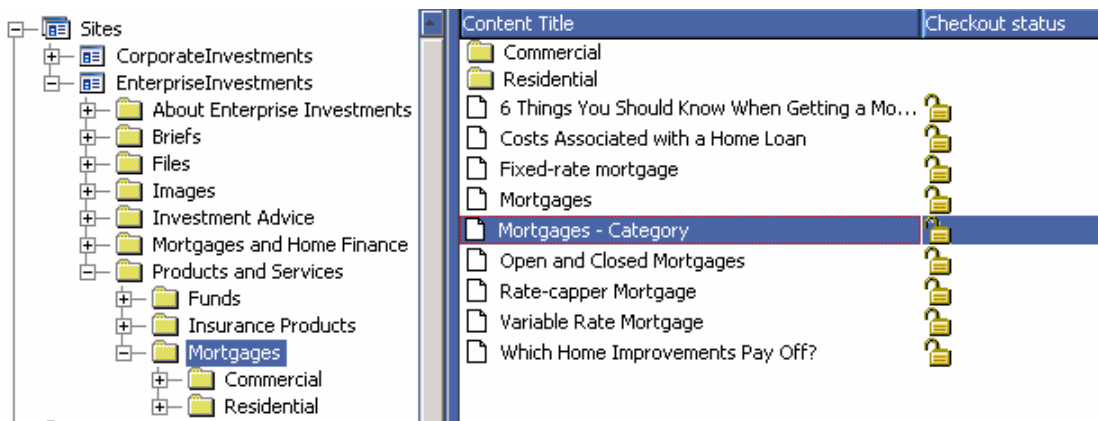


Figure 195: Representing Two Folders with One Navigation Item

Let us follow an example illustrating the first scenario. Assume we have decided to subdivide the current Mortgages section of our Web site into two new sections, Commercial Mortgages and Residential Mortgages. We will maintain the original Mortgages Site Folder and nest the two new sections within that folder.

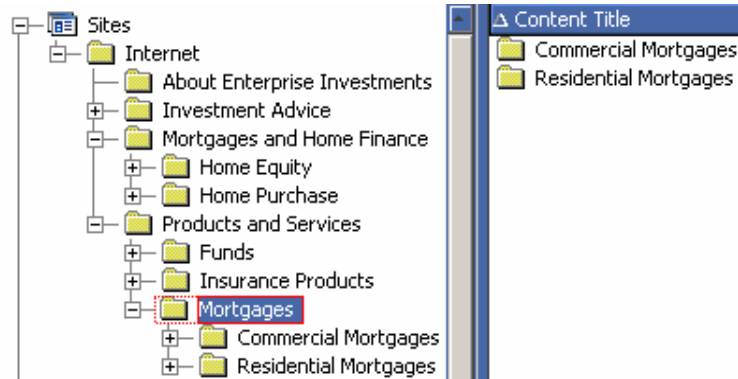


Figure 196: Site Folder without a Navon

The Mortgages Folder, though, will not itself be represented in the navigation.



Figure 197: Navigation without Split Sections

To split a Site Folder:

- 1 Log into Content Explorer and find the Site section you want to Split (Mortgages in this case).
- 2 Create the necessary sub folders as descendants of the original Site Folder.  
 In our example, we will create two new Folders, Commercial Mortgages and Residential Mortgages. Note that when we create these Folders, a Navon is added to each of them automatically.
- 3 Delete the Navon Item from the original Folder (from the Mortgages Folder in this case). Also delete or move any other Navigation Content Items, such as any NavImage Content Items in the Folder.
- 4 Drag and [Move] the Content Items from the Original Folder to the new descendant Folders.

- 5 Assign landing pages to the nav\_landingpage Slot of each new Navon. You may need to modify the Content Item assigned as the landing page to reflect the new contents of the Folder.
- 6 Create any new NavImage Items for the new Navons and assign them to the appropriate nav\_image Slot.

At this point, you should have the original Site Folder (Mortgages) with no currently associated Content Items. The Mortgages Folder contains two Subfolders, Commercial Mortgages and Residential Mortgages. Each of these Folders contains a single Navon and several Content Items, and possibly one or more new NavImage Content Items.

We want the navigation to skip the Mortgages Folder. We will have to add the new Navons to the nav\_submenu of Navon in the Folder that contains the Mortgages Folder (which is Products and Services in our example).

- 7 Open the Products and Services Folder and locate its Navon.
- 8 Right-click on the Navon and from the popup menu choose *Active Assembly Table Editor*. Rhythmyx displays the Active Assembly Table Editor.
- 9 Click the nav\_submenu link and search for Content Items with the word "mortgages" in the title.
- 10 Check the boxes for the Commercial Mortgages and Residential Mortgages Navons, then click the **[Link to Slot]** button.

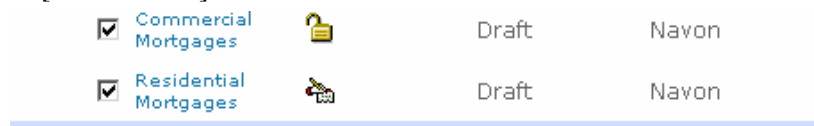


Figure 198: Adding Navon Items to the nav\_submenu Slot

- 11 Close the Active Assembly Table Editor.
- 12 Transition the new Navons the Public State.
- 13 Reset the navigation.
- 14 Preview the landing pages in each new descendant section. The navigation should not show the old Mortgages Section, but instead display each of the new descendant sections.

<b>About Enterprise Investments</b>
<b>Investment Advice</b>
<ul style="list-style-type: none"> <li>• Insurance Advice</li> <li>• Estate Planning</li> <li>• Retirement</li> <li>• Tax</li> </ul>
<b>Mortgages and Home Finance</b>
<ul style="list-style-type: none"> <li>• Home Purchase</li> <li>• Home Equity</li> </ul>
<b>Products and Services</b>
<ul style="list-style-type: none"> <li>• Funds</li> <li>• Insurance Products</li> <li>• Commercial Mortgages</li> <li>• Residential Mortgages</li> </ul>

Figure 199: Navigation with New Split Sections

## Merging Navigation Items

It is no less common to merge sections of a Web site than to split them. When you merge several Folders, you also need to merge their Navigation Content Items as well. For example, originally, we organized Press Releases by year:

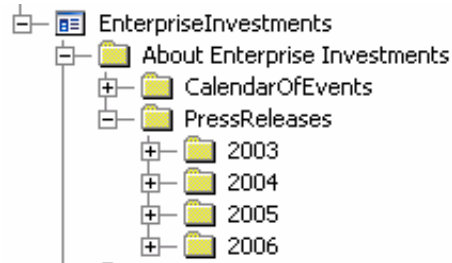


Figure 200: Press Releases by Year

After gathering five years worth of Items, we decided that any Press Release two years or older would be managed in an Archives Site Folder.

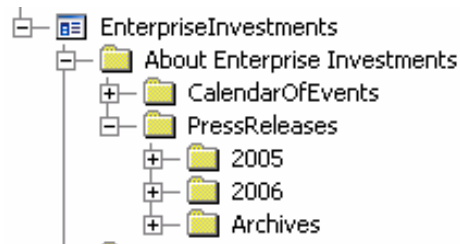


Figure 201: Press Releases by Year with Archive Folder

These Items would be represented by a single Navigation Item and sorted with an Auto Index by creation date. So we need to merge the originally separate yearly press releases Folders into the Archives Folder.

To merge Folders and navigation:

- 1 Log into the Content Explorer.
- 2 Create a new Folder to merge the existing Folders. In our example, create an Archives Folder. When the new Folder is created, a new Navon is created automatically. If desired, you can also create and associate a NavImage Content Item as well.
- 3 Move the necessary Content Items from the old Site Sections to the newly created Site Folder by selecting them, dragging them into the new Site Folder and choosing *Move* from the popup menu.
- 4 Specify a Landing page for the new Navon.
- 5 Delete the now stale Navigation Items from the old Site Folders.
- 6 Delete the now empty Site Folders.
- 7 Reset the Navigation.

## Reordering a Submenu

By default, when a Navon or NavTree Item is created, the `nav_submenu` slot is populated with links to the Navon Items in the directories immediately below the current one. These Relationships build a list of links to the Subfolders contained in a Folder. In the published output, when you choose certain navigation links (such as the left navigation), it expands to show links to the subsections. This list of sub menu Items is assembled in the order that the Folders appear in the Navigation pane of Content Explorer. However, you may want to modify this order.

To reorder a submenu:

- 1 Locate the Navon whose submenu you want to reorder.
- 2 Right-click on the Navon, and from the popup menu choose *Active Assembly Table Editor*. Rhythmyx displays the Active Assembly Table Editor.
- 3 Use the up and down arrow icons to the right of the Navon Items displayed in the `nav_submenu` Slot to adjust the order of the Navons.

Slot(ID): nav_submenu( 319)			
Item Title(ID)	Item Type(ID)	Item Variant(ID)	Action
2004( 329)	Navon( 314)	navlink( 338)	[Up/Down] [List] [Delete] [Edit]
2003( 330)	Navon( 314)	navlink( 338)	[Up/Down] [List] [Delete] [Edit]
2005( 372)	Navon( 314)	navlink( 338)	[Up/Down] [List] [Delete] [Edit]
2006 Navon( 375)	Navon( 314)	navlink( 338)	[Up/Down] [List] [Delete] [Edit]

Figure 202: Reordering Navons in the `nav_submenu` Slot

- 4 Close the Active Assembly Table Editor.

## Creating and Using Keywords

A Keyword defines a category of choices that are used in a drop list control or another selection mechanism in Rhythmyx. You define Keyword objects in the Rhythmyx Workbench and add Keyword Choices to them.

Using a Keyword Choice in a field can serve a few different purposes:

- A template can display the Keyword Choice in an output for informational purposes.
- An automated list slot can look for a specific Keyword Choice value when determining whether or not to include a Content Item.
- A custom search can look for Content Items that contain specific Keyword Choice values.

FastForward provides the `FF_Event_Types` Keyword with several choices for populating the `sys_DropDownSingle` control used with the Event Content Type's `event_type` field. Here we will demonstrate how you can create your own version of the `FF_Event_Types` Keyword and Keyword Choices.

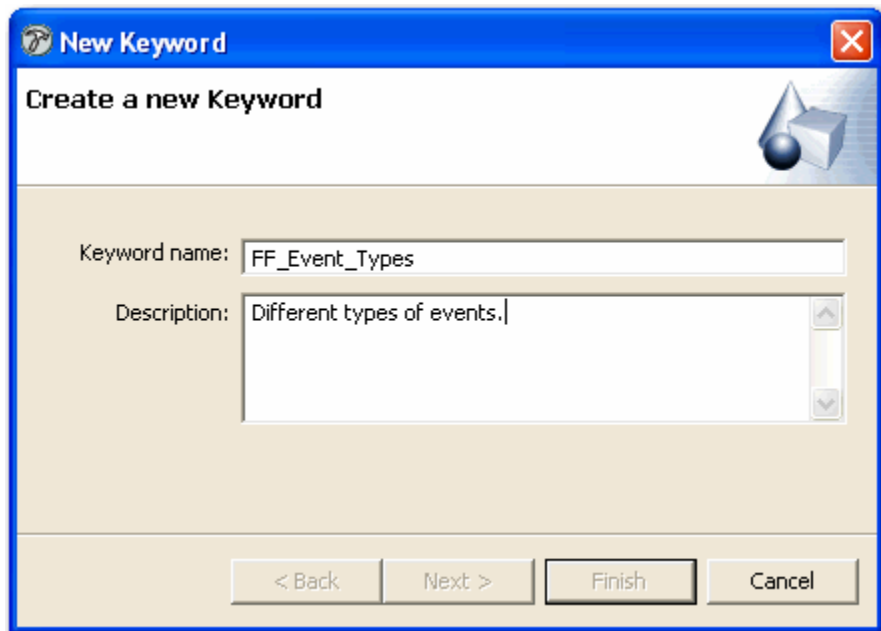
---

*Note: You cannot create a Keyword named `FF_Event_Types`, since it already exists in FastForward. Instead, create a similar Keyword included in your implementation plan or copy our steps but give your Keyword a different name.*

---

To create an `FF_Event_Types` Keyword and Keyword Choices:

- 1 In the Rhythmyx Workbench, open Content Design view.
- 2 Right-click the Keywords node and choose `New > Keyword`.  
The New Keyword wizard opens.
- 3 In `Keyword name`, enter `FF_Event_Types`.
- 4 In `Description`, enter *Different types of events*.



*Figure 203: New Keyword Wizard, first dialog*

- 5 Click [**Next**].  
The next wizard screen opens.

6 In the Choices table, enter the information in the following table:

Label	Value	Description
Conferences	Conference	General Conference Event
Seminar	Seminar	General Seminar Event
Training	Training	Training Event

Figure 204: Keyword Wizard, second dialog

7 Click [**Finish**].

The Keyword object appears under the Keywords node in Content Design View, and the Keyword Choices appear below the Keyword.

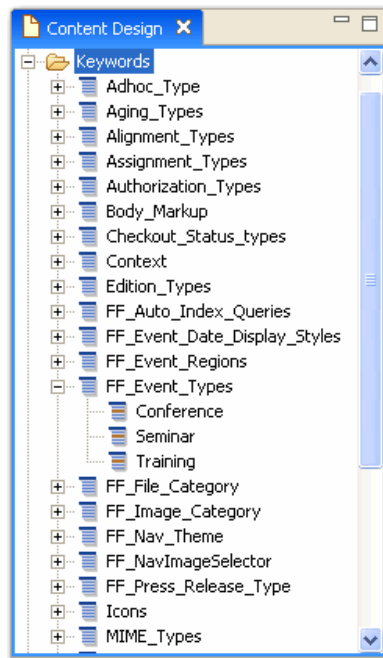
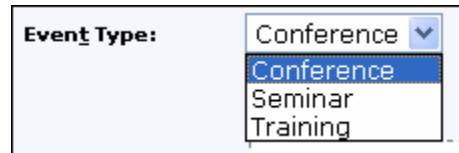


Figure 205: Keyword and choices added

The Keyword editor opens, but since the editor only duplicates the information in the wizard, click **X** in its tab to close it. When you are prompted to save it, click [**Yes**].

Now you can assign the Keyword to a list control or another component in Rhythmyx that uses Keywords. Refer to the topic ***Including a Child Field Set*** (see page 242), which explains how to assign the *FF\_Event\_Types* Keyword to the event\_type field. In the Content Editor, the field and drop list appear as:



*Figure 206: Keywords in drop list*

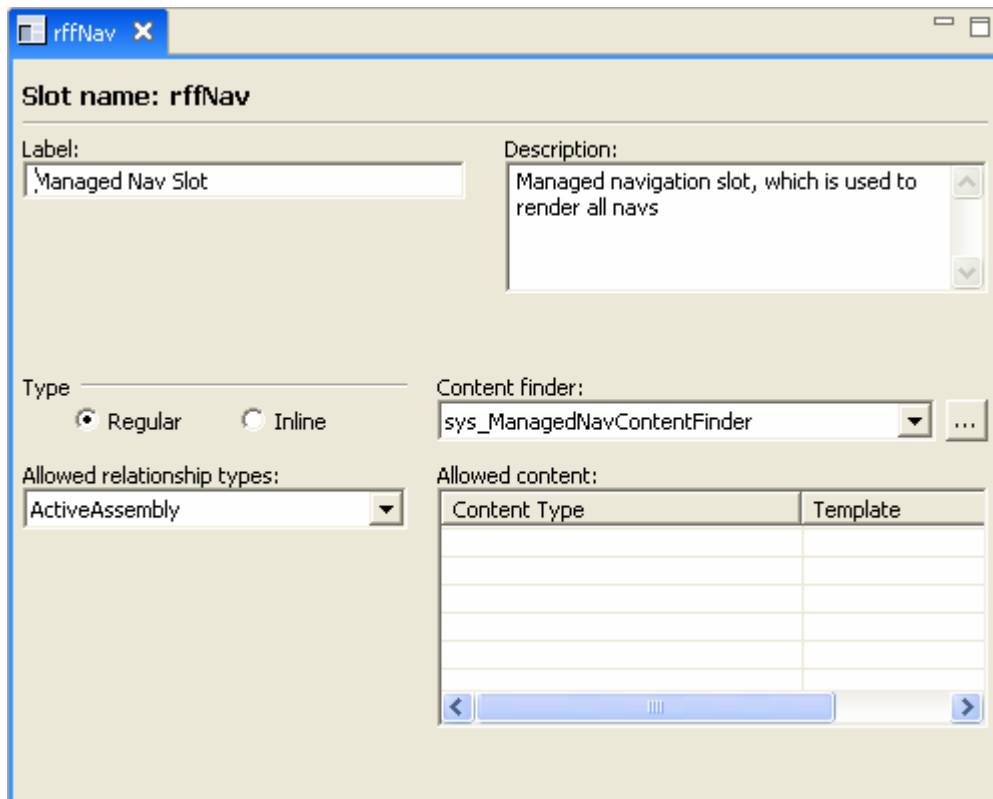
For more information about assigning Keywords to list controls, in the chapter *Creating Shared Fields*, see the topic ***Implementing a List Control*** (see page 90).



---

## Managed Navigation Slot

The Managed Navigation Slot is used to assign Managed Navigation to a Template. In most cases, you should be able to use the standard Managed Navigation Slot provided with Rhythmyx.



The screenshot shows a configuration window titled "rffNav" with the following fields and options:

- Slot name:** rffNav
- Label:** Managed Nav Slot
- Description:** Managed navigation slot, which is used to render all navs
- Type:** Regular (selected), Inline
- Content finder:** sys\_ManagedNavContentFinder
- Allowed relationship types:** ActiveAssembly
- Allowed content:** A table with two columns: Content Type and Template.

Content Type	Template

*Figure 207: Managed Navigation Slot*

The key feature of the Managed Navigation Slot is the Content Finder, `sys_ManagedNavigationContentFinder`. This Content Finder is used exclusively to assemble Managed Navigation.

---

## Customizing Navigation Look and Feel

You can customize the look and feel of Managed Navigation in two ways:

- Create new Managed Navigation Templates. You can use Dispatch Templates to select different Templates in different circumstances.
- Modify the Cascading Stylesheets used to define the specific formatting of the tagged output. You can also define several different CSS files and apply different stylesheets to different sections of your site using a Variable Selector.

### Creating Managed Navigation Templates

Managed Navigation Templates process the navigation tree XML to produce an HTML output. This output is merged with the Cascading Stylesheets to produce the final formatted output.

Managed Navigation Templates often include some HTML markup, but most of the code in these Templates consists of Velocity macros. For details about writing Velocity macros, see either of the following resources:

- Joseph D. Gradecki, *Mastering Apache Velocity*
- Rob Harrop, *Pro Jakarta Velocity*

## Left-Navigation Template

A left-navigation bar is among the most common forms of navigation in Web sites. In the FastForward implementation, the left-navigation bars are all text, so they provide a good starting point for examining navigation Templates. We will examine the rffNavLeftEI Template.

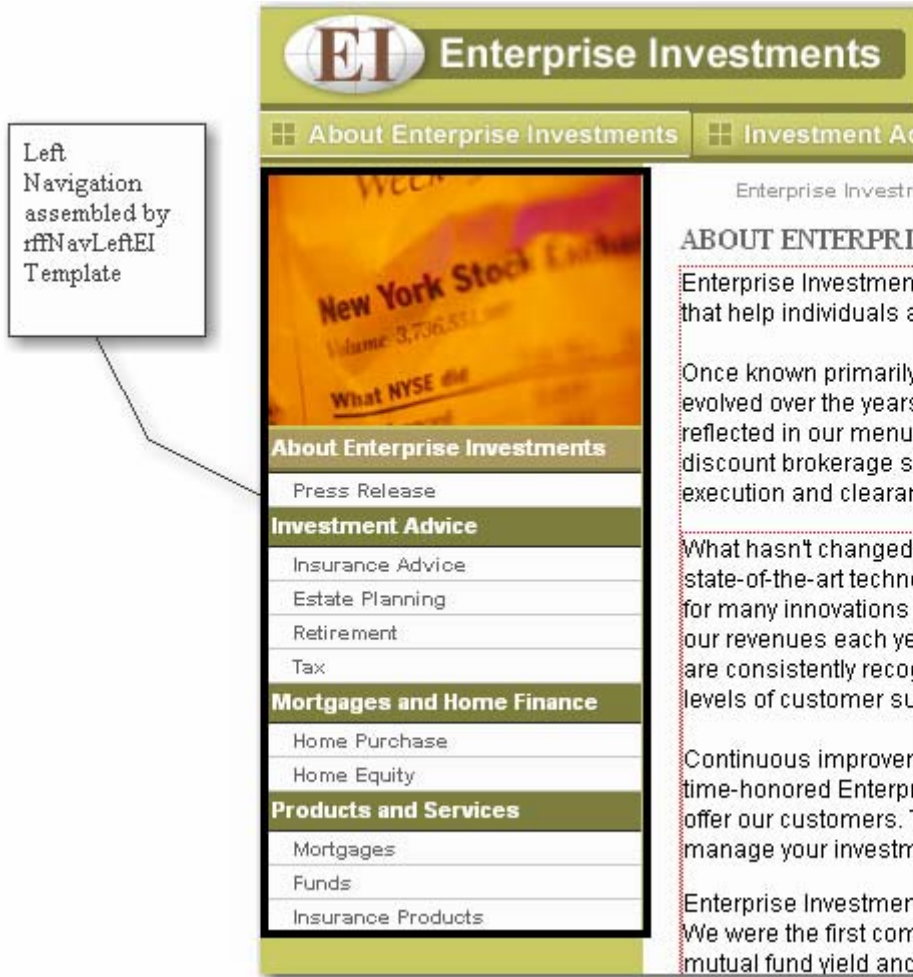


Figure 208: Page fragment calling out left navigation

A large chunk of HTML markup in this Template is the header, which is used only for previews:

```
<head>
  <!-- head is for preview only -->
  <title>EI Left Nav</title>
  <link href="$rxs_navbase/css/rxs_styles.css"
        rel="stylesheet" type="text/css">
  <script src="$rxs_navbase/js/mouseover.js"
          language="javascript" type="text/javascript"></script>
</head>
```

This markup matches what we have seen earlier when defining the header for a Global Template. The main difference is the use of the `$rxs_navbase` Context Variable. This variable is used specifically with Managed Navigation to allow for the use of variable Cascading Stylesheets. For additional details, see *Using Variable Selectors* (on page 289). Note that you must still define a binding for the Context Variable. In this case, the binding `$rxs_navbase=$sys.variable.rxs_navbase` has been defined.

The body of the Template consists of four macros:

- `#rootlevel`
- `#seflimage`
- `#firstlevel`
- `#secondlevel`

The only other markup is the call in the body to the `rootlevel` macro.

```
#rootlevel($nav.root)
```

This call initiates the assembly, passing the root of the navigation XML to the `#rootlevel` macro.

The `#rootlevel` macro builds the top image and builds the first level of the navigation bar. The macro consists of the following code:

```
#macro(rootlevel $node)
  #set($submenu = $node.getNodes("nav:submenu"))
  #selfImage($nav.self)
  #if ($node)
    #foreach ($navon in $submenu)
      #firstlevel ($navon)
    #end
  #end
#end
```

This macro has one parameter, `$node`, which takes the `$nav.root` passed from the macro call. The macro then sets the value of the variable `$submenu` as the set of Navon children of the navigation root. After that, it calls the `#selfimage` macro, which generates the image at the top of the left navigation. Finally, the macro iterates over the set of Navon children of the navigation root and calls the `#firstlevel` macro for each one.

The #selfimage macro adds the Section Image to the Left Navigation:

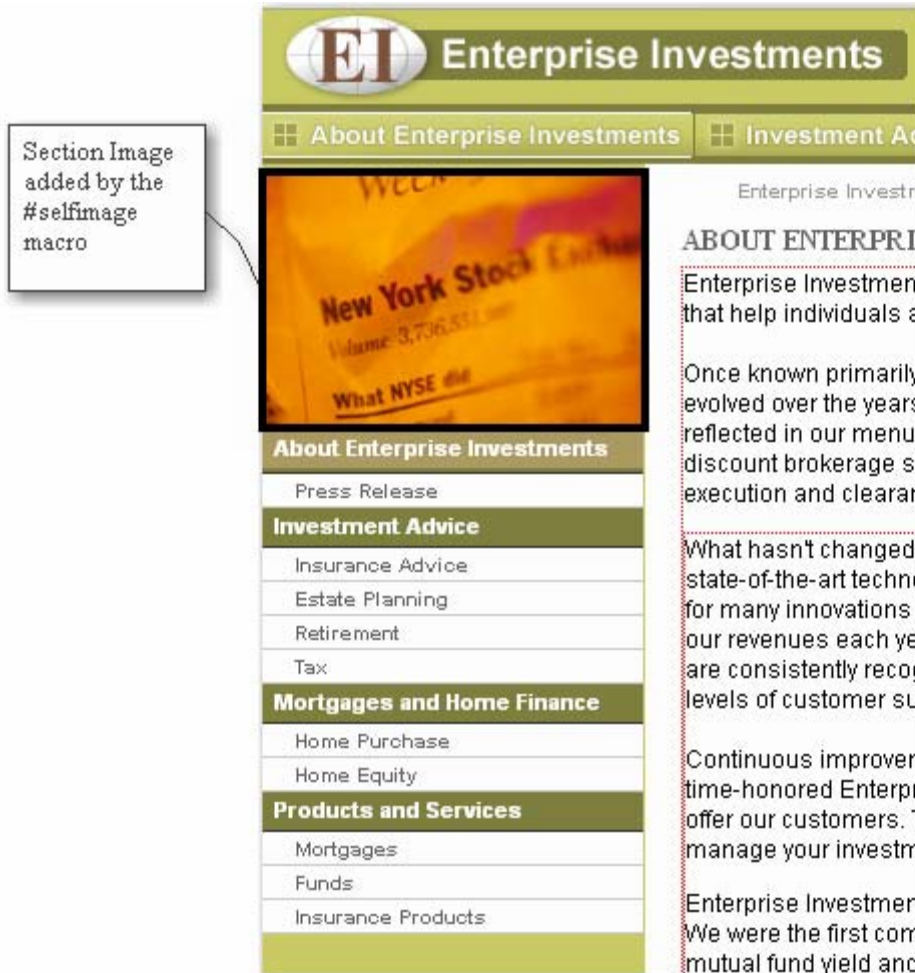


Figure 209: Left Navigation with Section Image highlighted

The code for this macro is:

```
#macro(selfImage $node)
  <!-- navon[@relation='self'] -->
  #set($image = "Bad")
  #set ($navimages = $node.getNodes("nav:image"))
  #foreach ( $navimg in $navimages )
    #set($selector = $navimg.getProperty("rx:selector").String)
    #if ($selector == "section")
      #set ( $image = $navimg )
    #end
  #end
  #set ($contentid = $image.getProperty("rx:sys_contentid").String)
  #set ($img_name = "img_a$contentid" )
  #set ($activeimage = "#imageurl($image 'active')")
  #set($title = $node.getProperty("rx:displaytitle").String)
  
#end
```

This macro tests to ensure that there is an image to display. If there is no image, it returns the text "Bad". If there is an image, the macro creates an `<img>` tag to include it in the output HTML page.

This macro first sets the value of the variable `$image` to the text "Bad". This text will be displayed if no section image is available. Next, the macro sorts through all of the NavImage Content Items associated with the Navon to find one with a value of "section" in the Image Selector field. If a Section Image Content Item is found, it is set as the value of the variable `$image` (superseding the value "Bad" set earlier).

Next, the macro sets several more variables:

- The variable `$img_name` is set to the the string "img\_a" plus the Content ID of the NavImage Content Item (which derived from the variable `$contentid`)
- The variable `$activeimage` is set to URL of the image file. (Note: the `#imageurl` macro called to set this variable is defined in the file `<Rhythmyxroot>\rx_resources\vm\rx_assembly.vm`, which is the standard location for developing custom macros that will be used in multiple Templates in the system.)

`$title` is set to the Display Title of the NavImage Content Item.

Finally, the `<img>` tag is defined. The variable values defined above are used to define the values for the attributes of the tag:

- The value for the `src` attribute is derived from the `$activeimage` variable (`src="$activeimage"`).
- The value for the `id` attribute is derived from the `$img_name` variable (`id="$img_name"`).
- The value of the `alt` attribute is derived from the `#title` variable (`alt="$title"`).

Note that some of these values could have been set more directly rather than using variables. Using variables provides somewhat more control over the macro, however.

The #firstlevel macro defines the formatting of the first level of the navigation.

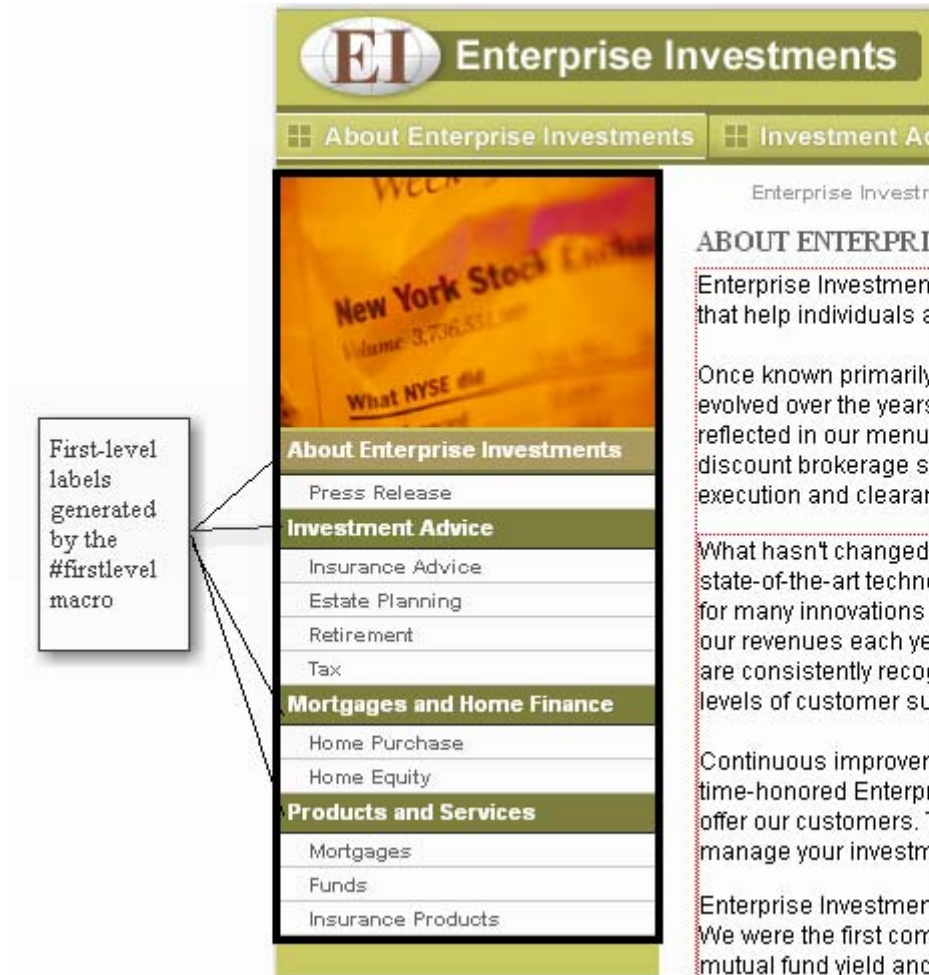


Figure 210: Left Navigation with first-level Navons highlighted

The code for this macro is:

```
#macro(firstlevel $node)
  ##<!-- "navon[@absolute-level='1']" -->
  ##<!-- xsl:variable name="indentclass" select="@relation"/ -->
  ##<!-- this macro processes the first level navons -->
  #set($title = $node.getProperty("rx:displaytitle").String)
  #set($landing_page = $node.getProperty("nav:url").String)
  #set($submenu = $node.getNodes("nav:submenu"))
  #set($axis = $node.getProperty("nav:axis").String)
  #set($indentclass = $axis.toLowerCase())

  #if ( $landing_page )
    <!-- don't process this nav if there is no Landing page -->
    <h3 class="$indentclass">
      <a href="$landing_page">$title</a>
    </h3>
    #if ( $submenu )
      <ul class="$indentclass">
```

```
        #foreach ($navon in $submenu)
            #secondlevel ($navon)
        #end
    </ul>
#end
#end
#end
```

This macro also has the `$node` parameter, which is the `navon` passed from the `#rootlevel` macro. First, the macro sets values for a number of parameters:

- `$title` is set to the Display Title of the Navon
- `$landing_page` is set to the URL of the landing page of the Navon
- `$submenu` is set to the set of Navon children of the Navon currently being processed by the macro
- `$indentclass` is set to the lower-case value of the axis of the Navon being processed by the macro in relation to the Navon that called the macro.

If the Navon being processed does not have a landing page, the macro terminates. If the Navon does have a landing page, the macro builds the HTML for the navigation link. The CSS class of the link is set to the value of the axis of the Navon being processed by the macro (`<h3 class=$indentclass>`). When defining the link, the `href` attribute of the anchor tag is set to the URL of the landing page (`href=$landing_page`) and the text of the link is set to the Display Title of the Navon the macro is processing (`<a>$title</a>`).

If the Navon has children, the macro then iterates over them, creating an unordered list of additional links as an unordered list. For each Navon child, the `#secondlevel` macro is called.



The #secondlevel macro defines the formatting for the second level of the navigation.

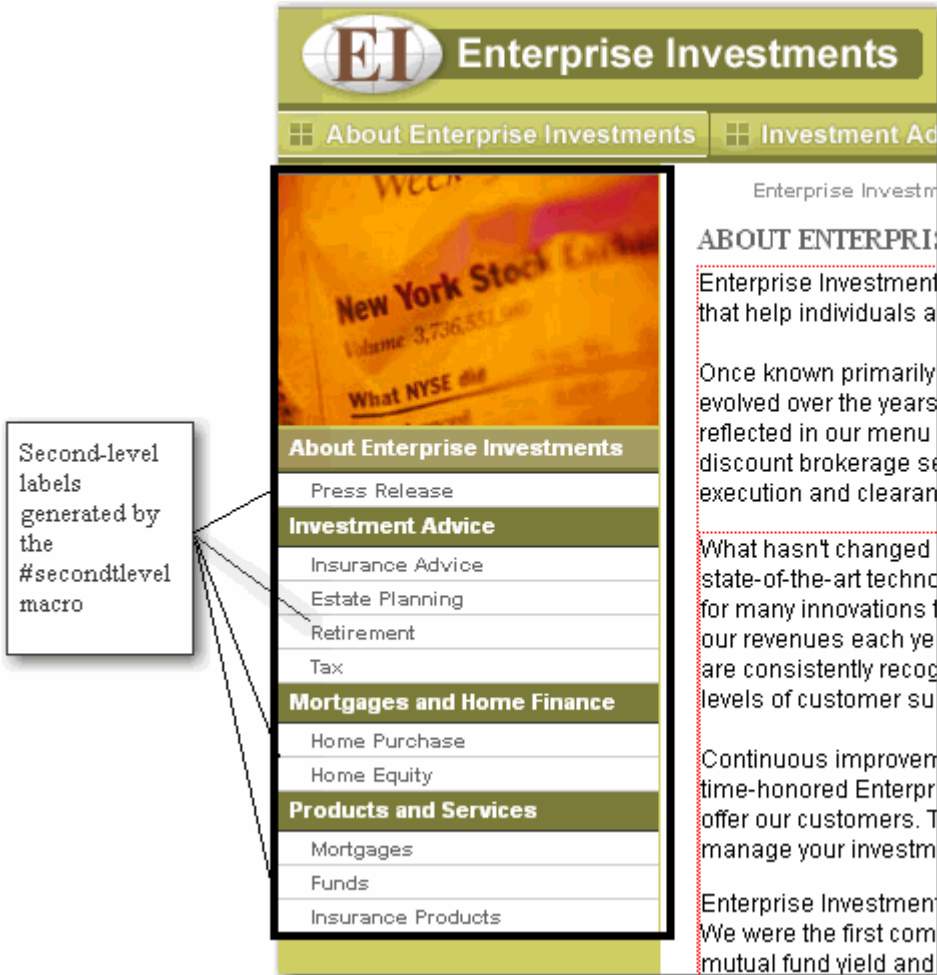


Figure 211: Left Navigation with second-level Navons highlighted

The code for this macro is:

```
#set($title = $node.getProperty("rx:displaytitle").String)
#set($landing_page = $node.getProperty("nav:url").String)
#set($submenu = $node.getNodes("nav:submenu"))
#set($axis = $node.getProperty("nav:axis").String)
#set($indentclass = $axis.toLowerCase())
  <li class="$indentclass">
    <a href="$landing_page">$title</a>
  </li>
#end
```

Like the other macros in this Template, this macro requires the \$node parameter, which is passed from the #firstlevel macro when calling the #secondlevel macro. The macro beings by setting the same set of parameters as the #firstlevel macro.

- \$title is set to the Display Title of the Navon
- \$landing\_page is set to the URL of the landing page of the Navon

- \$submenu is set to the set of Navon children of the Navon currently being processed by the macro
- \$indentclass is set to the lower-case value of the axis of the Navon being processed by the macro in relation to the Navon that called the macro.

Again, it tests whether the Navon has a landing page and terminates if it does not. If the Navon does have a landing page, the macro creates a list item whose contents are a link to the landing page of the Navon. The CSS class of the list item is specified by the \$indentclass (<li class=\$indentclass). The URL of the anchor tag is the URL of the Navon's landing page (href=\$landingpage) and the text of the link is Display Title of the Navon being processed(\$title).

## Customizing Navigation CSS

All installations need to adapt the look and feel of the navigation to conform to the overall site design. The FastForward implementation relies on a Cascading Stylesheet file (<Rhythmyxroot>\web\_resources\rxs\_nav\css\rxs\_styles.css) to define the output look and feel. Thus, when invoked, the left navigation Template *we examined earlier* (see page 279) produces the following HTML when previewed:

```

  <h3 class="self">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=3&sys_siteid=301&sys_authtype=0&sys_contentid=335&
        sys_variantid=505&sys_folderid=306&sys_context=0">About
        Enterprise Investments</a>
  </h3>
  <ul class="self">
    <li class="descendant">
      <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
          sys_revision=4&sys_siteid=301&sys_authtype=0&sys_contentid=494&
          sys_variantid=505&sys_folderid=511&sys_context=0">Press
          Release</a>
    </li>
  </ul>
  <h3 class="sibling">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=2&sys_siteid=301&sys_authtype=0&sys_contentid=476&
        sys_variantid=538&sys_folderid=307&sys_context=0">Investment
        Advice</a>
  </h3>
  <ul class="sibling">
    <li class="none">
      <a
href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
```

```

        sys_revision=4&sys_siteid=301&sys_authtype=0&
        sys_contentid=344&sys_variantid=538&sys_folderid=311&
        sys_context=0">Insurance Advice</a>
    </li>
    <li class="none">
        <a
href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=4&sys_siteid=301&sys_authtype=0&
        sys_contentid=339&sys_variantid=538&sys_folderid=310&
        sys_context=0">Estate Planning</a>
    </li>
    <li class="none">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=2&sys_siteid=301&sys_authtype=0&
        sys_contentid=350&sys_variantid=538&sys_folderid=312&
        sys_context=0">Retirement</a>
    </li>
    <li class="none">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=2&sys_siteid=301&sys_authtype=0&
        sys_contentid=356&sys_variantid=538&sys_folderid=313&
        sys_context=0">Tax</a>
    </li>
</ul>
<h3 class="sibling">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=1&sys_siteid=301&sys_authtype=0&
        sys_contentid=485&sys_variantid=538&sys_folderid=308&
        sys_context=0">Mortgages and Home Finance</a>
</h3>
    <ul class="sibling">
        <li class="none">
            <a
href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
                sys_revision=1&sys_siteid=301&sys_authtype=0&
                sys_contentid=371&sys_variantid=538&sys_folderid=315&
                sys_context=0">Home Purchase</a>
            </li>
            <li class="none">
                <a
href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
                    sys_revision=1&sys_siteid=301&sys_authtype=0&
                    sys_contentid=366&sys_variantid=538&sys_folderid=314&
                    sys_context=0">Home Equity</a>
                </li>
        </ul>
<h3 class="sibling">
    <a href="http://10.10.10.100:9662/Rhythmyx/assembler/render?
        sys_revision=3&sys_siteid=301&sys_authtype=0&
        sys_contentid=487&sys_variantid=538&sys_folderid=309&
        sys_context=0">Products and Services</a>
</h3>
    <ul class="sibling">
        <li class="none">

```

```

      <a
href="http://10.10.10.100:9662/Rhythmyx/ assembler/render?
      sys_revision=2&sys_siteid=301&sys_authtype=0&
      sys_contentid=408&sys_variantid=538&sys_folderid=318&
      sys_context=0">Mortgages</a>
    </li>
    <li class="none">
      <a
href="http://10.10.10.100:9662/Rhythmyx/ assembler/render?
      sys_revision=3&sys_siteid=301&sys_authtype=0&
      sys_contentid=376&sys_variantid=538&sys_folderid=316&
      sys_context=0">Funds</a>
    </li>
    <li class="none">
      <a
href="http://10.10.10.100:9662/Rhythmyx/ assembler/render?
      sys_revision=3&sys_siteid=301&sys_authtype=0&
      sys_contentid=402&sys_variantid=538&sys_folderid=317&
      sys_context=0">Insurance Products</a>
    </li>
  </ul>

```

Note the class attributes, which specify style classes defined in the `rxs_styles.css` stylesheet. Thus, you have substantial control over the format of the file output of the navigation by modifying the stylesheet or defining your own stylesheet and pointing your output to that file. You could even implement a different CSS convention, although you would have to use it for all of your Global Templates.

## Variable Selectors

Context Variables allow you to populate different values into a page depending on the output Context (preview, publish, etc.). These values are generally used to define the location of static, non-managed design resources such as CSS and images. Variable Selectors provide Web Masters with a greater amount of flexibility when working with Context Variables within a Site. Once the necessary Context Variables are defined, the Web Master has the ability to:

- Define a single Context Variable to be used throughout an entire Site;
- Define different Context Variables to be used in different Site Sections.

For example we could modify the Products and Services section of the Corporate Investments Site to use a different look and feel from the rest of the Site by applying a different set of static images and CSS for that section than we use for the rest of the Site. To implement this, we would need to define the additional set of images and CSS, then define a Variable Selector, which we would choose for the Navons in the Products and Services Folder and its Subfolders.

The screenshot shows a configuration form with the following fields and values:

- \* System Title:** Products and Services
- \* Title:** Products and Services
- \* Start Date:** 2004-08-18 00:00:00.0
- Image Selector:** None
- Variable Selector:** ProductServices

At the bottom of the form are two buttons: "Update" and "Close".

Figure 212: Defining an Alternate Variable for a Site Section

This section of the Site would use the same Content Types as the rest of the Site, but applying different CSS and static images would result in a different look and feel for the output in this section.

### Default Variable Selector Variables

FasfFoward comes with a pre-defined Variable Selector Variable: `rxs_navbase`. Use this Context Variable in any Site in which you want to use Variable Selectors. You should define a value for this Context Variable for each output Context for each Site. The values you define for this Context Variable should specify a generic location where the CSS, JavaScript, and static images for each Site will be stored.

The `rxs_navbase` Context Variable is specified as the Variable Selector value in the `Navigation.properties` file (`<Rthyxroot>\rxconfig\server\Navigation.properties`).

```
navon.variant.info=navinfo
navon.variant.navlink=navlink
navon.variant.tree=navontree
navtree.content_type=navtree
navtree.field.theme=nt_theme
navtree.theme.default=DefaultTheme
navtree.variable=rxs_navbase
navtree.variant.info=navtreeinfo
navtree.param.theme=nav_theme
```

Figure 213: Variable Selector Value as Defined in `navigation.properties`

If you want to use a different Context Variable name, you must change the value of the `navtree.variable` entry in this file.

### Using Variable Selectors

For the purposes of this procedure, we will assume that you are using the default Variable Selector Context Variable, `rxs_navbase`.

- 1 Create the CSS files, static images, and JavaScript files you want to use create the look and feel for your Site section, and copy them in the appropriate web resources directories.
- 2 In any Templates used on the Site, substitute the Variable Selector Context Variable for any existing Context Variable (usually `ResourcePath`). For example, change `$sys.variables.ResourcePath\css\rxs_styles.css` to `$sys.variables.rxs_navbase\css\rxs_styles.css`
- 3 Register a new Context Variable pointing to the new design elements you want to use for your Site section. For example, to implement a different look and feel for the Products and Services Site section as discussed earlier, we might define the Context Variable `ProductServices`:



ProductServices		Add Value	
Value	Site(id)	Context(id)	
 <code>./web_resources/prodservices</code>	Internet(301)	Preview(0)	
 <code>/web_resources/prodservices</code>	Internet(301)	Publish(1)	

Figure 214: Context Variables for the Products and Services Site Section

- 4 Start Content Explorer and find the Navon in the Products and Services Folder. Edit the Navon (use Quick Edit if the Navon is already Public), specifying name of your Context Variable in the Selector drop list. Update and close the Content Item, Transitioning it back to Public if necessary.

Preview an Item in the Products and Services section. These pages now use the look and feel created by the CSS, static images, and JavaScript you created. The same templates are simply using different CSS and design images to format the output.

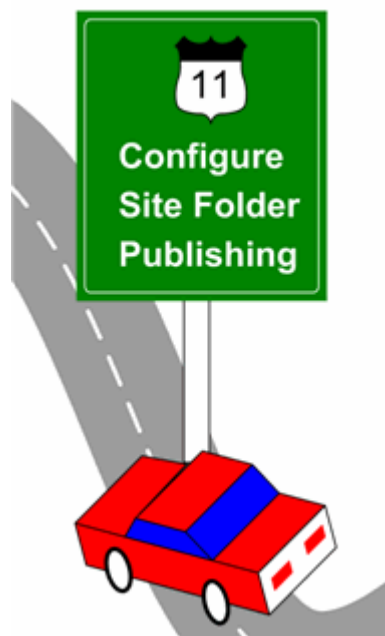
## Navon Properties

Navons have several unique properties and child nodes that play an important role in implementing Managed Navigation.

Navon properties are only used in Templates.

Variable	Data Type	Description
nav:axis	String	The axis of the Navon being processed in relation to the Navon from which processing was initiated. Available options include: <ul style="list-style-type: none"> <li>▪ ANCESTOR: a node in in the path of the Navon higher than the PARENT Navon node</li> <li>▪ DESCENDANT: A node in the path after the Navon</li> <li>▪ NONE: No other category applies</li> <li>▪ PARENT: The immediate predecessor of the Navon in its path.</li> <li>▪ SELF: The Navon itself.</li> <li>▪ SIBLING: Another Navon that shares the PARENT of the Navon.</li> </ul>
nav:url	String	The URL of the Navon's landing page.
nav:landingPage	Node	The landing page Content Item.
nav:leaf	Boolean	Boolean specifying whether the Navon is a leaf (in other words, has no children) <ul style="list-style-type: none"> <li>▪ True: The Navon is a leaf (has no children).</li> <li>▪ False: The Navon is not a leaf (has children).</li> </ul>
nav:submenu	Node Iterator	The variable contains all Navon children of the Navon being processed.
nav:image	Node Iterator	This variable contains all NavImage children of the Navon being processed.
nav:selectedImage	Node	The NavImage selected by the Selector

# Configuring Publishing



During the modelling and design process you examined how your current publishing process works, and determined any changes that you want to make to the process in Rhythmyx. Now that you have determined how you want publishing to function in your system, you can plan and implement the components required.

You have already defined your publishing Site in the chapter *Setting up the Publishing Site and Basic Navigation* (on page 61) by setting up the Site's folder hierarchy in Rhythmyx Content Explorer and registering the Site, which involved specifying the Site address, the home page, and other information.

Before you implement the other publishing components, we will define these components and explain how publishing in Rhythmyx works. Then we will review how you want your publishing system to work, and demonstrate how to implement the components necessary.

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

- A Content List specifies which Content Items to process 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 application to run the job, and passes the Content Lists to publish to the Publisher application. Rhythmyx sends the publishing request to the Publishing Manager when an Edition is published.

- Publishers are applications that are 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 Publisher runs after it receives a request from the Publishing Manager, and it resides on the Rhythmyx Server or remote locations.
- 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 manage multiple Sites on the same machine.
- A Context is a location or environment in which content is published or assembled. For example, the publishing location for Enterprise Investments content is one context, and the location for previewing assembled content before publishing it is another context.
- Location Scheme - Location Schemes are associated with specific contexts. Location schemes specify the rules for configuring the addresses of content items. Location schemes have various uses. Two of the main uses of Location Schemes are:
  - telling the Publisher where to publish content on a file system.
  - creating URLs so that content items can link to each other when they appear in a browser.
- Variables - Variables are used in Location Schemes to enable you to use the same Location Schemes for different Sites and Contexts. For example, if a content item's URL includes the name of the Site, the Site name can be specified in a variable.

Now that the components and their functions are defined, we will describe in more detail how publishing works.

A System Administrator can manually initiate a publishing request by clicking the [**Publish**] button next to an Edition on the Editions page in Content Explorer or can 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.



The following graphics illustrate this process.

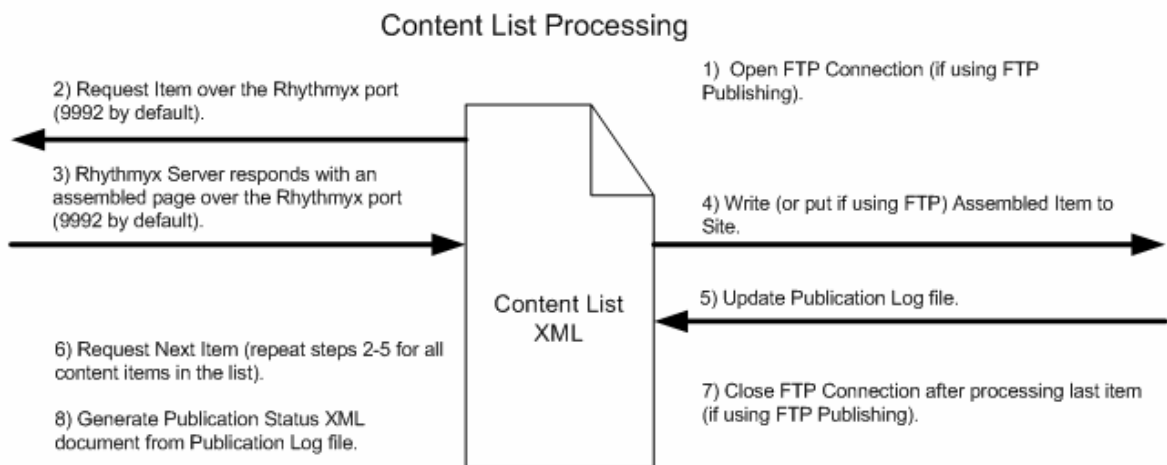
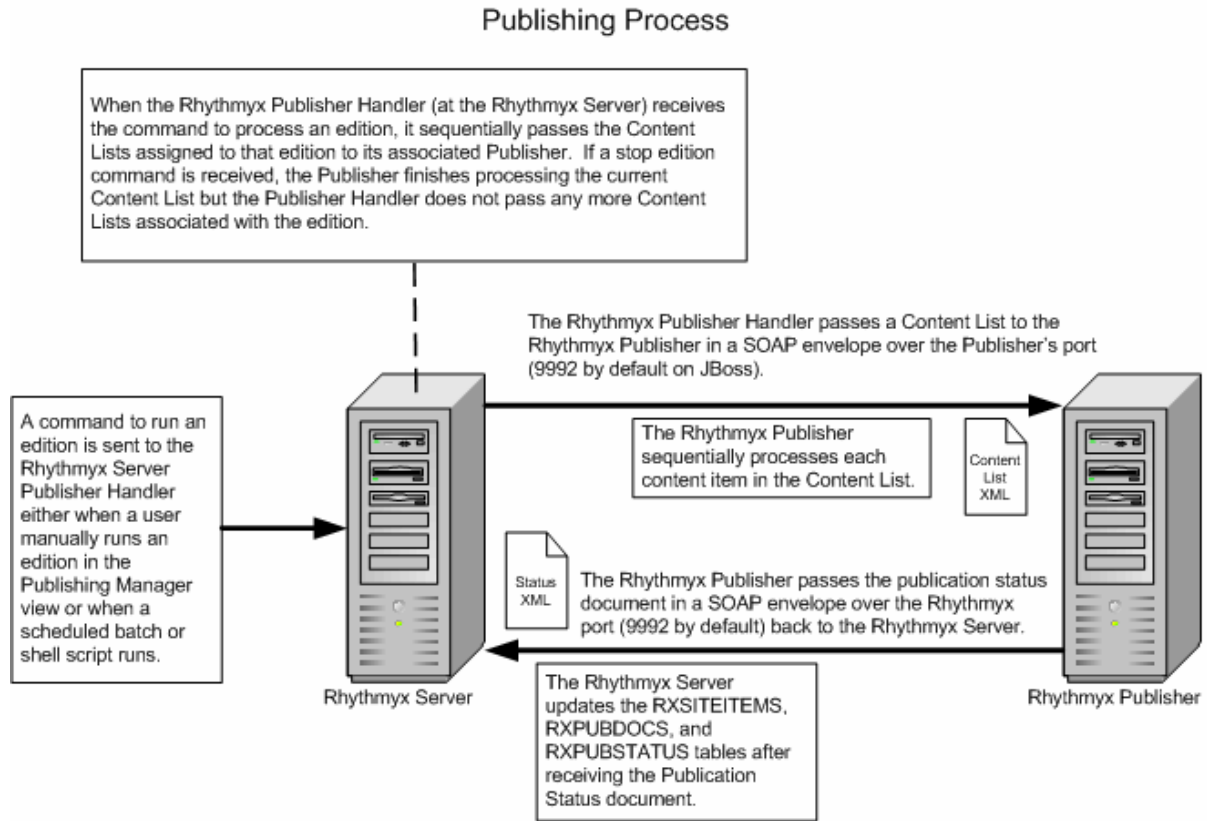


Figure 215: Publishing Engine Processing

---

## Publishing Specifications

The following publishing specifications were created at the end of your Modelling and Design session.

**Site Folder hierarchy** - We will use the FastForward Enterprise Investments Site folder hierarchy because it is sufficiently detailed to demonstrate how the Site Folder hierarchy in Content Explorer is duplicated on the publish Site after publishing. The procedure for creating a Site folder hierarchy has already been demonstrated in the topic *Setting up the Publishing Site and Basic Navigation* (on page 61), so we will not duplicate it in this section.

**Publisher** - The Publisher application should be installed locally in Rhythmyx's JBoss application server, and run on port 9992 with Rhythmyx. For many of your publishing scenarios, you can use the default Publisher registration that FastForward provides: *Localhost Publisher Default Port*. Therefore, we will use this Publisher to perform publishing in this section. For changes you may need to make to the Publisher registration to meet the requirements of your implementation, see Setting Up Publishing in the Production Environment.

**Content Lists** The following Content Lists are specified in your system's specifications.

- A Content List that selects all binary Content Items that are ready to be published and specifies that they should be delivered to the registered Site on the file system. We will create this Content List because in most implementations users require a separate Content List for publishing only image and file items. Since publishing binary Content Items takes longer than publishing non-binary Content Items, it is useful to have the option of publishing these items separately.
- A Content List that selects all non-binary Content Items that are ready to be published and specifies that they should be delivered to the registered Site on the file system. We will create this Content List because users require it in most implementations. As we have already stated, it is useful to have the option of publishing binary and non-binary Content items separately because of the differences in time involved. (For example, to save time, you may wish to only publish non-binary Content Items in some publishing runs.)
- A Content List that selects all new and modified public Content Items (an incremental Content List) and specifies that they should be delivered to the registered Site on the file system. We include this Content List because it is also standard in most implementations. It enables customers to update their Web Site as often as necessary without going through the lengthy process of republishing all of their content.
- A Content List that selects all Content Items in an archive state and unpublishes them. Again, we include this Content List because it is required in most implementations. It is best practice to include a separate Unpublishing Content List to avoid unpublishing a version of a Content Item that was meant to be newly published.

---

Note: For a system to run you need at least a Content List for publishing and a Content List for Unpublishing. However, most implementations benefit from separate Content Lists for binary and text data and an incremental Content List.

---

**Editions** The following Editions are listed in your specifications. Both of these Editions are automatic since they will be scheduled to run.

- An Edition scheduled to run once a week that publishes all items ready to be published to the Enterprise Investments site, and unpublishes all items in an archive state. We include this type of edition because most customers will require an edition that republishes the Site at intervals, but not on a daily basis.
- An Edition scheduled to run twice a day that publishes all new and modified items in a publish state to the Enterprise Investments site, and unpublishes all items in an archive state. We include this type of edition because it is required by many customers who need to update their Sites as soon as new content arrives (sometimes several times a day).

**Contexts** The following Contexts are listed in your specifications:

- A preview Context - Rhythmyx provides this Context. It must exist so that users can preview assembled content in Content Explorer before publishing it.
- A publishing Context - We are including this Context because it is necessary for all users. It tells the Publisher where to deliver Content Items that are ready to be published.
- A Context for assembling items - We are including this Context because it is usually necessary for telling the Web browser where to find the Content Items that other Content Items link to. This address is different from the publishing context because the browser looks at the Site address rather than the publishing root defined for your Site. In some cases, this Context is not necessary, for example, if the customer is publishing content to another database and not immediately planning to display it on a Web Site.

**Location Schemes** The following Location Schemes are listed in your specifications:

- A location scheme that generates the path for publishing each item on the file system. We will associate this Location Scheme with the publishing Context. It will create the delivery address for Content Items that are ready to be published.
- A location scheme that generates the URL for each item (so items can link to one another). We will associated this Location Scheme with the Context for assembling items. It will generate the Site address of published Content Items for the browser to use.

## Content Explorer's Publishing Tab

The publishing components in your system, including Sites, Publishers, Content Lists, Editions, Contexts, Location Schemes, and Variables are registered or configured in the Publishing Administrator of Content Explorer. The Publishing Administrator is accessed through the Publishing tab, which is only visible to users with the proper access.

On the left side, the Publishing Administrator displays a navigation pane with links to the editors for adding and modifying different publishing components and the Publications Log, which is shown in the graphic below. On the right side the Publishing Administrator displays the various editors.

The screenshot shows the Rhythmyx Publishing Administrator interface. The top navigation bar includes 'Content', 'Publishing', and 'Workflow' tabs, along with a 'Logout' button. The user information is displayed as 'User: admin1', 'Roles: Admin, Default, E1\_Ad...', and 'Community: Enterprise\_Investments'. The main content area is titled 'Enterprise Investments (301)' and shows a table of publications. The left navigation pane includes links for Publishers, Sites, Contexts, Variables, Editions, Content Lists, and Publication Log.

		Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status	Publications by Site				
Publishers By Name By ID		2006-08-02 09:54:19.0 (329:329)	rffEiFullBinary (310)	Full Enterprise Investments		Success	73	0	0	0	0
Sites By Name By ID		2006-08-02 09:54:34.0 (329:330)	rffEiFullNonBinary (311)	Full Enterprise Investments		Success	68	0	0	0	0

Figure 216: Publishing Administrator

Access each editor by clicking its link on the left. For more information on the Publishing Administrator, see the online *CMS Help*.

---

## Defining Content Lists

A Content List is a Rhythmyx query that defines which content items are extracted from the database for publishing or for unpublishing (removal from a Site).

Content Lists may query on any properties of Content Items, such as their Content Type, to determine whether or not they are published. Content lists can define whether Rhythmyx will publish the whole Site (a Full Publish) or only those content items that have been added or updated since the last Publication run (an Incremental Publish). Many customers perform a Full Publish every one or two weeks, and an incremental publish one or more times a day.

Defining Content Lists make them available to be included in an Edition. We will discuss Editions later in this chapter but for now we will focus on defining Content Lists.

Your specifications require four Content Lists:

- one that selects binary content items
- one that selects non-binary content items
- one that selects new and modified content items
- one that unpublishes content items that are in an archive state.

In the following topics, we use the Publishing Administrator to create the following FastForward Content Lists:

- *rffEiFullBinary*
- *rffEiFullNonBinary*
- *rffEiIncremental*
- *rffEiUnpublish*

FastForward includes additional Content Lists that we will not recreate in this section.

---

*Note: You cannot create a new Content List with the same name as a Content List that exists in FastForward. Instead, create similar Content Lists included in your implementation plan or copy our steps but give your Content Lists different names.*

---

## Defining the Full Binary Content List

The first type of Content List you will create is the Full Binary. Full Binary Content Lists publish all binary content that is in a public state within the specified Site Root folder.

To define the Full Binary Content List:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Click the Publishing tab.
- 3 Click the By Name link under the Content Lists heading in the left navigation pane.  
Content Explorer displays the Content List administration page.

- 4 Click the [New Content List] button.  
Content Explorer displays the Edit Content List page.

Edit Content List		Errors		
Id	0-21-330			
Name	<input type="text" value="new_0-21-330"/>			
Description	<input type="text"/>			
Url	<input type="text" value="/Rhythmyx/contentlist?sys_deliverytype=filesystem&amp;sys_contentlist=new_0-21-330"/>			
Type	<input checked="" type="radio"/> Normal <input type="radio"/> Incremental			
Edition Type	Automatic			
Generator	-- Unassigned --			
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead></table>	Name	Value	
Name	Value			
Template Expander	-- Unassigned --			
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead></table>	Name	Value	
Name	Value			
Item Filter	public			
<input type="button" value="Save"/> <input type="button" value="Cancel"/>				

Figure 217: Content List Editor

Note: The Content List user interface may change in future releases.

- 5 The Name field is mandatory and is assigned a system-generated ID, with the prefix "new." This field is editable. It is a best practice to give the Content List a unique Name. For this example enter something similar to *rffEIFullBinary* since Rhythmyx will not let you enter the name of an existing Content List. (Note that no spaces are allowed in the name.)
- 6 The Description field is optional; however it is a best practice to provide a description for the Content List. For this example enter the following text: *Site Root Full for Binary Content Types - Enterprise Investments*.
- 7 The URL field designates the location of the content list for the site and passes any values of variables that the publisher needs. By default, it is filled with a sample URL which gives you parameters designating the type of publishing as filesystem and the default Content List name. Change the URL to the following, replacing *rffEIFullBinary* at the end of the URL with the name you have given your Content List.

`/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_assembly_context=301&sys_contentlist=rffEIFullBinary.`

This content list URL passes the following parameters:

- `sys_deliverytype` - The name of the publishing plugin; in this case it is *filesystem*, but other options include *ftp* and *database*.
- `sys_assembly_context` - The context to use to create assembly URLs. If not supplied, the publisher uses the Context specified in the Edition.
- `sys_contentlist` - The name of this content list.

Note: For a list of all of the possible Content List parameters, see the online *CMS Help*.

- 8 Leave Type as *Normal*. Only change this selection when you are creating an Incremental Content List.

- 9** Leave **Edition Type** as Automatic. Automatic Editions run according to schedules and use pre-defined Content List queries. For more information about Edition types, see *Creating Editions*. (see "Creating Editions" on page 328)
- 10** The **Generator** field specifies the extension (java plugin) that creates the list of Content Items to be published. The choices for Generators are:
- Java/Global/percussion/system/sys\_SearchGenerator - Queries the repository for content matching the query that follows, and generates the content list using the matching content.
  - Java/Global/percussion/system/sys\_SelectedItemsGenerator - Used for dynamic publishing only. Locates the content item IDs from an HTTP parameter, and generates the content list using these content items.

Choose Java/Global/percussion/system/sys\_SearchGenerator because we are creating a typical Content List. (For this exercise, always use the sys\_SearchGenerator).

Notice that the editor displays a text field named **Query** when you make the selection in the **Generator** field above.

- 11** In the **Query** field enter the following query text:

```
select rx:sys_contentid, rx:sys_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where
jcr:path like '//Sites/EnterpriseInvestments%'
```

This is a *JSR-170 query* (see page 190) that we can break down as follows:

Expression in query	Meaning
select rx:sys_contentid, rx:sys_folderid	Return each content item and its folder (including all of the content item and folder fields and properties)
from rx:rfffile,rx:rffimage,rx:rffnavimage	Query only the Content Types rfffile, rffimage, and rffnavimage.
where jcr:path like '//Sites/EnterpriseInvestments%'	Only look in the Content Explorer Site Folder path //Sites/EnterpriseInvestments

In other words, the query tells the generator to return the content item and folder of all file, image, and navimage content items in the folders and subfolders of the Site Folder path //Sites/EnterpriseInvestments:

- 12** The **Template Expander** field specifies an extension (java plugin) that provides templates for assembling each Content Item chosen to be published. You must supply parameters to each default **Template Expander**. The choices for **Template Expanders** are:
- sys\_SiteTemplateExpander - Used for site folder publishing. Finds the default Page Template or the binary template associated with each Content Item. (For this exercise, always use sys\_SiteTemplateExpander.)
  - sys\_ListTemplateExpander - Lets user specify a list of Templates for assembling the content item.

In the drop down list select Java/Global/percussion/system/sys\_SiteTemplateExpander since we are using Site Folder publishing. Fields for the parameters **siteid** and **default\_template** appear below the **Template Expander** drop list.

- 13** In the `siteid` field below the Template Expander field enter 301, the site id for Enterprise Investments.
- 14** The `default_template` field lets you specify which Templates to publish for a Content Item of a certain Content Type. The **Publish** value for each Template is marked in the Template editor in the Rhythmyx Workbench:
- *all or unspecified* (blank) - Use all Templates whose **Publish** value is *Default*.
  - *dispatch* - Use all dispatch Templates assigned to the Content Item. Dispatch Templates include conditions for choosing the correct Template.
  - *none* - Use all Templates whose **Publish** value is *Always*.

Leave `default_template` blank to indicate that you want to use all Templates associated with the Content Type for the Site whose **Publish** value is *Default*.

- 15** The `Item Filter` value specifies a filter for Content Items included on the Content List. The following values are valid:
- `unpublish` - include all items in an archive State.
  - `preview` - include all items.
  - `public` - include all items in a public State.
  - `sitefolder` - include all items in the specified site folder.
  - `Unassigned` - indicates that no filter has been chosen. Not a valid value; will cause a publishing error.

Leave `public`, since you want to include all Content Items in a public State in the Content List.



Your completed editor should appear similar to the following screenshot:

The screenshot shows the 'Edit Content List' form with the following fields and values:

- Id:** 0-21-310
- Name:** rffEiFullBinary
- Description:** Site Root Full for Binary Content Types - Enterprise Investments
- Url:** /Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_c
- Type:** Normal (selected), Incremental
- Edition Type:** Automatic
- Generator:** Java/global/percussion/system/sys\_SearchGenerator
- Query:** select rx:sys\_contentid, rx:sys\_folderid from rx:rffile,rx:rffimage,rx:rffnavimage where jcr:|  
(Required) The JSR-170 sql style query to use in finding the base items for the content list.
- Template Expander:** Java/global/percussion/system/sys\_SiteTemplateExpander
- siteid:** 301  
The id of the site to use when expanding the templates. If not specified this may be provided in the request.
- default\_template:** (empty)  
This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.
- Item Filter:** public

Buttons: Save, Cancel

Figure 218: Full Binary Content List definition

16 Click [Save].

## Defining the Full Non-Binary Content List

Next, you will define a Full Non-Binary Content List that publishes all non-binary content that is in a public state within the specified Site Folder. The purpose of including separate binary and non-binary Content Lists is to enable you to publish the Binary content first, so that the binary content items are available on the publishing site when non-binary content items look for them to include in their assembled pages. In addition, it takes longer to publish binary content, so you may want the ability to only schedule publishing of your non-binary Content List on some publishing runs.

Since we have covered how to open the Edit Content List page and described its fields in depth in the topic *Defining the Full Binary Content List* (see page 297), see that topic for more information about the following steps.

To define the Full Non-Binary Content List:

- 1 Open a new Edit Content List page.
- 2 In Name, enter something similar to (but not the same as) *rffEiFullNonBinary*.
- 3 In Description, enter *Site Root Full for Non-Binary Content Types - Enterprise Investments*.
- 4 Change the URL to:

*/Rhythmyx/contentlist?sys\_deliverytype=filessystem&sys\_assembly\_context=301&sys\_contentlist=rffEiFullNonBinary*

Replace *rffEiFullNonBinary* at the end of the URL with the name you have given your Content List.

- 5** Leave **Type** as *Normal*.
- 6** Leave **Edition Type** as *Automatic*.
- 7** Choose Java/Global/percussion/system/sys\_SearchGenerator in the Generator drop list.  
The editor displays the text field **Query** when you make the selection in the **Generator** field.
- 8** In the **Query** field, enter the following query text:  

```
select rx:sys_contentid, rx:sys_folderid from
rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffcontacts,rx:rffevent,rx:rffexternallink,rx:rffgenericword,rx:rffgeneric,rx:rffhome,rx:rffpressrelease where jcr:path like
'//Sites/EnterpriseInvestments%'
```

This JSR-170 query tells the generator to return the contentid and folderid of all autoindex, brief, calendar, contacts, event, external link, generic word, generic, home, and press release content items in the Site Folder path //Sites/EnterpriseInvestments.
- 9** In the Template Expander drop down list select Java/Global/percussion/system/sys\_SiteTemplateExpander since we are using Site Folder publishing. Fields for the parameters **siteid** and **default\_template** appear below the Template Expander drop list.
- 10** In the **siteid** field below the Template Expander field enter 301, the site id for Enterprise Investments.
- 11** Leave the **default\_template** field blank to indicate that you want to use all Templates associated with the Content Type for the Site whose **Publish** value is *Default*.
- 12** In **Item Filter**, leave *public*, since you want to include all Content Items in a public State in the Content List.

Your completed editor should appear similar to the following screenshot:

Edit Content List		Errors						
<b>Id</b>	0-21-311							
<b>Name</b>	rffEiFullNonBinary							
<b>Description</b>	Site Root Full for Non-Binary Content Types - Enterprise Investments							
<b>Url</b>	/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_assembly_context=301&sys_c							
<b>Type</b>	<input checked="" type="radio"/> Normal <input type="radio"/> Incremental							
<b>Edition Type</b>	Automatic							
<b>Generator</b>	Java/global/percussion/system/sys_SearchGenerator							
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>query</td> <td>select rx:sys_contentid, rx:sys_folderid from rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffc</td> </tr> </tbody> </table> <p>(Required) The JSR-170 sql style query to use in finding the base items for the content list.</p>		Name	Value	query	select rx:sys_contentid, rx:sys_folderid from rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffc		
Name	Value							
query	select rx:sys_contentid, rx:sys_folderid from rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffc							
<b>Template Expander</b>	Java/global/percussion/system/sys_SiteTemplateExpander							
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>siteid</td> <td>301</td> </tr> <tr> <td>default_template</td> <td></td> </tr> </tbody> </table> <p>The id of the site to use when expanding the templates. If not specified this may be provided in the request.</p> <p>This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.</p>		Name	Value	siteid	301	default_template	
Name	Value							
siteid	301							
default_template								
<b>Item Filter</b>	public							
<input type="button" value="Save"/> <input type="button" value="Cancel"/>								

Figure 219: Registration for rffEiFullNonBinary Content List

13 Click [Save].

## Defining the Incremental Content List

An Incremental Publish differs from a Full Publish in that it publishes only Content Items that you have changed since they were last published and new Content Items. Full Publishing publishes every Content Item in a Public State.

Many customers perform a Full Publish every one or two weeks, and an incremental publish one or more times a day. Since fewer items are published during Incremental Publishing, fewer system resources are used, thereby decreasing processing time.

Since we have covered how to open the Edit Content List page and described its fields in depth in the topic *Defining the Full Binary Content List* (see page 297), see that topic for more information about the following steps.

To define the Incremental Publish Content List:

- 1 Open a new Edit Content List page.
- 2 In Name, enter something similar to (but not the same as) *rffEiIncremental*.
- 3 In Description, enter *Site Root Incremental - Enterprise Investments*.

- 4** Change the URL to:

```
/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_assembly_context=301&sys_contentlist=rffEiIncremental.
```

Replacing *rffEiIncremental* at the end of the URL with the name you have given your Content List.

- 5** Change **Type** to *Incremental* since you want to select only Content Items that qualify as incremental.

- 6** Leave **Edition Type** as *Automatic*.

- 7** Choose *Java/Global/percussion/system/sys\_SearchGenerator* in the **Generator** drop list. The editor displays the text field **Query** when you make the selection in the **Generator** field.

- 8** In the **Query** field, enter the following query text:

```
select rx:sys_contentid, rx:sys_folderid from nt:base where jcr:path like '//Sites/EnterpriseInvestments%'
```

This JSR-170 query tells the generator to return the contentid and folderid of content items of all Content Types (*nt:base* represents all Content Types) in the Site Folder path *//Sites/EnterpriseInvestments*.

- 9** In the **Template Expander** drop down list select *Java/Global/percussion/system/sys\_SiteTemplateExpander* since we are using Site Folder publishing. Fields for the parameters **siteid** and **default\_template** appear below the **Template Expander** drop list.

- 10** In the **siteid** field below the **Template Expander** field enter *301*.

- 11** Leave the **default\_template** field blank to indicate that you want to use all Templates associated with the Content Type for the Site whose **Publish** value is *Default*.

- 12** In **Item Filter**, leave *public*, since you want to include all Content Items in a public State in the Content List.

Your completed editor should appear similar to the following screenshot:

Edit Content List		Errors						
<b>Id</b>	0-21-323							
<b>Name</b>	rffEIncremental							
<b>Description</b>	Site Root Incremental - Enterprise Investments							
<b>Url</b>	/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_assembly_context=301&sys_c							
<b>Type</b>	<input type="radio"/> Normal <input checked="" type="radio"/> Incremental							
<b>Edition Type</b>	Automatic							
<b>Generator</b>	Java/global/percussion/system/sys_SearchGenerator							
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>query</td> <td>           select rx:sys_contentid, rx:sys_folderid from nt:base where jcr:path like //Sites/Enterprise            (Required) The JSR-170 sql style query to use in finding the base items for the content list.         </td> </tr> </tbody> </table>		Name	Value	query	select rx:sys_contentid, rx:sys_folderid from nt:base where jcr:path like //Sites/Enterprise (Required) The JSR-170 sql style query to use in finding the base items for the content list.		
Name	Value							
query	select rx:sys_contentid, rx:sys_folderid from nt:base where jcr:path like //Sites/Enterprise (Required) The JSR-170 sql style query to use in finding the base items for the content list.							
<b>Template Expander</b>	Java/global/percussion/system/sys_SiteTemplateExpander							
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>siteid</td> <td>301 The id of the site to use when expanding the templates. If not specified this may be provided in the request.</td> </tr> <tr> <td>default_template</td> <td>This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.</td> </tr> </tbody> </table>		Name	Value	siteid	301 The id of the site to use when expanding the templates. If not specified this may be provided in the request.	default_template	This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.
Name	Value							
siteid	301 The id of the site to use when expanding the templates. If not specified this may be provided in the request.							
default_template	This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.							
<b>Item Filter</b>	public							
<input type="button" value="Save"/> <input type="button" value="Cancel"/>								

Figure 220: Registration for rffEIncremental Content List

13 Click [Save].

## Defining the Unpublish Content List

In addition to defining new and updated content that Rhythmyx publishes to your site, Content Lists also define the content that Rhythmyx removes from your Site, or unpublishes. Define a separate Unpublish Content List for each Edition.

A Content List for unpublishing defines a list of Content Items that will be removed from the published Site. An unpublish Content List differs from Publishing Content Lists in the following ways:

- The Content List URL includes the HTML parameter `sys_publish=unpublish`. This parameter instructs the Rhythmyx Publisher to remove Content Items returned by the Content List from the Site.
- The Content List uses the `sys_PublishedSiteItems` Content List Generator. This generator returns all Content Items currently published on the Site specified in the Content List. You must use this Content List Generator in conjunction with the `sys_SiteTemplateExpander`.
- The Content List uses the unpublish Item Filter. This Item Filter consists of the `sys_filterByPublishableFlag` Item Filter Rule, with the value of the `sys_flagValues` parameter set to `u`. In general, all Workflow States other the Public and Quick Edit have this value.

The result of this configuration is that all Content Items on the specified Site that are in States other than Publish or Quick Edit will be removed from the Site.

Since we have covered how to open the Edit Content List page and described its fields in depth in the topic *Defining the Full Binary Content List* (see page 297), see that topic for more information about the following steps.

To define the Unpublish Content List:

- 1 Open a new Edit Content List page.
- 2 In **Name**, enter something similar to (but not the same as) *rffEiUnpublish*.
- 3 In **Description**, enter *Delete any published pages of items in an archive state - Enterprise Investments*.
- 4 Change the **URL** to:  

```
/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_publish=unpublish&sys_assembly_context=301&sys_contentlist=rffEiUnpublish.
```

Replace *rffEiUnpublish* at the end of the URL with the name you have given your Content List.
- 5 In our other Content List definitions, we have set the value of the `sys_publish` HTML parameter to *publish*; in this definition we set it to *unpublish*, so the Publisher application knows to remove the items in this Content List from the Site.
- 6 Leave **Type** as *Normal*.
- 7 Leave **Edition Type** as *Automatic*.
- 8 In the **Generator** field choose *Java/Global/percussion/system/sys\_PublishedSiteItems* from the drop down list.
- 9 In the **Template Expander** drop down list select *Java/Global/percussion/system/sys\_SiteTemplateExpander* since we are using Site Folder publishing. Fields for the parameters `siteid` and `default_template` appear below the **Template Expander** drop list.

---

Note that you must use the `sys_PublishedSiteItems` Content List Generator in conjunction with the `sys_SiteTemplateExpander`. The **Template Expander** defines the Site whose Content Items the `sys_PublishedSiteItems` will return.

---

- 10 In the `siteid` field below the **Template Expander** field enter *301*.
- 11 Leave the `default_template` field blank to indicate that you want to unpublish all Templates associated with the Content Type for the Site whose **Publish** value is *Default*.
- 12 In **Item Filter**, choose *unpublish*, since you want to include all Content Items in States other than *Public* or *Quick Edit*.

Your completed editor should appear similar to the following screenshot:

Edit Content List		Errors
<b>Id</b>	0-21-316	
<b>Name</b>	rffEIUnpublish	
<b>Description</b>	Delete any published pages of items in an archive state - Enterprise Investments	
<b>Url</b>	/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_publish=unpublish&sys_ass	
<b>Type</b>	<input checked="" type="radio"/> Normal <input type="radio"/> Incremental	
<b>Edition Type</b>	Automatic	
<b>Generator</b>	Java/global/percussion/system/sys_PublishedSiteItems	
	<b>Name</b>	<b>Value</b>
<b>Template Expander</b>	Java/global/percussion/system/sys_SiteTemplateExpander	
	<b>Name</b>	<b>Value</b>
	siteid	301 The id of the site to use when expanding the templates. If not specified this may be provided in the request.
	default_template	This has one of three values. The value all or unspecified means to include all default templates. The value dispatch means to include default templates that use the dispatch assembly plugin. The value none means to ignore default templates.
<b>Item Filter</b>	unpublish	

Figure 221: Registration for rffEIUnpublish Content List

**13** Click [Save].

## Defining Contexts and Location Schemes

A Context is a location or environment in which content is published or assembled. For example, the publishing location for Enterprise Investments content is one context, and the location for previewing assembled content before publishing it is another context.

Location Schemes are associated with specific contexts, and specify the rules for configuring the addresses of content items. Location schemes have various uses. Two of the main uses of location schemes are:

- telling the Publisher where to publish content on a file system.
- creating URLs so that content items can link to each other when they appear in a browser.

Your specification requires three Contexts:

- A preview Context - Rhythmyx provides this Context. It enables previewing of content by linking to referenced content such as images and other pages.
- A publishing Context - We are including this Context because it is necessary for all users of file system publishing. It tells the Publisher where to deliver Content Items that are ready to be published to the file system.
- A Context for assembling items - We are including this Context because it creates links to other Content Items. This address is different from the publishing Context for two reasons. First, because even for content published to a file system, URL paths to the Content Item may differ from the location where the file resides (depending on the Web Server's addressing conventions). Second, for other types of publishing, content may be stored and retrieved using a different approach than a pre-generated location (for example, the system may pass a request parameter to the Web application to retrieve the correct content.)

Your specification also requires two Location Schemes:

- A location scheme that generates the path for publishing each item on the file system. We will associate this Location Scheme with the publishing Context. It will create the delivery address for Content Items that are ready to be published.
- A location scheme that generates the URL for each item (so items can link to one another). We will associate this Location Scheme with the Context for assembling items. It will generate the Site address of published Content Items for the browser to use.

In the following topics, we use the Publishing Administrator to create the Contexts and Location schemes which are already defined by the following FastForward Content Lists and Location Schemes:

Contexts:

- *Publish*
- *Site Folder Assembly*

Location Schemes

- *Generic (for Publish Context)*
- *Generic (for Site Folder Assembly Context)*



FastForward includes additional Location Schemes that we will not recreate in this section.

---

*Note: You cannot create Contexts and Location Schemes with the above names since they already exist in FastForward. Instead, create similar Contexts and Location Schemes included in your implementation plan or copy our steps but give your Contexts and Location Schemes different names.*

---

## Creating the Publish Context and its Generic Location Scheme

Begin by creating your Publish Context.

To create a Publish Context:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Click the Publishing tab.
- 3 Click the By Name link under the Contexts heading in the left navigation pane.  
Content Explorer displays the Contexts page.
- 4 Click [**New Context**].
- 5 Content Explorer displays the Edit Context Page.

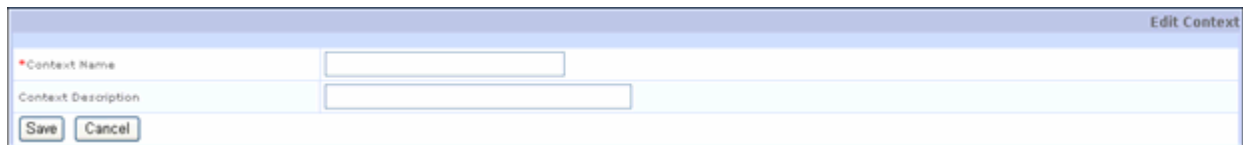


Figure 222: Edit Context page

- 6 In **Context Name**, enter something similar to *Publish*, to distinguish your entry from FastForward's *Publish* entry.
- 7 In **Context Description**, enter *Create the appropriate path for the publishing location, related to but not identical with the assembly location.*
- 8 Click [**Save**].

The Edit Context page closes. The Context page lists your new Context.

Now add your Location Scheme to the Publish Context.

To add the Location Scheme:

- 1 Click your Publish Context title in the Contexts page.

Content Explorer opens the Edit Context page. It now has a new section for adding Location Schemes.

Figure 223: Edit Context Page with Location Scheme fields

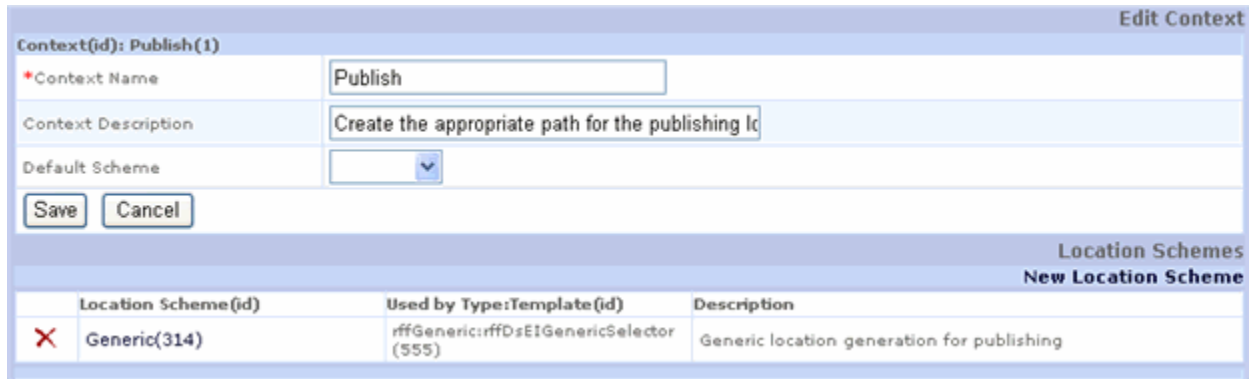
**2** Click [**New Location Scheme**].

Content Explorer opens the Edit Location Scheme page.

Figure 224: Edit Location Scheme page

- 3** In **Name**, enter *Generic*.
- 4** In **Description**, enter *Generic location generation for publishing*.
- 5** The **Generator** drop list includes several default location scheme generator extensions (java plugins). Choose *sys\_JexlAssemblyLocation*, which is compatible with Rhythmyx's current JEXL-based Assembly engine. The other generators are included for special cases and backwards compatibility.
- 6** In **Content Type**, choose *Generic*. This specifies that this is the default Location Scheme for the *Generic* Content Type when it uses the Template selected in the next field. However, we will make it the default Location Scheme for all Content Types without a Location Scheme assigned in this Context.
- 7** In **Template Type**, choose *D - EI Generic*. Now this is the default Location Scheme for the *Generic Content Type* when it uses the *D - EI Generic* Template.
- 8** Click [**Save**].

The *Generic* Location Scheme is now listed as a Location Scheme for the *Publish* Context. In addition the Context registration section displays a field for a **Default Scheme**. The editor displays this field after at least one Location Scheme is registered. It lets you choose a default Location Scheme to apply to all outputs assigned to this Context that use Content Types and Templates that are not assigned a specific Location Scheme.




Location Schemes		
New Location Scheme		
Location Scheme(id)	Used by Type:Template(id)	Description
 Generic(314)	rffGeneric:rffDsEIGenericSelector (555)	Generic location generation for publishing

Figure 225: Publish Context with Generic Location Scheme added

- 9 In Default Scheme, choose *Generic* (the only choice available).

Now all content published to the *Publish* Context will use the *Generic* Location Scheme.



Location Schemes		
New Location Scheme		
Location Scheme(id)	Used by Type:Template(id)	Description
 Generic(314)	rffGeneric:rffDsEIGenericSelector (555)	Generic location generation for publishing

Figure 226: Publish Context with Default Scheme

The location scheme generator that you have specified, `sys_JexlAssemblyLocation`, requires that you pass a parameter that holds a JEXL expression for creating the publishing path.

To create a Location Scheme Parameter:

- 1 Click *Generic* in the Location Scheme column to open its Edit Location Scheme page.

Now the page includes a section for adding Location Scheme parameters.

The screenshot shows the 'Edit Location Scheme' interface. At the top, it says 'Location Scheme(id): Generic (323)'. The form contains the following fields:

- Name:** Generic
- Description:** Generic location generation for publishing.
- Generator:** sys\_JexlAssemblyLocation
- Content Type:** Generic
- Template Type:** D- EI Generic

Below the form are 'Save' and 'Cancel' buttons. Underneath is a section titled 'Location Scheme Parameters' with a sub-header 'New Location Scheme Parameter'. It contains a table with the following structure:

Name(id)	Type	Sequence	Value
No entries found.			

Figure 227: Edit Location Scheme page with section for adding parameters

**2** Click New Location Scheme Parameter.

Content Explorer opens the Edit Location Scheme Parameter page.

The screenshot shows the 'Edit Location Scheme Parameter' interface. The form contains the following fields:

- Name:** (empty text box)
- Type:** String
- Sequence:** (empty text box)
- Value:** (empty text area)

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

Figure 228: Edit Location Scheme Parameter page

**3** You must enter a value in **Name**. Enter *expression*.

**4** The **Type** drop list includes two options:

- *String* - Indicates that the value you enter in **Value** is a text string.
- *Backend Column* - Indicates that the value you enter in **Value** is a backend column.

Leave the **Type** as *String*, since you will enter a text string in **Value**.

**5** The value you enter in **Sequence** is the order in which multiple Location Scheme Parameters are arranged so that they compose the folder path or URL correctly. Since you are only entering one Location Scheme parameter, enter *1*.

**6** The string or backend column that you enter in **Value** composes part of the folder path or URL or, as in this case, the entire folder path or URL. Enter the JEXL expression:

```
$sys.pub_path + $sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String +
$rx.location.getFirstDefined($sys.item, 'rx:activeimg_ext,rx:sys_s
uffix', '.html')
```

This JEXL expression uses bindings (variables or functions) already defined for Rhythmyx. See *Bindings* (see page 136) for more information. Content Type names in Rhythmyx are expressed in the JEXL expression with rx: preceding them. The bindings used in the above expression have the following values and functions:

Binding	Value
\$sys.pub_path	<p>sys.pub_path holds the file system path to which content is published. It consists of the path assigned to the item's Content Explorer folder in its folder properties or if no path is assigned, the actual folder path holding the Content Item under the Site folder root in Content Explorer.</p> <p>Note: Do not use \$sys.site.path in place of \$sys.pub_path. \$sys.site.path takes the value of sys_siteid. If a user action causes sys_siteid to be set to a different site immediately before a publishing run, the edition will be published to the wrong site.</p>
\$sys.template.prefix	sys.template.prefix holds the default Template's prefix value, if a prefix has been entered.
\$sys.item	Holds the current Content Item's fields and children.
\$sys.item.getProperty(rx:sys_contentid').String	getProperty returns the value of the specified property in the current Content Item. String indicates that the value is returned as a text string. Therefore this binding returns the Content ID of the current content item in string format.
\$rx.location.getFirstDefined(\$sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')	<p>\$rx.location returns a hypertext link.</p> <p>\$rx.location.getFirstDefined returns the value of the first defined property for \$sys.item in the list that follows it in the parenthesis. Therefore, if the activeimg_ext field is filled, that value is returned; otherwise, if the sys_suffix field is filled, that value is returned; otherwise, the value ".html" is returned.</p>

For example, let's demonstrate how the Location Scheme parameter composes the publishing path when you publish the item *About EI HomePage Image (NYSE Papers).jpg* which is located in the Folder Path *AboutEnterpriseInvestments* to the Enterprise Investments Site:

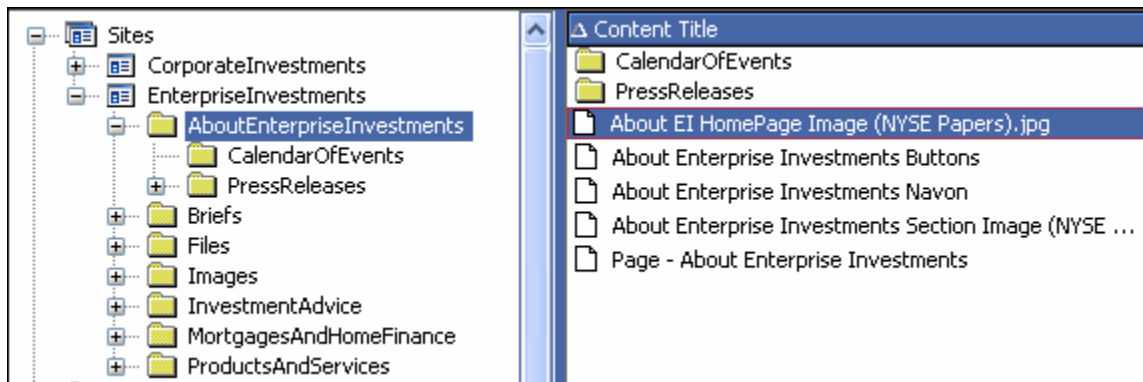


Figure 229: *About EI HomePage Image (NYSE Papers).jpg*

The Location Scheme parameter should create the following publishing path:

- Since no path is assigned to the *AboutEnterpriseInvestments* folder, `sys.pub_path` equals `/AboutEnterpriseInvestments`
- Since none of the Templates associated with the `rffImage` Content Type have a Prefix, `$sys.template.prefix` has no value.
- Since the Content ID of *About EI HomePage Image (NYSE Papers).jpg* is 481, `$sys.item.getProperty(rx:sys_contentid').String` equals `481`.
- Since the Image Content Type does not have an `activeimg_ext` field (`activeimg_ext` is a field in the Navimage Content Type), `$rx.location.getFirstDefined` takes the value of the next field, `sys_suffix` (`sys_suffix` is a field in the Image Content Type), which equals `.jpg` in the content item.

When we put the parts of the path together as indicated in the Location Scheme parameter, the result is:

```
/AboutEnterpriseInvestments/item481.jpg
```

However, for a publishing Context (the Context associated with the Edition Content list and *not* specified in the `sys_assembly_context` parameter in the Content List URL) the publisher inserts the **Publishing Root Location** (in the Site registration) assigned to the Site to which the content item is published in front of the location.

Since the **Publishing Root Location** for the Enterprise Investments Site is `../EI_Home.war`, the complete location to which the content is published is

```
../EI_Home.war/AboutEnterpriseInvestments/item481.jpg
```

By making changes to this JEXL expression, which is supplied by FastForward, you can customize the paths to which you publish your content items.

## 7 Click [Save].

The Location Scheme parameter is listed under the Location Scheme and is used whenever a Content Type is published using this Context.

Name(id)	Type	Sequence	Value
expression	String	1	<code>`\${sys.pub_path} + \${sys.template.prefix} + 'item' + \${sys.item.getProperty('rx:sys_contentid')}.String + \${rx.location.getFirstDefined('\${sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')}</code>

Figure 230: Location Scheme Parameter added

- 8 Click [Save].

The Edit Location Scheme page closes. Content Explorer displays the Edit Context page.

Location Scheme(id)	Used by Type:Template(id)	Description
Generic(314)	rffGeneric:rffDsEIGenericSelector (555)	Generic location generation for publishing

Figure 231: Publish Context with Default Location Scheme

- 9 Click [Save] to return to the Contexts editor and enter your *SiteFolderAssembly* Context.

## Creating the Assembly Context and its Generic Location Scheme

Many of the steps and fields in this topic have already been explained in the previous topic, Creating the Publishing Context and its Generic Location Scheme. See that topic for details about the fields and values discussed in the following procedure.

To create the Assembly Context:

- 1 Open a new Edit Context Page.
- 2 In Context Name, enter a name similar to *Site Folder Assembly*.

- 3 In **Context Description**, enter *Create the appropriate path for the site folder assembly location*.
- 4 Click [**Save**].

The Edit Context page closes. Your new Context is listed on the Contexts page. Now add your Location Scheme to the *Site Folder Assembly* Context.

To add the Location Scheme:

- 1 Click *Site Folder Assembly* on the Contexts page.  
Content Explorer displays the Edit Context page with a new section for adding Location Schemes.
- 2 Click [**New Location Scheme**].  
Content Explorer displays the Edit Location Scheme page.
- 3 In **Name**, enter *Generic*.
- 4 In **Description** enter *Generic Location Scheme for assembly*.
- 5 The **Generator** drop list includes several default location scheme generator extensions (java plugins). Choose `sys_JexlAssemblyLocation`, which is compatible with Rhythmyx's JEXL-based Assembly engine. The other generators are included for special cases and backwards compatibility.
- 6 In **Content Type**, choose *Generic*. This specifies that this is the default Location Scheme for the *Generic* Content Type when it uses the Template selected in the next field. However, we will make it the default Location Scheme for all Content Types without a Location Scheme assigned to this Context.
- 7 In **Template Type**, choose *D - EI Generic*. Now this is the default Location Scheme for the *Generic Content Type* when it uses the *D - EI Generic* Template.
- 8 Click [**Save**].

The *Generic* Location Scheme is now listed as a Location Scheme for the *Site Folder Assembly* Context. In addition the Context registration section now displays a field for a **Default Scheme**. The editor displays this field after at least one Location Scheme is registered. It lets you choose a default Location Scheme to apply to all outputs that use this Context but use Content Types and Templates that are not assigned a specific Location Scheme.

- 9 In **Default Location Scheme**, choose *Generic* (the only choice available).

Location Scheme(id)	Used by Type:Template(id)	Description
Generic(315)	rffGeneric:rffDsEIGenericSelector (555)	Generic location scheme for assembly

Figure 232: Generic Location Scheme



Now all content published to the *Site Folder Assembly* Context will use the *Generic* Location Scheme.

The location scheme generator that you have specified, `sys_JexlAssemblyLocation`, requires that you pass a parameter that holds a JEXL expression for creating the URLs that link items to one another.

To create a Location Scheme Parameter:

- 1 Click *Generic* in the Location Scheme column to open its Edit Location Scheme page.  
Now the page displays a section for adding Location Scheme parameters.
- 2 Click New Location Scheme Parameter.  
Content Explorer displays the Edit Location Scheme Parameter page.
- 3 You must enter a value in **Name**. Enter *expression*.
- 4 Leave the **Type** as *String*, since you will enter a text string in **Value**.
- 5 The value you enter in **Sequence** is the order in which multiple Location Scheme Parameters are arranged so that they compose the folder path or URL correctly. Since you are only entering one Location Scheme parameter, enter *1*.
- 6 The string or backend column that you enter in **Value** composes part of the folder path or URL or, as in this case, the entire folder path or URL. Enter the JEXL expression:

```
if ($sys.crossSiteLink) {$prefix = $sys.site.url;} else {$prefix =
$sys.variables.rxs_urlroot}; $prefix + $sys.pub_path +
$sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String +
$rx.location.getFirstDefined($sys.item, 'rx:activeimg_ext,rx:sys_suffix',
'.html')
```

This is almost the same JEXL expression that you used for the Location Scheme parameter for the Publish Context. However, it begins with the JEXL `if . . . else` function to define the part of the URL that the browser uses in front of the folder path to locate items. This enables content items in the Site to link to one another when a browser displays them.

In this case, if the link is a cross-site link (a link to a different Site), the expression returns the root of the remote Site. If it is not a cross site link, the function returns the root URL of the current Site (`rxs_urlroot` is a **Context Variable** (see "Converting References to Static Files" on page 174)) and gives it the value `/EI_Home` for this Site and Context. The remainder of the expression is exactly the same as that used in the Location Scheme parameter for the Publish Context. So the full path for Content Item with contentid 481 would appear as:

```
/EI_Home/AboutEnterpriseInvestments/item481.jpg
```

See *Creating the Publish Context and its Generic Location Scheme* for an explanation of the rest of the expression.

**7** Click **[Save]**.

The Location Scheme parameter is listed under the Location Scheme and is used whenever a Content Type is assigned to this Context.

**Location Scheme(id): Generic(315)** Edit Location Scheme

\*Name:

Description:

\*Generator:

\*Content Type:

\*Template Type:

**Location Scheme Parameters**  
**New Location Scheme Parameter**

Name(id)	Type	Sequence	Value
<span style="color: red;">✗</span> expression	String	1	if (\$sys.crossSiteLink) {\$prefix = \$sys.site.url;} else {\$prefix = \$sys.variables.rxs_urroot;} \$prefix + \$sys.pub_path + \$sys.template.prefix + 'item' + \$sys.item.getProperty('rx:sys_contentid').String + \$rx.location.getFirstDefined(\$sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')

Figure 233: Generic Location Scheme

**8** Click **[Save]**.

The Edit Location Scheme page closes and Content Explorer displays the Edit Context page.

**Context(id): Site Folder Assembly(301)** Edit Context

\*Context Name:

Context Description:

Default Scheme:

**Location Schemes**  
**New Location Scheme**

Location Scheme(id)	Used by Type:Template(id)	Description
<span style="color: red;">✗</span> Generic(315)	rffGeneric:rffDsEIGenericSelector (555)	Generic location scheme for assembly

Figure 234: Generic Location Scheme

**9** Click **[Save]** to save the Context.


## Checking for Errors in the Location Scheme

If you want to check your Location Schemes for errors prior to publishing, do the following:

- To check a publishing Location Scheme view your Edition's Content Lists prior to publishing.
- To check an assembly Location Scheme preview a Content Item that includes links to related content and images using a Page Template. See if the links to related content work and if the images appear. If not, check the page source to see errors in the links.

To check a Publishing Location Scheme:

- 1 In the Publishing tab, go to the Editions page.
- 2 Click the name of the Edition containing the Location Schemes that you want to check.

At the bottom of the Edit Editions Properties page, the Content Lists that you have added for the Edition are listed. To the right of each listing is a Preview button .

Edit Edition Properties

Edition(id): Full Enterprise Investments(301)

<b>*Edition Name</b>	<input type="text" value="Full Enterprise Investments"/>
<b>Description</b>	<input type="text" value="Publish all items in public state"/>
<b>*Destination Site</b>	<input type="text" value="Enterprise Investments"/> ▾
<b>Edition Type</b>	<input type="text" value="Automatic"/> ▾
<b>Recovery Publication(id)</b>	<input type="text" value=""/> (Recovery only)
<b>Mirror Source Site</b>	<input type="text" value="--Choose--"/> ▾ (Mirror only)

Edit Edition: Allowed Content
Add Content List




	Content List Name (id)	Sequence	Authorization Type	Context	Preview
✘	rffEiUnpublish (316)	1	All Public Content	Publish	
✘	rffEiFullBinary (310)	2	All Public Content	Publish	
✘	rffEiFullNonBinary (311)	3	All Public Content	Publish	

Figure 235: Full Enterprise Investments Edition with Content Lists

- 3 In this example, we have intentionally created an incorrect publish Location Scheme. We click the Preview button for the rffEiFullBinary Content List to see the XML file representing the Content List. Notice that the `<location></location>` tags only include the item file names but not their paths. The paths must be present for the items to be delivered to the correct publishing locations. Once we see that the path is missing, we return to the Publish Context and correct this Location Scheme to include the publishing path.

```

<?xml version="1.0" encoding="utf-8" ?>
- <contentlist context="1" deliverytype="filesystem">
- <contentitem contentid="383" revision="1" unpublish="no" variantid="506">
  <title>EI Global Financial Service Fund - allocation.jpg</title>
  <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sy
- <delivery>
  <location>item383.jpg</location>
</delivery>
  <modifydate>2005-03-24 00:00:00</modifydate>
  <modifyuser>admin1</modifyuser>
  <contenttype>307</contenttype>
</contentitem>
- <contentitem contentid="384" revision="1" unpublish="no" variantid="506">
  <title>EI Global Financial Service Fund - regional mix.jpg</title>
  <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sy
- <delivery>
  <location>item384.jpg</location>
</delivery>
  <modifydate>2005-03-24 00:00:00</modifydate>
  <modifyuser>admin1</modifyuser>
  <contenttype>307</contenttype>
</contentitem>
- <contentitem contentid="385" revision="1" unpublish="no" variantid="506">
  <title>EI Global Financial Service Fund - returns.jpg</title>
  <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sy
- <delivery>
  <location>item385.jpg</location>
</delivery>
  <modifydate>2005-03-24 00:00:00</modifydate>

```

Figure 236: Content List showing Location Scheme error

When corrected, the Content List should appear as follows. In the <delivery></delivery> tags the publish path /Images/Funds/EIGlobalServicesFund/ appears before the item's file name.

```

I
<?xml version="1.0" encoding="utf-8" ?>
- <contentlist context="303" deliverytype="filesystem">
  - <contentitem contentid="383" revision="1" unpublsh="no" variantid="506">
    <title>EI Global Financial Service Fund - allocation.jpg</title>
    <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
      sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=pe
  - <delivery>
    <location>/Images/Funds/EIGlobalServicesFund/item383.jpg</location>
  </delivery>
    <modifydate>2005-03-24 00:00:00</modifydate>
    <modifyuser>admin1</modifyuser>
    <contenttype>307</contenttype>
  </contentitem>
  - <contentitem contentid="384" revision="1" unpublsh="no" variantid="506">
    <title>EI Global Financial Service Fund - regional mix.jpg</title>
    <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
      sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=pe
  - <delivery>
    <location>/Images/Funds/EIGlobalServicesFund/item384.jpg</location>
  </delivery>
    <modifydate>2005-03-24 00:00:00</modifydate>
    <modifyuser>admin1</modifyuser>
    <contenttype>307</contenttype>
  </contentitem>
  - <contentitem contentid="385" revision="1" unpublsh="no" variantid="506">
    <title>EI Global Financial Service Fund - returns.jpg</title>
    <contenturl>http://10.10.10.136:9817/Rhythmyx/assembler/render?
      sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=pe
  - <delivery>
    <location>/Images/Funds/EIGlobalServicesFund/item385.jpg</location>
  </delivery>
    <modifydate>2005-03-24 00:00:00</modifydate>

```

Figure 237: Content List with correct Publish Location Scheme

To check an assembly Location Scheme:

- 1 In Content Explorer, preview a page Template of a Content Item that includes graphics and related content.

Notice that the Context parameter at the end of the URL is sys\_context=0.

- 2 Change the value of sys\_context to the Context ID of the Assembly Location Scheme that you want to check.

First, we show an example of a valid Location Scheme. We change the value of `sys_context` to 301. Note that even though the Location Scheme is correct, the page cannot display the graphics in the Global Template that are not yet published. If you hover the cursor over a page link, the browser displays its correct URL below the page as in the following graphic.

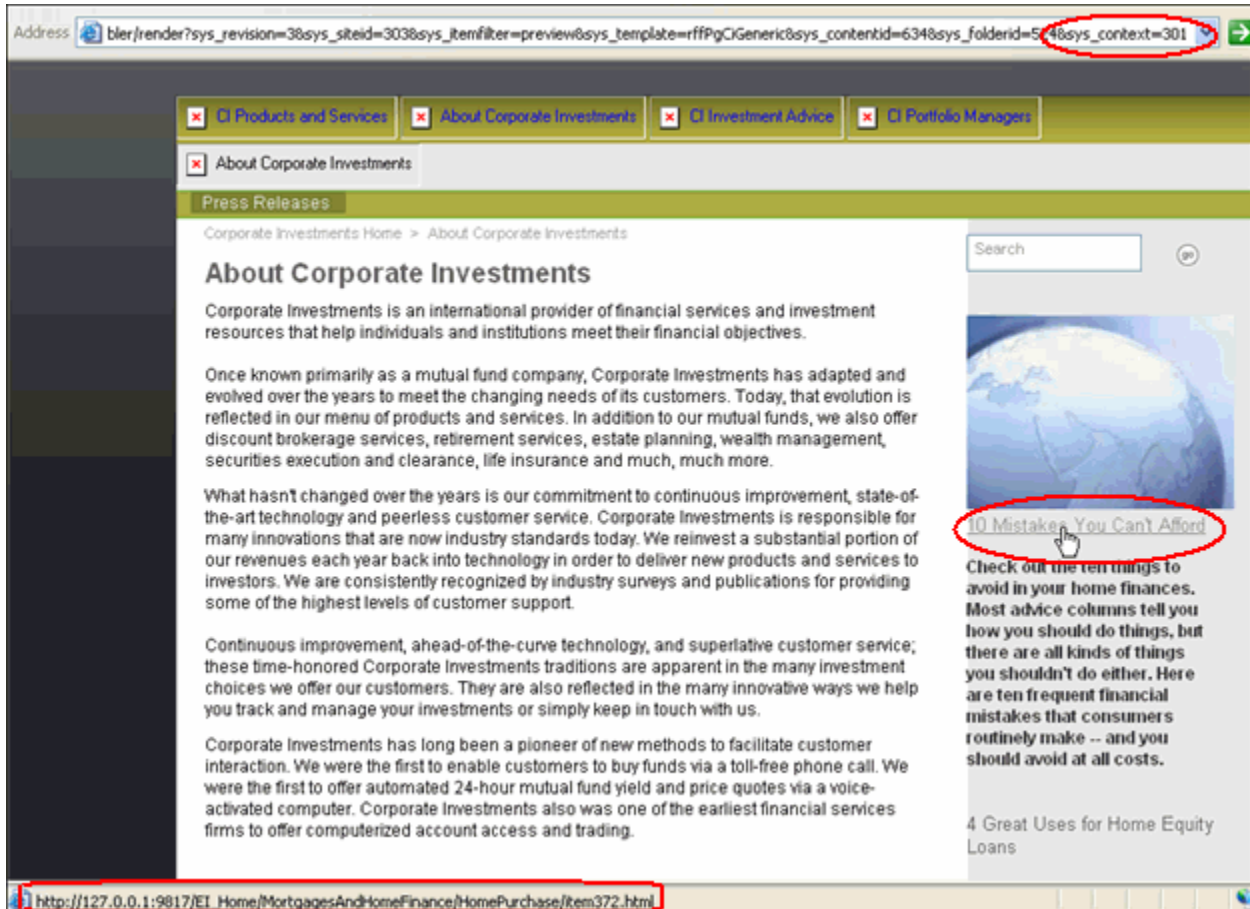
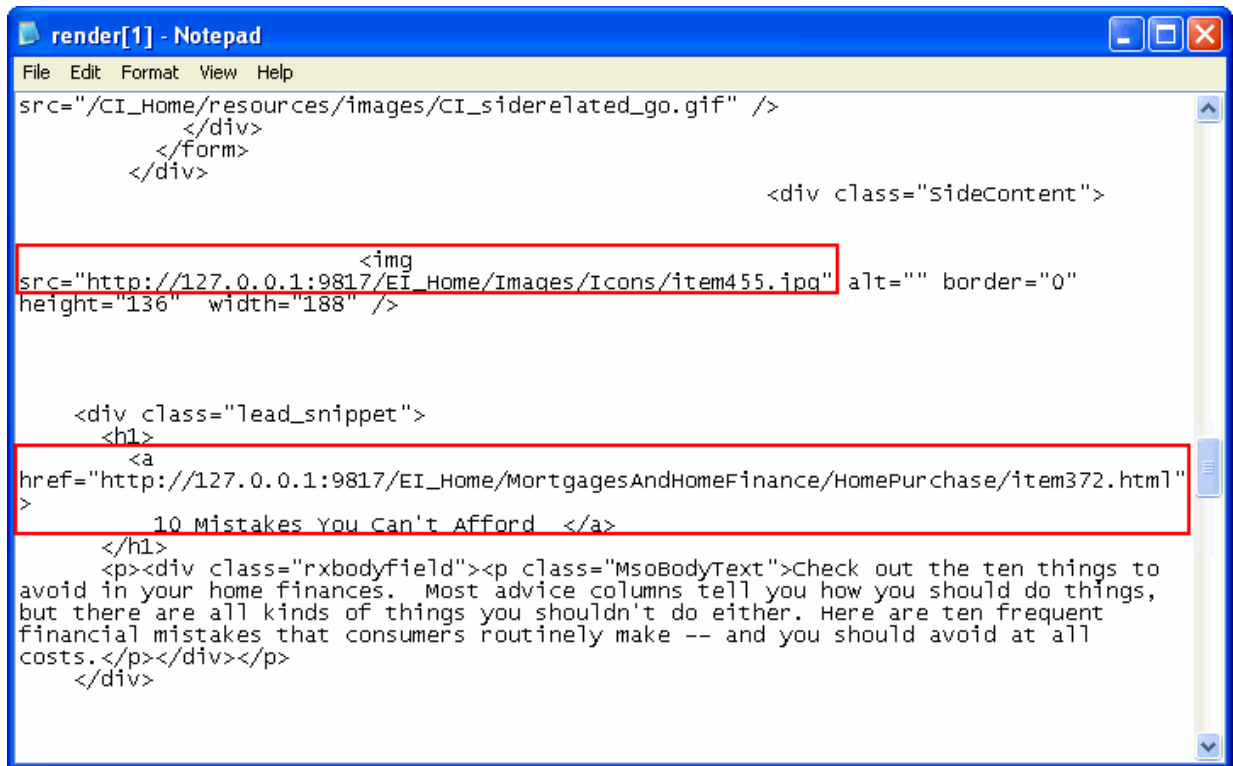


Figure 238: Viewing a Content Item in Preview Context

Now view the source and check a hyperlink <a> tag to confirm that the href attribute specifies a URL that resolves correctly and <img> tag to confirm that the src attribute also contains a URL that resolves correctly. Do not use <a> and <img> tags associated with the Global Template to confirm that the address is correct - they should hold different addresses. The following source shows the addresses for the blue globe graphic inserted into the Content Item and the "10 Mistakes You Can't Afford" link below it. The URLs in the tags are correct and confirm that the Location Scheme is valid.



```

render[1] - Notepad
File Edit Format View Help
src="/CI_Home/resources/images/CI_siderelated_go.gif" />
  </div>
  </form>
</div>
                                <div class="sideContent">
                                

                                <div class="lead_snippet">
                                <h1>
                                <a
href="http://127.0.0.1:9817/EI_Home/MortgagesAndHomeFinance/HomePurchase/item372.html"
>
                                10 Mistakes You Can't Afford </a>
                                </h1>
                                <p><div class="rxbodyfield"><p class="MsobodyText">Check out the ten things to
avoid in your home finances. Most advice columns tell you how you should do things,
but there are all kinds of things you shouldn't do either. Here are ten frequent
financial mistakes that consumers routinely make -- and you should avoid at all
costs.</p></div></p>
                                </div>

```

Figure 239: Source code for valid assembly Location Scheme

Now we change the value of the sys\_context parameter to 304. This Assembly Context uses an invalid Location Scheme Parameter. By looking at the way the page handles this error, you can determine whether the error is in the JEXL code or literal text in the parameter. The page appears as the following graphic. Notice that it cannot link to the blue globe graphic, and when the cursor hovers over a page link, the browser displays an incomplete URL at the bottom of the page. The incomplete URL suggests that the error is in the JEXL code. If it were in a piece of literal text included in the parameter, when the cursor hovers over a link, the browser would probably display the incorrect URL below the page.

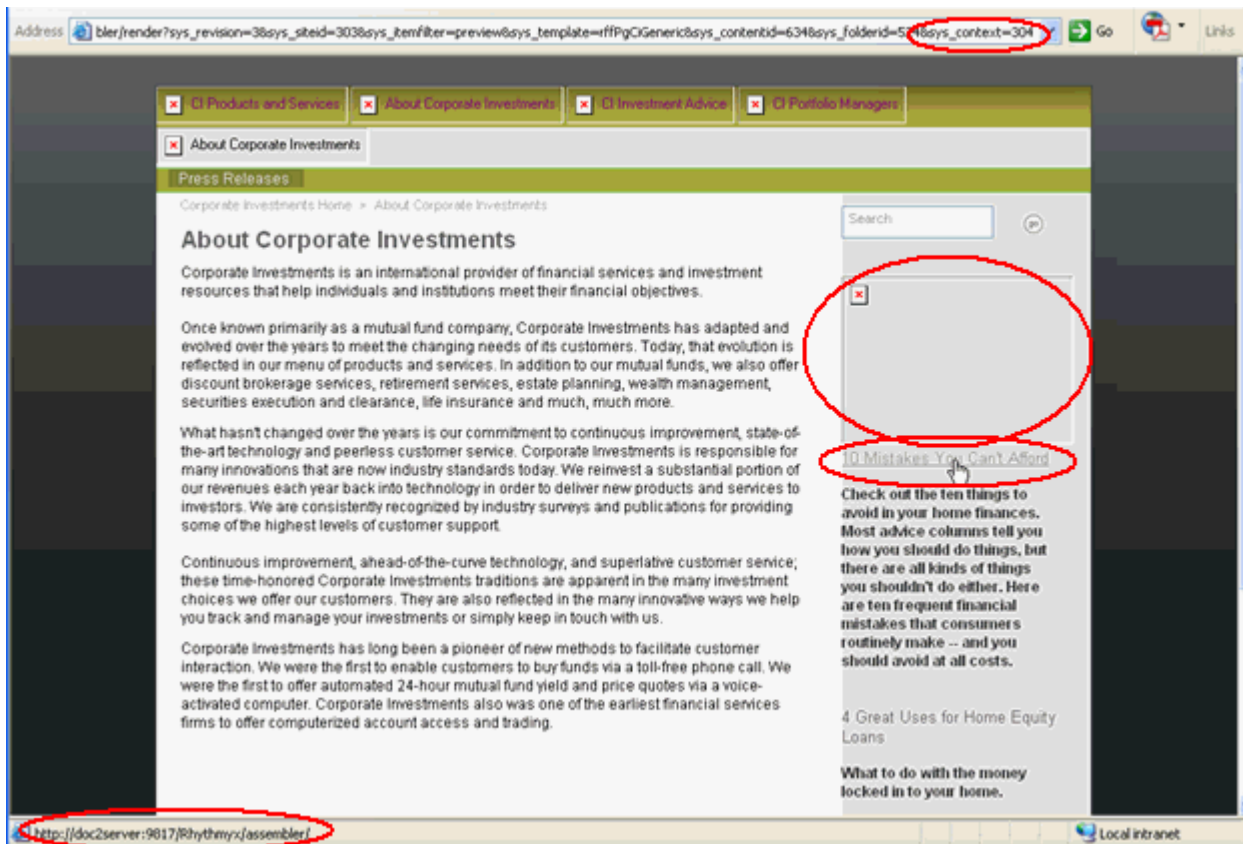
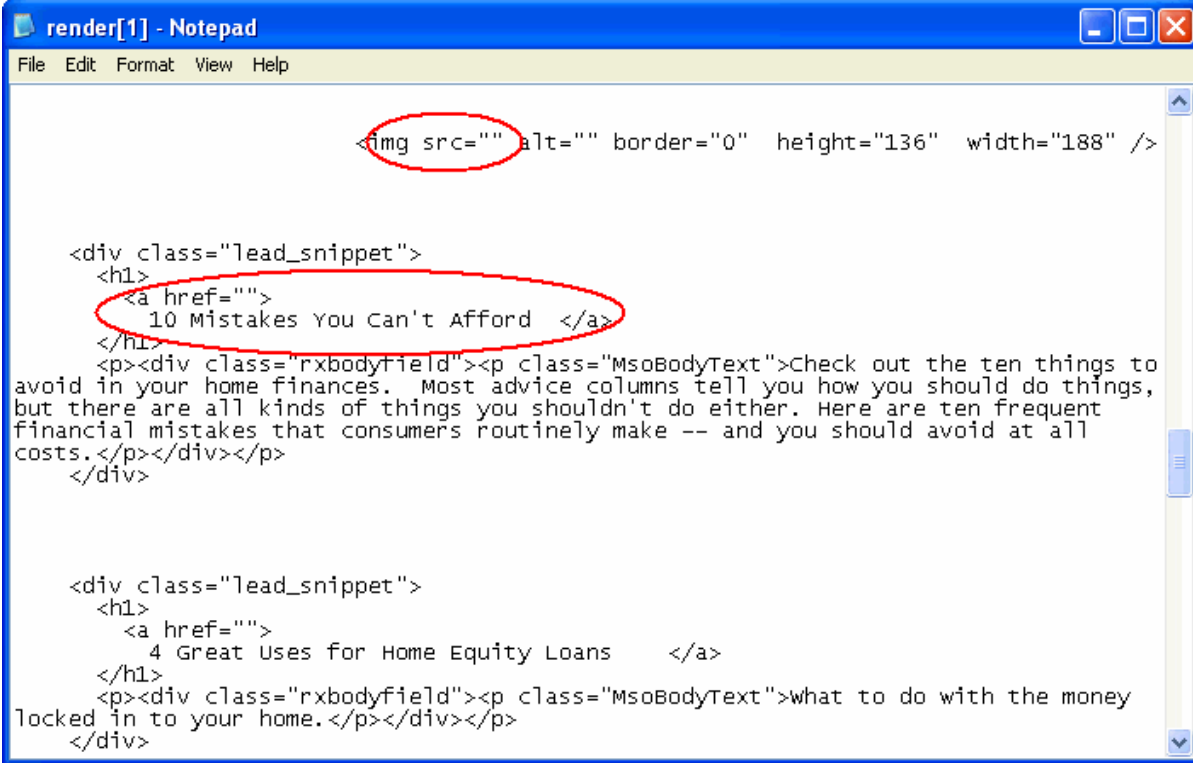


Figure 240: Preview with invalid assembly Location Scheme



Now view the source and check the href attribute of the <a> tag and src attribute of the <img> tag that we looked at with the valid Location Scheme Parameter. Notice that the attributes do not have a value. The absence of a value in these attributes indicates that the error is in the JEXL code.



```

render[1] - Notepad
File Edit Format View Help

<img src="" alt="" border="0" height="136" width="188" />

<div class="lead_snippet">
  <h1>
    <a href="">
      10 Mistakes You Can't Afford </a>
    </h1>
    <p><div class="rxbodyfield"><p class="MsoBodyText">Check out the ten things to
avoid in your home finances. Most advice columns tell you how you should do things,
but there are all kinds of things you shouldn't do either. Here are ten frequent
financial mistakes that consumers routinely make -- and you should avoid at all
costs.</p></div></p>
</div>

<div class="lead_snippet">
  <h1>
    <a href="">
      4 Great Uses for Home Equity Loans </a>
    </h1>
    <p><div class="rxbodyfield"><p class="MsoBodyText">what to do with the money
locked in to your home.</p></div></p>
</div>

```

Figure 241: Source code for invalid assembly Location Scheme

Check the Location Scheme Parameter. One of the following errors has occurred:

- The test condition of the if...else function is not enclosed in parentheses; for example, if the Location Scheme Parameter were specified as if \$sys.crossSiteLink... rather than if (\$sys.crossSiteLink)....
- The string value being tested is not enclosed in quotation marks. For example, if \$sys.site.path=\\EnterpriseInvestments\InvestmentAdvice.. would result in an empty href or src attribute. The correct syntax is if \$sys.site.path='\\EnterpriseInvestments\InvestmentAdvice' .....
- The consequent statements do not each end with a semicolon (";"); for example,

```

if ($sys.crossSiteLink) $prefix = $sys.site.url else
$prefix = $sys.variables.rxs_urlroot

```

is incorrect. The statement should be:

```

if ($sys.crossSiteLink) $prefix = $sys.site.url; else
$prefix = $sys.variables.rxs_urlroot;

```

Now look at the Location Scheme Parameter. The first function is spelled wrong. It should read `$rx.cond.choose` but reads `$rx.aond.choose`.

If you correct the function, but add an error in the literal code, the page handles the error differently. For example, suppose the `if...else` function was specified correctly but the Location Scheme parameter specified + `'jpg'` at the end.

```
if ($sys.crossSiteLink) $prefix = $sys.site.url; else $prefix =
$sys.variables.rxs_urlroot; $prefix + $sys.pub_path +
$sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String +
$rx.location.getFirstDefined($sys.item, 'rx:activeimg_ext,rx:sys_su
ffix', '.html')+'.jpg'
```

This parameter is invalid because each location will append `jpg` at the end of a file that already has a suffix. However, the system understands how to generate the expression.

If you refresh the page that uses `sys_contextid=304`, it appears the same as it did with the other invalid Location Scheme Parameter, but when you hover the cursor over a page link, the browser displays a generated URL with an incorrect suffix on the filename below the page (`item372.htmljpg`).

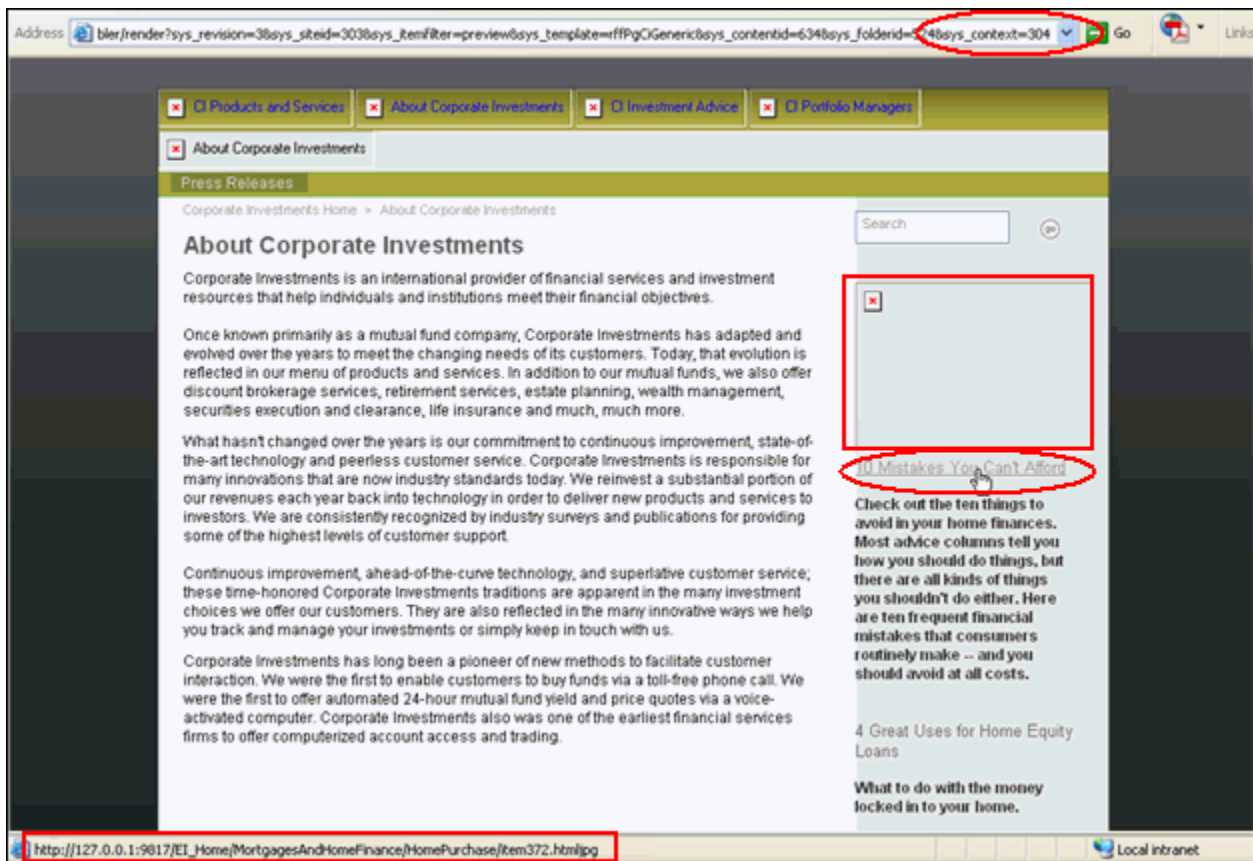
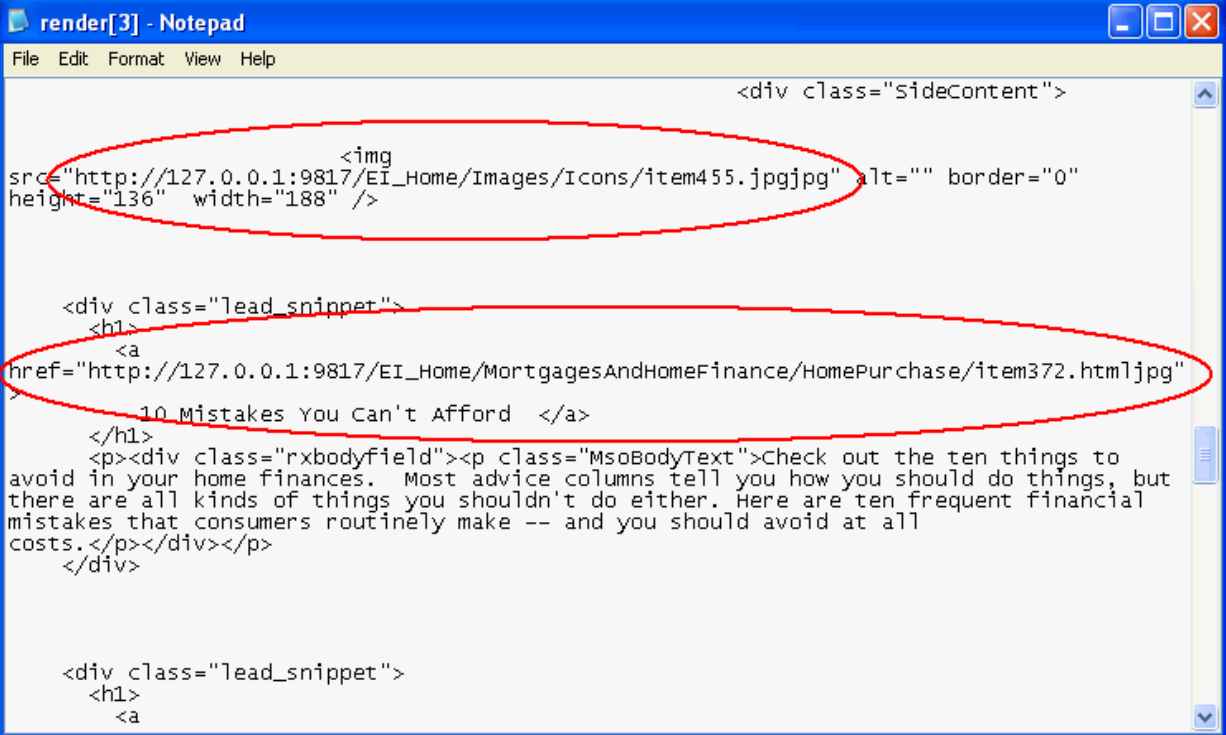


Figure 242: Page with error in Location Scheme Parameter suffix

If you look at the source and view the same hyperlink reference `<a href>` tag and `<img src>` tag that we looked at previously, you will see that the tags include the URL with the invalid suffix in their parentheses.

A screenshot of a Notepad window titled "render[3] - Notepad". The window displays HTML source code. Two red circles are drawn around specific parts of the code. The first circle highlights the `` tag, where the URL ends with ".jpgjpg". The second circle highlights the `<a href="http://127.0.0.1:9817/EI_Home/MortgagesAndHomeFinance/HomePurchase/item372.htmljpg">` tag, where the URL ends with ".htmljpg". The rest of the code includes a `<div class="lead_snippet">` block containing an `<h1>` tag with the text "10 Mistakes You Can't Afford", followed by a paragraph of text about home finances, and another `<div class="lead_snippet">` block starting with an `<h1>` tag.

```
render[3] - Notepad
File Edit Format View Help
<div class="sideContent">
    

    <div class="lead_snippet">
    <h1>
    <a
href="http://127.0.0.1:9817/EI_Home/MortgagesAndHomeFinance/HomePurchase/item372.htmljpg"
>
    10 Mistakes You Can't Afford </a>
    </h1>
    <p><div class="rxbodyfield"><p class="MsoBodyText">Check out the ten things to
avoid in your home finances. Most advice columns tell you how you should do things, but
there are all kinds of things you shouldn't do either. Here are ten frequent financial
mistakes that consumers routinely make -- and you should avoid at all
costs.</p></div></p>
    </div>

    <div class="lead_snippet">
    <h1>
    <a
```

Figure 243: Page source showing URLs with invalid suffixes

---

## Creating Editions

Now that you have created your Content Lists and Contexts you are ready to create your Editions. An Edition specifies the set of Content Lists to publish, which Site to publish them to, and the sequence in which to publish them. The sequence is important for the following reasons:

- A page is composed of binary items and text items. In general binary items represent portions of a page rather than an entire page. Items that represent portions of pages should be published before the pages that include them, so the pages can locate these items when they are assembled.
- Sequence is critical when you unpublish and publish to the same URL. It is possible for one Content List to both unpublish and publish, but if you do not control the order correctly, you could publish new content and then immediately unpublish it. To avoid this problem, build separate Content Lists that unpublish content and that publish content. When defining the Edition, sequence the unpublish Content Lists before publish Content Lists so old content is unpublished, and new content is not inadvertently unpublished.

There are four types of Editions that Rhythmyx defines:

- Automatic - Regularly scheduled editions that use predefined Content List queries.
- Manual - Editions that run when a user clicks the Publish button in the Editions page.
- Recovery - Editions that republish an earlier Edition of a Site or its contents.
- Mirror - Editions that copy content from one Site to another.

We are going to discuss Automatic Editions. Creating the other Edition types requires assistance from Rhythmyx Professional Services Organization.

When you first open the Editions page within the Publishing tab you see all of the currently registered Editions.





















					Editions	
					New Edition	Copy Edition
	Publish	Edition Name (id)	Description	Destination Site (id)	Edition Type	
✗		Full Enterprise Investments (301)	Publish all items in public state	Enterprise Investments ( 301)		
✗		Incremental Enterprise Investments (302)	Publish only new and modified items in public state	Enterprise Investments ( 301)		
✗		Full Non-Binary Enterprise Investments (303)	Publish all non-binary items in public state	Enterprise Investments ( 301)		
✗		EI Publish Now (310)	Publish selected content to the EI site immediately	Enterprise Investments ( 301)		
✗		Full Corporate Investments (307)	Publish all items in public state	Corporate Investments ( 303)		
✗		Incremental Corporate Investments (308)	Publish only new and modified items in public state	Corporate Investments ( 303)		
✗		Full Non-Binary Corporate Investments (309)	Publish all non-binary items in public state	Corporate Investments ( 303)		

Figure 244: Editions Page

The following table describes the columns on the Edition page.

Column	Description
Delete	Represented by the  button, which the user clicks to delete the Edition that resides in that row.
Publish	Represented by the  button. The user clicks this button to initiate the Publishing process.
Edition Name (id)	Name and ID assigned to the Edition.
Description	Description given to the Edition when it was first created.
Destination Site	The Site to which the Edition will be published.
Edition Type	The type of edition: Automatic  , Manual  , Mirror  , or Recovery  .

Your specifications require two Editions:

- An edition scheduled to run once a week that publishes all items ready to be published to the Enterprise Investments site, and unpublishes all items in an archive state.
- An edition schedule to run twice a day that publishes all new and modified items in a publish state to the Enterprise Investments site, and unpublishes all items in an archive state.

In the following topics, you use the Publishing Administrator to create these Editions, which are already defined by the following FastForward Editions:

- Full Enterprise Investments  
The Content Lists included in this Edition are:
  - rffEiUnpublish
  - rffEiFullBinary
  - rffEiFullNonbinary
- Incremental Enterprise Investments  
The Content Lists included in this Edition are:
  - rffEiUnpublish
  - rffEiIncremental

FastForward includes additional Editions that we will not recreate in this section.

---

Note: You most likely already have the FastForward Editions installed. Give your versions of the Editions different names than the FastForward ones, or use the information in your implementation plan as a substitute for the data used in the instructions in this chapter.

---

## Full Publish Edition

In the following exercise we are going to define a full publish Edition. In general, full publish Editions publish all content that is in a public state and remove all content that is in an archive state.

To create the Full Publish Edition:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Click the Publishing tab.
- 3 In the left navigation pane click the Editions by Site link.

---

Note: there are two additional options to choose from; you could list Editions by Name or Type. Selecting Site is a good practice when you have multiple sites in your system.

---

Content Explorer displays the Editions administration page.

- 4 Click the [**New Edition**] link.

Content Explorer displays the Edit Editions Properties page.

Figure 245: Edit Edition Properties page

- 5 Fill in **Edition Name**. This is a required field and the name you enter is displayed on the Publishing tab. Enter something similar to *Full Enterprise Investments*.
- 6 In the optional **Description** field enter *Publish all items in public state*.
- 7 Choose the **Destination Site** from the drop down list. The selections available are all Sites currently defined in your system. Select *Enterprise Investments* from the list.
- 8 Leave *Automatic* in the **Edition Type** drop down list since this is a scheduled edition. Options are *Automatic*, *Manual*, *Recovery*, and *Mirror*. For more information about the other types of editions, see the Publishing section in the online *CMS Help*.
- 9 Do not fill in **Recovery Publication(id)** and **Mirror Source Site** since this is not a Recovery or Mirror Edition.

At this point, your Edition registration appears similar to:

Figure 246: Edition registration prior to adding Content Lists

- 10 Click [OK] to save the Edition record.  
The Edition is saved, and the Edit Edition Page closes. The Edition is listed in the Editions Administration page.

Now you need to add Content Lists to the Edition.

- 11** Click the Full Enterprise Investments Edition in the Editions Administration page to open it in the Edit Edition Properties page.

Now the Edit Edition Properties page includes a section for including Content Lists.

**Edit Edition Properties**

Edition(id): Full Enterprise Investments(301)

\*Edition Name: Full Enterprise Investments

Description: Publish all items in public state

\*Destination Site: Enterprise Investments

Edition Type: Automatic

Recovery Publication(id): (Recovery only)

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

Save Cancel

**Edit Edition: Allowed Content** **Add Content List**

Content List Name (id)	Sequence	Authorization Type	Context	Preview
No entries found.				

Figure 247: Edit Edition page

- 12** Click the Add Content List link.

Content Explorer displays the Add Content List page.

**Edition Content List**

Edition(id): Full Enterprise Investments ( 301)

\*Content List: --Choose--

Sequence:

\*Authorization Type: --Choose--

\*Context: --Choose--

Save Cancel

Figure 248: Add Content List page

Add three of the Content Lists that you have created in the following order:

- your Content List for unpublishing
- your Content List for publishing binary content items
- your Content List for publishing non-binary content items

By publishing your Content Lists in this order, you will avoid accidentally publishing a Content Item and then unpublishing it. You will also have all graphics and files included on text pages available in your publishing location before the text pages are published.

- 13** From the Content List drop list choose *rffEIUnpublish*.
- 14** In the Sequence field enter *1* to publish this Content List first



**15** Authorization Types (or Auth Types) filter the related Content Items for which link URLs will be published when publishing a content item. Auth Types determine what Snippets will appear on a Page. Auth Types have two major purposes:

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

Rhythmyx comes with four predefined Auth Types:

- All Content (authtype=0): Output is assembled with all related Content Items, regardless of whether they are Public or not.
- All Public Content (authtype=1): Output is assembled with only those related Content Items that are Public.
- Custom (authtype=2): Output is assembled with those related Content Items that match a custom Auth Type that you have added.
- Site Folder Content (authtype=101): Output is assembled with all related Content Items already present on the publish Site.

From the **Authorization Type** field drop list select *All Public Content* so that your pages will not attempt to link to related content items that are not in a public State.

**16** From the **Context** field drop list select *Publish* since you want to use the Location Generation Scheme associated with the *Publish* Context.

**17** Click [**Save**].

The Content List is now added to the Edition.

**18** Click the Add Content List link again.

**19** From the **Content List** field drop down select *rffEIFullBinary*.

**20** In the **Sequence** field enter 2 to publish this Content List second.

**21** From the **Authorization Type** field drop down list select *All Public Content* so that your pages will not attempt to link to related content items that have not been published.

**22** From the **Context** field drop down list select *Publish* since you want to use the Location Generation Scheme associated with the *Publish* Context.

**23** Click [**Save**].

The second Content List is now added to the Edition.

**24** Click the Add Content List link again.

**25** From the **Content List** field drop down select *rffEIFullNonBinary*.

**26** In the **Sequence** field enter 3 to publish this Content List last.

**27** From the **Authorization Type** field drop down list select *All Public Content* so that your pages will not attempt to link to related content items that have not been published.

**28** From the **Context** field drop down list select *Publish* since you want to use the Location Generation Scheme associated with the *Publish* Context.

**29** Click [**Save**].

The third and final Content List is added to the Edition.






Edit Edition Properties					
Edition(id): Full Enterprise Investments(301)					
*Edition Name	Full Enterprise Investments				
Description	Publish all items in public state				
*Destination Site	Enterprise Investments				
Edition Type	Automatic				
Recovery Publication(id)	<input type="text"/> (Recovery only)				
Mirror Source Site	--Choose-- (Mirror only)				
<input type="button" value="Save"/> <input type="button" value="Cancel"/>					
Edit Edition: Allowed Content				Add Content List	
Content List Name (id)	Sequence	Authorization Type	Context	Preview	
 rffEiUnpublish (316)	1	All Public Content	Publish		
 rffEiFullBinary (310)	2	All Public Content	Publish		
 rffEiFullNonBinary (311)	3	All Public Content	Publish		

Figure 249: Full Enterprise Investments Edition with Content Lists

**30** Click [**Save**]. The Edition is now complete and can be used to run a Full Publish of the Enterprise Investments site.

## Incremental Publish Edition

The next type of Edition you will create is an Incremental Edition. An Incremental Edition is needed in order to publish new or updated information since the last publishing run. Many incremental Editions also remove all content that is in an archive state.

Many of the following steps have already been discussed in detail in the previous topic. See **Full Publish Edition** (see page 330) for a full discussion of the fields and options on the Edit Editions Properties page and the Add Content List page.

To create the Incremental Publish Edition:

- 1 Open a new Edit Editions Properties page.
- 2 In the **Edition Name** field enter a name similar to *Incremental Enterprise Investments*.
- 3 In the optional **Description** field enter the following text: *Publish only new and modified items in public state*.
- 4 Choose the **Destination Site** from the drop down list. The selections available are all Sites currently defined in your system. Select *Enterprise Investments* from the list.
- 5 Leave *Automatic* in the **Edition Type** drop down list since this is a scheduled edition.
- 6 Click [**OK**] to save the Edition record.  
The Edition is saved and you are returned to the Editions page.
- 7 Do not fill in **Recovery Publication(id)** and **Mirror Source Site** since this is not a Recovery or Mirror Edition.

At this point, your Edition registration appears similar to:

Figure 250: Edition registration prior to adding Content Lists

- 8 Click [OK] to save the Edition record.  
The Edition is saved, and the Edit Edition Page closes. The Edition is listed in the Editions Administration page.  
Now you need to add Content Lists to the Edition.
- 9 Click the *Incremental Enterprise Investments* Edition in the Editions Administration page to open it in the Edit Edition Properties page.  
Now the Edit Edition Properties page includes a section for including Content Lists.
- 10 Click the Add Content List link.  
Content Explorer displays the Add Content List page.  
Add two of the Content Lists that you have created in the following order:
  - your Content List for unpublishing
  - your Content List for incremental publishing
 By publishing your Content Lists in this order, you will avoid accidentally publishing a Content Item and then unpublishing it.
- 11 From the Content List drop list choose *rffEIUnpublish*.
- 12 In the Sequence field enter 1 to publish this Content List first
- 13 From the Authorization Type field drop list select *All Public Content* so that your pages will not attempt to link to related content items that are not in a public State.
- 14 From the Context field drop list select *Publish* since you want to use the Location Generation Scheme associated with the *Publish* Context.
- 15 Click [Save].  
The Content List is now added to the Edition.
- 16 Click the Add Content List link again.
- 17 From the Content List field drop down select *rffEIIncremental*.
- 18 In the Sequence field enter 2 to publish this Content List second.
- 19 From the Authorization Type field drop down list select *All Public Content* so that your pages will not attempt to link to related content items that have not been published.

**20** From the Context field drop down list select *Publish* since you want to use the Location Generation Scheme associated with the *Publish* Context.

**21** Click [**Save**].

The second Content List is now added to the Edition.

Edit Edition Properties

Edition(id): Incremental Enterprise Investments(302)

\*Edition Name: Incremental Enterprise Investments

Description: Publish only new and modified items in public s

\*Destination Site: Enterprise Investments

Edition Type: Automatic

Recovery Publication(id): (Recovery only)

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

Save Cancel

---

Edit Edition: Allowed Content Add Content List

Content List Name (id)	Sequence	Authorization Type	Context	Preview
✘ rffEiUnpublish (316)	1	All Public Content	Publish	
✘ rffEiIncremental (323)	2	All Public Content	Publish	

*Figure 251: Incremental Enterprise Investments Edition with Content Lists*

**22** Click [**Save**]. The Edition is now complete and can be used to run an Incremental Publish of the Enterprise Investments site.

## Testing your Content Lists

After you have added a Content List to an Edition, you can test it. You should now test your Content Lists prior to publishing.

To test your Content Lists:

- 1 In the Publishing tab, go to the Editions page and click on an Edition name to open the Edit Edition Properties page.

Edit Edition Properties


Edition(id): Full Enterprise Investments(301)

*Edition Name	Full Enterprise Investments
Description	Publish all items in public state
*Destination Site	Enterprise Investments ▾
Edition Type	Automatic ▾
Recovery Publication(id)	<input type="text"/> (Recovery only)
Mirror Source Site	--Choose-- ▾ (Mirror only)

Edit Edition: Allowed Content
Add Content List

Content List Name (id)	Sequence	Authorization Type	Context	Preview
✘ rffEiUnpublish (316)	1	All Public Content	Publish	
✘ rffEiFullBinary (310)	2	All Public Content	Publish	
✘ rffEiFullNonBinary (311)	3	All Public Content	Publish	

Figure 252: Full Enterprise Investments Edition with Content Lists

- 2 At the bottom of the page where the Content Lists are added, click the Preview button  for each Content List at the end of the row. It should display the xml list of content to be published:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <contentlist context="1" deliverytype="filesystem">
- <contentitem contentid="383" revision="1" unpublsh="no" variantid="506">
  <title>EI Global Financial Service Fund - allocation.jpg</title>
  <contenturl>http://doc2server:9975/Rhythmyx/Assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sys_contentid=383&sys_folderid=387&sys_cont
- <delivery>
  <location>/Images/Funds/EIGlobalServicesFund/item383.jpg</location>
</delivery>
<modifydate>2005-03-24 00:00:00</modifydate>
<modifyuser>admin1</modifyuser>
<contenttype>307</contenttype>
</contentitem>
- <contentitem contentid="384" revision="1" unpublsh="no" variantid="506">
  <title>EI Global Financial Service Fund - regional mix.jpg</title>
  <contenturl>http://doc2server:9975/Rhythmyx/Assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sys_contentid=384&sys_folderid=387&sys_cont
- <delivery>
  <location>/Images/Funds/EIGlobalServicesFund/item384.jpg</location>
</delivery>
<modifydate>2005-03-24 00:00:00</modifydate>
<modifyuser>admin1</modifyuser>
<contenttype>307</contenttype>
</contentitem>
- <contentitem contentid="385" revision="1" unpublsh="no" variantid="506">
  <title>EI Global Financial Service Fund - returns.jpg</title>
  <contenturl>http://doc2server:9975/Rhythmyx/Assembler/render?
    sys_revision=1&sys_siteid=301&sys_template=506&sys_itemfilter=public&sys_contentid=385&sys_folderid=387&sys_cont
- <delivery>
  <location>/Images/Funds/EIGlobalServicesFund/item385.jpg</location>
</delivery>
```

Figure 253: Preview of Content List

If the Content List is empty (no content matches its criteria), the preview appears as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<contentlist context="1" deliverytype="filesystem" />
```

Figure 254: Empty Content List

If your Content List preview does not resemble one of the above examples, or the Content List is empty but should not be, go back and resolve the error in your Content List before testing publishing of your content.

## Testing Publishing of your Editions

You have now registered or configured all of your publishing components and are ready to publish each of the editions that you have set up to verify that they publish correctly. The difference between these tests and regular publishing runs of these editions is that you would set up scheduled publishing for regular publishing runs, and in this case you will manually publish each of your editions.


**Publishing the Full Publish Edition** (below)

**Publishing the Incremental Publish Edition** (see page 334)

### Publishing the Full Publish Edition

Your FastForward Content Items should all already be in a public State, so you simply have to publish the Full Publish Edition to test it.

To publish the Full Publish Edition:

- 1 Log in to Rhythmyx Content Explorer.
- 2 Click the Publishing tab.  
In the left navigation pane click the Editions by Site link.  
Content Explorer displays the Editions page.
- 3 In the row for the *Full Enterprise Investments* Edition, click the button  next to the Edition Name in the Publish column.

Content Explorer shows you the following message that publishing has begun:

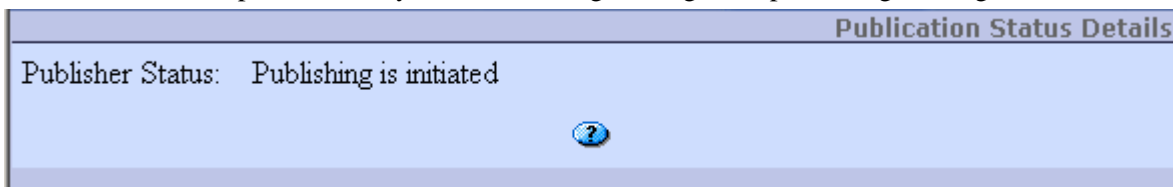


Figure 255: Publisher Status: Publishing is initiated

As publishing progresses, a message displays the Edition name and which Content List is currently being published:



Figure 256: Currently published Content List

When publishing completes, a message informs you that it is finished.

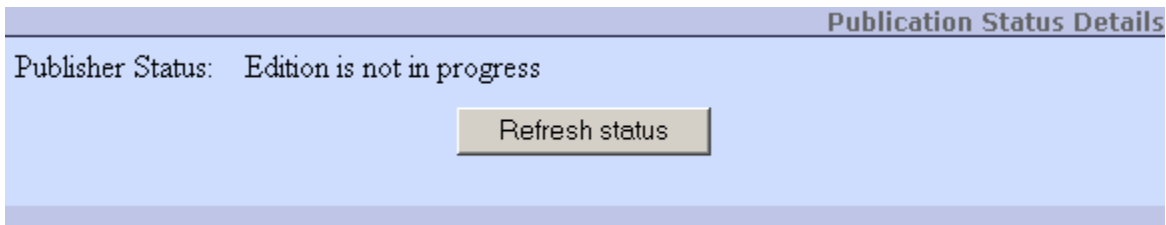


Figure 257: Publisher Status: Edition is not in progress

Now review the log for the Full Publish Edition, and check the publishing Site. (see page 340)

## Reviewing the Full Publish Log and Site

- 1 Click the Publishing tab in Content Explorer and click one of the Publication Log links. Locate the entries for your Publication. One entry is shown for each Content List published. Your Edition included three Content Lists, but since you did not have any content to unpublish, two entries appear for your edition, an entry for the binary Content List and an entry for the non-binary Content List.

Enterprise Investments (301)					Publications by Site				
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status					
2006-08-29 08:46:07.0 (301:302)	rffEiFullBinary (310)	Full Enterprise Investments		Success	73	0	0	0	0
2006-08-29 08:46:07.0 (301:303)	rffEiFullNonBinary (311)	Full Enterprise Investments		Success	68	0	0	0	0

Figure 258: Publishing log

The icons to the right of the log signify the following:

- Inserted Content Items
- Updated Content Items
- Removed Content Items
- Skipped Content Items
- Content Items that generated errors

You should only have numbers listed under since you only should have inserted Content Items.

The **Status** column either displays *Success* or *Error*. You can click on *Success* or *Error* to view a detailed log of the events that occurred during publishing.



- 2 Click the first entry (*rffEIFullBinary* Content List) under Date/Time (Publication ID:Pubstatus ID) to see a hierarchical representation of the Content Items published and the locations where they were published. Notice that when you put together the address of the fifth item under Filename `../EI_Home.war/AboutEnterpriseInvestments/item481.jpg` you have the same address we demonstrated for the Publish Context Generic Location Scheme.

				Publication Details
Filename	Operation	Status	CMS Link	
../				
EI_Home.war/				
AboutEnterpriseInvestments/				
active_item469.gif	publish	success	About Enterprise Investments Buttons	
active_item482.jpg	publish	success	About Enterprise Investments Section Image (NYSE Papers)	
inactive_item469.gif	publish	success	About Enterprise Investments Buttons	
inactive_item482.jpg	publish	success	About Enterprise Investments Section Image (NYSE Papers)	
item481.jpg	publish	success	About EI HomePage Image (NYSE Papers).jpg	
rollover_item469.gif	publish	success	About Enterprise Investments Buttons	
rollover_item482.jpg	publish	success	About Enterprise Investments Section Image (NYSE Papers)	
Files/				
item489.pdf	publish	success	EI Sample PDF.pdf	
item490.doc	publish	success	EI Sample Word Document.doc	
Images/				
CreditCard/				
item442.gif	publish	success	EIHomeEquityCreditCard.gif	
Funds/				
EIGlobalHealthSciencesFund/				

Figure 259: Publication Map

You can click on each page link to validate that the page was assembled correctly.

- 3 If you want to check your actual publishing root, look in <Rhythmyx root>\AppServer\server\rx\deploy\EI\_Home.war. (In **Registering the Publishing Site with Rhythmyx** (see page 65), we explained that the default root represented by .. is <Rhythmyx root>\AppServer\server\rx\deploy\). You see the same hierarchy of folders and files as in the log entry that you just looked at, except any non-binary files from the testEIFullNonBinary Content List are also present:

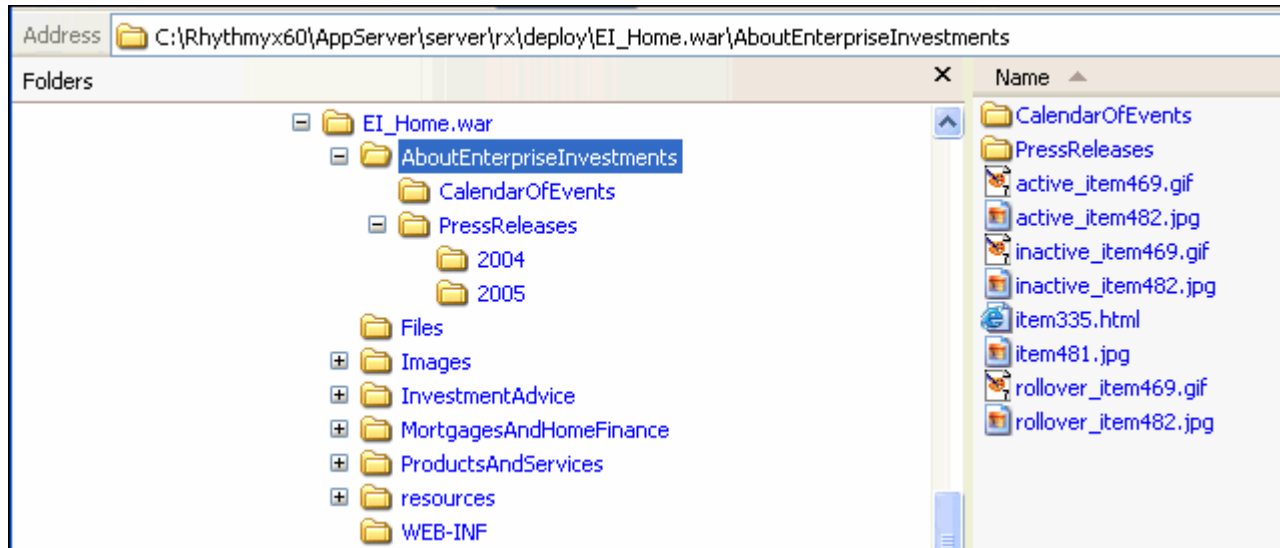


Figure 260: Documents published to publishing root

If your content has published to the wrong site, check if you have used the variable `$sys.site.path` instead of `$sys.pub_path` in your location scheme. `sys.pub_path` takes the value of the path assigned to the item's Content Explorer folder in its folder properties or if no path is assigned, the actual folder path holding the Content Item under the Site folder root in Content Explorer. `$sys.site.path` takes the value of `sys_siteid`, and will hold the incorrect site if a user action (such as previewing an item in a different site) changes `sys_siteid` prior to publishing.

- 4 To view the published Site from a Web browser, enter `http://localhost:9992/EI_Home/index.html`. This is the browser's address for Rhythmyx plus your assembly location scheme resolved for the home page. Entering this address in the browser causes it to display the home page of the Web Site:

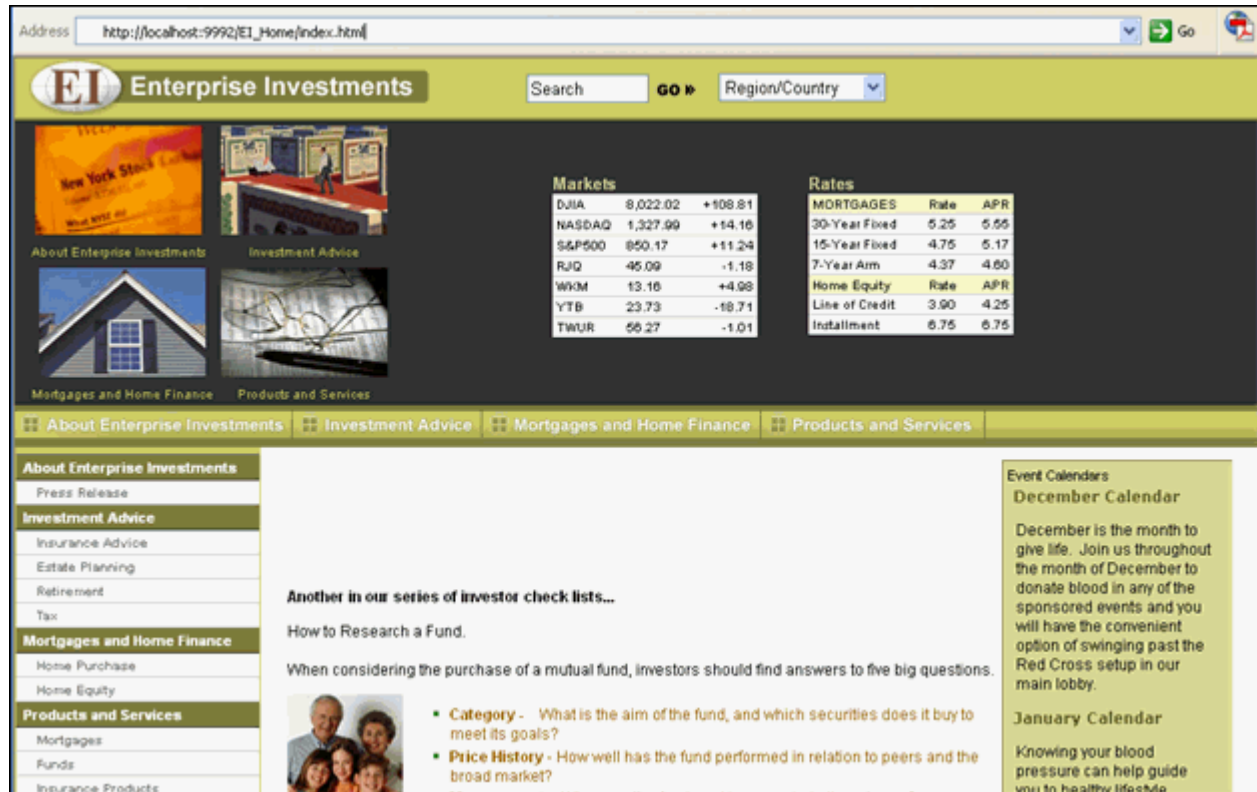


Figure 261: Published Enterprise Investments Generic Page

## Publishing the Incremental Publish Edition

In order to see results when you publish your Incremental Edition, you must change the content of some items in a public State or add some new items and move them to a public State.

In this example, we make the following changes to our content:

- In the folder Sites/EnterpriseInvestments/AboutEnterpriseInvestments/Press Releases/2005 create a new Press Release Content Item with the following values:
  - In System Title and Title enter *EI Receives Top Ratings*
  - In Callout enter *B & B Ratings gives Enterprise Investments straight A's in all categories.*
  - Change the year in the Start Date to 2005.


Save the item and move it into a Public State.

- make a text change to the About Enterprise Investments Generic Content Item;
- move the EI Reinsurance Generic Content Item into an Archive State.

For help creating content items, modifying published content items, and changing the Workflow State of content items, see the *Content Explorer online help*.

Many of the publishing steps have already been covered in the topic Publishing the Full Publish Edition. For more complete details for the following procedure, see *Publishing the Full Publish Edition* (see page 339).

To publish the Incremental Publish Edition:

- 1 Make the changes listed above to the specified Content Items (or similar changes to other Content Items).
- 2 In the Editions page, click the button  in the Incremental Enterprise Investments row. Content Explorer shows you messages telling you how publishing is progressing.
- 3 When you see the *Edition is not in progress* message, **review the log for the Incremental Publish Edition, and check the publishing Site** (see page 344).

## Reviewing the Incremental Publish Log and Site

- 1 Open the Publishing log. You should see two entries for the Incremental Enterprise Investments Edition, one for the rffEIUnpublish Content List and one for the rffEIIncremental Content List.










Enterprise Investments (301)					Publications by Site					
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status						
2006-08-29 10:42:26.0 (317:324)	rffEIUnpublish (316)	Incremental Enterprise Investments		Success	0	0	1	0	0	
2006-08-29 10:42:32.0 (317:325)	rffEIIncremental (323)	Incremental Enterprise Investments		Success	1	58	0	0	0	

Figure 262: Publishing Log Entries

rffEIUnpublish displays 1 in the  Removed Content Items column because you archived and removed one content item.

rffEIIncremental displays 1 in the  Inserted content items column and 58 in the  Updated content items column. You added one content item and modified another, so you may expect to see 1 in the Updated column. However, all Automatic Indexes are regenerated when an incremental Edition runs to ensure that any Automatic Indexes that include the new or modified content items add them to their lists.

- Click the entry for rffEIUnpublish under Date/Time (Publication ID:Pubstatus ID). The hierarchical listing appears as:

Filename	Operation	Status	Publication Details CMS Link
../ EI_Home.war/ ProductsAndServices/ InsuranceProducts/ item406.html	unpublish	success	EI Reinsurance

Figure 263: Log for archived item

The entry shows that the EI Reinsurance content item that you archived was successfully unpublished.

- Click the entry for rffEIIncremental under Date/Time (Publication ID:Pubstatus ID) to confirm that the content items that you modified and added were published and to see which content items were republished for inclusion in Automatic Indexes.
- Enter [http://localhost:9992/EI\\_Home/index.html](http://localhost:9992/EI_Home/index.html) in your Web browser again.
- Click the navigation link for Press Release.

Your EI Receives Top Ratings item is now listed under the 2005 Press Releases listing.

The screenshot shows the Enterprise Investments website. The header includes the EI logo, the text "Enterprise Investments", a search box, and a "Region/Country" dropdown menu. Below the header is a navigation bar with links for "About Enterprise Investments", "Investment Advice", "Mortgages and Home Finance", and "Products and Services". The main content area is titled "Enterprise Investments Home" and features a "Related..." section with "2005 Press Releases" and "2004 Press Releases". The 2005 press releases include:

- Sat, Jan 1, 2005 [Four Revenue Streams - Record Production](#)
- Sat, Jan 1, 2005 [Michael Flanders Appointed President of Enterprise Investments](#)
- Mon, Aug 22, 2005 [EI Receives Top Ratings](#)

The 2004 press releases include:

- Fri, Nov 5, 2004 [Enterprise Investments Acquires Merchant Credit Card From Provincial](#)
- Wed, Nov 10, 2004 [Highlights of 2004 Tax Changes](#)

On the right side, there is a highlighted box titled "Five Terrible Mortgage Mistakes" with the text: "Before you sign anything, make sure you are making the right decisions. Now looming in front of you are two big questions: How much can I afford? How do I choose the right home? Here's a start at answering those questions."

On the left side, there is a sidebar with a "Markets" section showing the following data:

Market	Value	Change
DJIA	8,022.02	+108.81
NASDAQ	1,327.99	+14.16
S&P500	850.17	+11.24

Figure 264: Press Release page

## Setting Up Publishing to a Local Web Server

Percussion Software recommends that in your implementation environment you publish to a robust Web server. This section describes how to configure Publishing to deliver content to a new Site, Express Investments, on the two most common production Web servers: Microsoft Internet Information Server and the Apache Web Server.

We will assume that the Web servers are installed using the defaults, and that a virtual root, named XI\_Home, has been set up for a new Site, Express Investments. We will also assume that you have copied the resources directory for your Site to this location, and that it contains all of the CSS, JavaScript, and static images required to support your Site.

Consult the documentation for your preferred Web server for details about installing and configuring the server, and for instructions to set up the Virtual Root.

Configuring delivery to the Express Investments Site in the Web server involves the following steps:

- 1 Create a new Site Registration for Express Investments.
- 2 Create a new Context Variable for Express Investments, with the required Values.
- 3 Create a new Edition to Publish the Express Investments Site.

## Creating a New Site Registration to Deliver to a Web Server

We will copy the existing Enterprise Investments Site Registration to create the Express Investments Site Registration. To create the new Site Registration:

- 1 In Content Explorer, click the Publishing tab.
- 2 In the left navigation, under Sites, click the By Name link.  
Content Explorer displays the Sites page.
- 3 Click the Copy Site link.  
Content Explorer displays the Copy Site page.
- 4 In the Source Site drop list, choose *Enterprise Investments*.
- 5 In the New Site field, enter *Express Investments*.

Copy Site	
Source Site	Select the Site you want to copy
	Enterprise Investments
New Site	
*Name	Express Investments
Create	Cancel

Figure 265: Copying the Enterprise Investments Site Registration

- 6 Click the [Create] button.

Content Explorer saves your new Registration and redirects you to the Sites page.

- 7 On the Sites page, click the *Express Investments* link.

Content Explorer displays the Express Investments Site Registration. Other than the name, the data will be the same as the Enterprise Investments Site Registration.

- 8 To configure the Site to Publish to Microsoft Internet Information Server:
- Change the value in the **Description** field from *Represents the Enterprise Investments web site* to *Represents the Express Investments web site*.
  - Change the value in the **Site Address (URL)** field from *http://127.0.0.1:9980/EI\_Home* to *http://127.0.0.1:80/XI\_Home*. (This value assumes that you have used the default port for IIS, 80. If you chose a different port for IIS, substitute that value for 9980.)
  - Change the value of the **Publishing Root Location** field from *../webapps* to *c:\inetpub\wwwroot\XI\_Home*.
  - Change the value of the **Folder Root** field from *//Sites/EnterpriseInvestments* to *//Sites/ExpressInvestments*.

Edit Site Properties	
<b>Site(id): (306)</b>	
*Site Name	Express Investments
Description	Represents the Enterprise Investments web site
Site Address (URL)	http://127.0.0.1:80/XI_Home
Home Page (URL)	
Publishing Root Location	c:\inetpub\wwwroot\XI_Home
*Publisher	localhost Publisher Default Port ▼
Status	Active ▼
Folder Root	//Sites/EnterpriseInvestments
Global Template	enterprise-global-template ▼
Nav Theme	▼
<b>FTP Information:</b>	
IP Address	127.0.0.1
Port Number	21
User ID	
Password	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 266: Site Registration for Internet Information Server

- 9 To configure the Site to publish to the Apache web server, the value of the Publishing Root Location field must point to a subdirectory of Apache's default publishing root, <Apacheroot>\htdocs. For example: c:\Program Files\Apache Group\Apache\htdocs\XI\_Home. Otherwise the Site Registration is the same as for IIS.

Edit Site Properties	
<b>Site(id): (306)</b>	
*Site Name	Express Investments
Description	Represents the Enterprise Investments web site
Site Address (URL)	http://127.0.0.1:80/XI_Home
Home Page (URL)	
Publishing Root Location	c:\Program Files\Apache Group\Apache\htdocs\XI_Home
*Publisher	Localhost Publisher Default Port
Status	Active
Folder Root	//Sites/EnterpriseInvestments
Global Template	enterprise-global-template
Nav Theme	
<b>FTP Information:</b>	
IP Address	127.0.0.1
Port Number	21
User ID	
Password	
Save Cancel	

Figure 267: Site Registration to Deliver to the Apache Web Server

- 10 Click the [Save] button to save the Site registration.

## Creating Context Variables for Delivery to a Web Server

Context Variables substitute local paths for internal Rhythmyx paths in published output. These substitute paths ensure that the HTML pages will retrieve CSS, Javascript files, and static images from the local resources directory rather than from the directory on the Rhythmyx server.

To create a Context Variable:

- 1 On the Publishing tab of Content Explorer, in the left navigation, under Variables, click the [By Name](#) link.  
Content Explorer displays the Variables page.
- 2 Click the [New Variable](#) link.  
Content Explorer displays the Edit Global Variables page.



- 3 In the **Name** field enter *Express Investments*.
- 4 In the **Value** field enter */XI\_Home/resources*. (This is the path to the resources directory for the Express Investments Site on IIS.)
- 5 In the **Context** drop list, choose *Publish*.
- 6 In the **Site** drop list, choose *Express Investments*.

Figure 268: Global Variable for Internet Information Server

- 7 Click the [**Save**] button to save the new Variable.

You may also want to create an additional Value for the Preview Context. The Preview Context points to a location on the Rhythmyx server (typically a subdirectory of the web\_resources directory; in our case, this subdirectory is named express\_investments.)

To add another Value to the Express Investments Global Variable:

- 1 Navigate to the Variables page on the Publishing tab of Content Explorer.
- 2 In the row with Express Investments, click the Add Value link.
- 3 Content Explorer displays the Edit Global Variable page.
- 4 In the **Value** field, enter *../web\_resources/express\_investments*.
- 5 In the **Context** drop list, leave the value as *Preview*.
- 6 In the **Site** drop list, choose *Express Investments*.
- 7 Click the [**Save**] button to save the new Value.

## Creating a New Edition to Publish to a Web Server

We will create the new Edition by copying the Full Enterprise Investments Edition. To create the new Edition:

- 1 On the Publishing tab of Content Explorer, in the left navigation, under Editions, click the By Name link.  
Content Explorer displays the Editions page.
- 2 Click the Copy Edition link.  
Content Explorer displays the Copy Edition page.
- 3 In the **Source Edition** drop list, choose *Full Enterprise Investments*.

- 4 In the New Edition field, enter *Full Express Investments*.




Figure 269: Copying the Full Enterprise Investments Edition

- 5 Click the [**Create**] button to create the new Edition.  
Content Explorer processes your request and redirects you to the Editions page.
- 6 Click the [Full Express Investments](#) link.  
Content Explorer displays the Edit Edition Properties page for the Full Express Investments Edition.
- 7 In the Destination Site drop list, choose *Express Investments*.
- 8 Click the [**Save**] button to save your changes.

---

## Setting Up FTP Publishing

Common practice is to run the Rhythmyx server on a separate machine from the production Web server. To publish to a remote machine, you must use FTP publishing.

We will assume that you have your virtual root defined on your Web server. As with your local Web server, you must copy the contents of the `<Rhythmyxroot>/web_resources` directory to the virtual folder directory.

You must also create a Virtual Directory in your FTP server that will point to the virtual root in your Web server. For the purposes of this exercise, we will assume that the virtual directory is called `ftpUser` and that it points to the `wwwroot` directory. We will also assume that the user account for access to the FTP site is `ftpUser`, and that this user has *password* as their password.

To set up FTP Publishing:

- 1 Create a Publisher registration for a FTP Publisher.
- 2 Create a new Site definition that uses the Publisher created in Step 1
- 3 Create a Content List Registration with a URL that includes *deliverytype=ftp*.
- 4 Define a new Edition that uses the Content List Registration created in Step 3

To use FTP, you must define a Publisher configuration with the necessary data.

To define a Publisher to use FTP:

- 1 In Content Explorer, choose the Publishing tab.
- 2 In the left navigation, under Publishers, click the By Name link.  
Rhythmyx displays the Publishers page.
- 3 Click the New Publisher link.  
Rhythmyx displays a blank Edit Publisher page.
- 4 In the Publisher Name field, enter *Localhost FTP Publisher*.
- 5 Optionally, in the Description field, enter a description, such as *Publish to remote Web server via FTP*.
- 6 In the IP Address field, enter *localhost* or the IP address of your local Rhythmyx server.
- 7 In the Port field, enter *9980* (or the port you assigned to the Rhythmyx Web application server).
- 8 In the Status drop list, select *Active*.
- 9 In the CMS User Name field, enter *rxpublisher*. In the CMS Password field, enter *demo*.
- 10 In the Publisher User Name field, enter *ftpUser* (or the name of the user you have configured to access your FTP server). In the Publisher Password field, enter *password* (or the password of the user you have configured to access your FTP server).

The completed configuration should resemble the following screenshot:

Publisher(id): Localhost FTP Publisher(303)		Edit Publisher
*Publisher Name	Localhost FTP Publisher	
Description	Publish to laptop FTP site	
*IP Address	10.10.10.147	
Port	9980	
Status	Active	
CMS User Name	rxpublisher	
CMS Password	*****	
Publisher User Name	ftpUser	
Publisher Password	*****	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>		
Configuration Parameters		Add User Param
Name	Value	Description
<input checked="" type="checkbox"/> debug	true	test debug
<input checked="" type="checkbox"/> filesystem	com.percussion.publisher.client.PSFilePublisherHandler	test filesystem
<input checked="" type="checkbox"/> ftp	com.percussion.publisher.client.PSFtpPublisherHandler	test ftp
<input checked="" type="checkbox"/> statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	test statusurl

Figure 270: FTP Publisher Configuration

- 11 Click the [Save] button to save the Publisher configuration.

Rhythmyx returns you to the Publishers page. The new Publisher will be included in the list of Publishers.

Publishers				
New Publisher				
	Publisher(id)	Description	IP Address	Status
<input checked="" type="checkbox"/>	Localhost FTP Publisher(303)	Publish to laptop FTP site	bobj	Active
<input checked="" type="checkbox"/>	Localhost Publisher Default Port(301)	Connects to a publisher running on default port on the same machine as the Rhythmyx server	127.0.0.1	Active

Figure 271: Publisher Page showing Localhost FTP Publisher listed

## Defining a Site Registration for FTP Publishing

You must define a Site configuration to use the FTP Publisher to publish the Content on the remote machine. We will copy and modify the Enterprise Investments Site Registration for our Site.

To create a Site for FTP Publishing:

- 1 In Content Explorer, choose the Publishing tab.
- 2 In the left navigation, under Sites, click the By Name link.  
Rhythmyx displays the Sites page.
- 3 Click on the Copy Site Link.  
Content Explorer displays the Copy Site Page
- 4 In the Source Site drop list, choose *Enterprise Investments*.
- 5 In the New Site field enter *Enterprise Investments FTP*.

Figure 272: Copying the Enterprise Investments Site Registration as Enterprise Investments FTP

- 6 Click the **[Create]** button to save the new Site Registration.
- 7 On the Sites Page, click the Enterprise Investments FTP link.  
Rhythmyx displays the Edit Site page for the Enterprise Investments FTP Site.
- 8 In the Description field, add the phrase *published to FTP* to the end of the text.
- 9 Delete the value in the Publishing Root Location field. This value is not useful when publishing through FTP.
- 10 In the Publisher drop list, select *Localhost FTP Publisher* (specifying that this Site will use the Publisher we just registered).
- 11 In the FTP Information: IP Address field, enter the IP address of your FTP server.
- 12 In the FTP Information: Port Number field, enter the port of your FTP server (the default is 21).
- 13 In the FTP Information: User ID field, enter *ftpUser* (or the name of the user with access rights to your FTP server).
- 14 In the FTP Information : Password field, enter *password* (or the password of the user with access rights to your FTP server).

The completed configuration should resemble the following screenshot:

Edit Site Properties	
<b>Site(id): (307)</b>	
*Site Name	Enterprise Investments FTP
Description	Represents the Enterprise Investments web site
Site Address (URL)	http://127.0.0.1:9980/EI_Home
Home Page (URL)	
Publishing Root Location	
*Publisher	Localhost FTP Publisher
Status	Active
Folder Root	//Sites/EnterpriseInvestments
Global Template	enterprise-global-template
Nav Theme	
<b>FTP Information:</b>	
IP Address	10.10.10.32
Port Number	21
User ID	ftpUser
Password	XXXXXXXXXX
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 273: Site Registration for FTP Publishing

- 15** Click the [Save] button to save the Site configuration.

The new site configuration will be added to the list of available Sites:






Sites			
	Site(id)	Description	Last Publication Date
	Corporate Investments (303)		2005-03-29 16:25:12.0
	Corporate Investments - Mirror(304)		
	Enterprise Investments (301)	Represents the Enterprise Investments web site	2005-03-30 08:25:52.0
	Enterprise Investments FTP(307)	Represents the Enterprise Investments web site	
	Enterprise Investments-Mirror(302)	Represents a mirror server for Enterprise Investments web site	

Figure 274: List of Sites showing new FTP Site

## Defining an FTP Content List Registration

The Site Root Full Content List included with FastForward is configured to Publish to a local file system. It includes a URL that defines the *deliverytype* as *filesystem*.

We want to use the same Content List on the Rhythmyx server to generate the content for Publishing, but we want to Publish to a remote location using FTP. We will therefore copy the URL from the Site Root Full Content List registration, and modify it in our new registration.

To create an FTP Content List registration:

- 1 In Content Explorer, choose the Publishing tab.
- 2 In the left navigation bar, under Content Lists, click the [By Name](#) link.  
Rhythmyx displays the Content List Administration page.
- 3 Click the Site Root Full link.  
Rhythmyx displays the edit Content List page for the Site Root Full Content List.
- 4 Copy the value of the *URL* field to the clipboard.
- 5 Under Content Lists, click the [By Name](#) link to return to the Content List Administrator..
- 6 Click the New Content List link.  
Rhythmyx displays a blank Edit Content List page.
- 7 In the **Name** field, enter *Site Root Full - FTP*. Enter the same value in the **Description** field.
- 8 In the **URL** field, paste the URL from the clipboard.
- 9 In the URL, find the *deliverytype* attribute. Change the value of the attribute from *filesystem* to *ftp*.  
Original URL:  
`/Rhythmyx/rx_Support_pub/folder_clist.xml?valid=y,i&delivery=filesystem&pubOp=pub`  
Modified URL:  
`/Rhythmyx/rx_Support_pub/folder_clist.xml?valid=y,i&delivery=ftp&pubOp=pub`
- 10 In the **Edition Type** drop list, select *Automatic*.

When you are finished, the new Content List registration should look like the following screenshot:

Figure 275: Content List Registration for FTP. Notice that the `deliverytype=ftp` attribute has been highlighted.

- 11 Click the [Save] button to save the new registration.

Rhythmyx returns you to the Content List Administrator page. The new Content List registration will be included in the list of registrations.

## Defining an Edition to Use the FTP Content List

Once you have defined your Content List registration, you must create an Edition that will use this Content List. We will copy and modify the Full Enterprise Investments Edition.


To create an FTP Edition:

- 1 In Content Explorer, choose the Publishing tab.
- 2 In the left navigation bar, under Editions, click the By Type link.  
Rhythmyx displays the Editions page.
- 3 Click the *Copy Edition* link.  
Content Explorer displays the Copy Edition page.
- 4 In the **Source Edition** drop list, choose *Full Enterprise Investments*.
- 5 In the **New Edition** field, enter *Full Enterprise Investments FTP*.

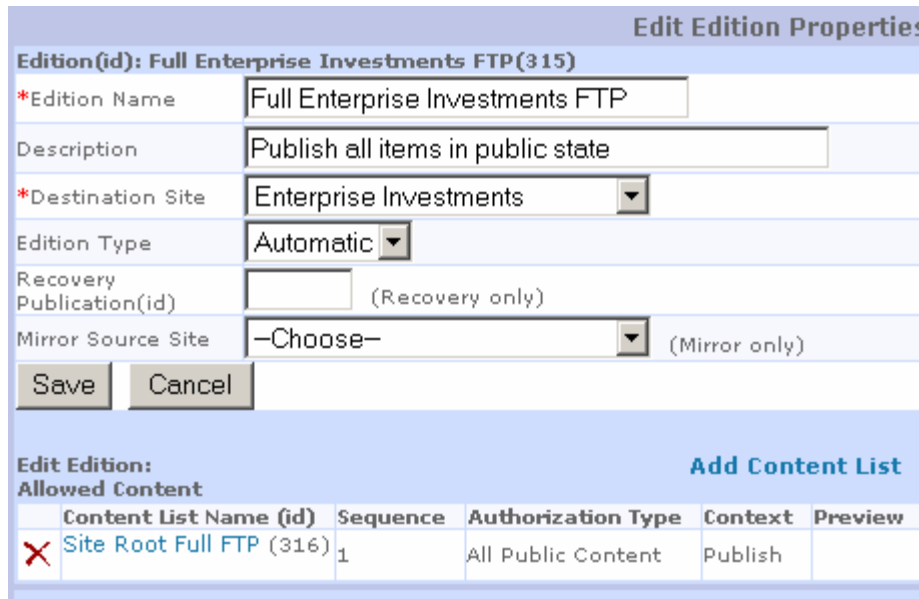
Figure 276: Copying the Full Enterprise Investments Edition for FTP Publishing

- 6 Click the [Create] button to create the new Edition.
- 7 On the Editions page, click the Full Enterprise Investments FTP link.  
Content Explorer displays the Edit Edition Properties page.



- 8 In the Description field, add *using FTP* to the end of the text.
- 9 In the Destination Site drop list, choose *Enterprise Investments - FTP*.
- 10 Click the delete button  in the row of the Site Root Full Content List to remove this Content List from the Edition.
- 11 Click the [Add Edition Content List](#) link.  
Rhythmyx displays the Edit Edition Content List page.
- 12 In the Content List drop list, choose *Site Root Full - FTP*.
- 13 In the Sequence field, enter *1*.
- 14 In the Authorization Type drop list, choose *All Public Content*.
- 15 In the Context drop list, choose *Publish*.
- 16 Click the [Save] button to save the Edition Content List assignment.

Rhythmyx returns you to the Edit Edition Content List page for the Full Internet - FTP Edition. The completed Edition should resemble the following screenshot:



**Edit Edition Properties**

**Edition(id): Full Enterprise Investments FTP(315)**

\*Edition Name: Full Enterprise Investments FTP

Description: Publish all items in public state

\*Destination Site: Enterprise Investments

Edition Type: Automatic

Recovery Publication(id): (Recovery only)

Mirror Source Site: -Choose- (Mirror only)

Save Cancel

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


Content List Name (id)	Sequence	Authorization Type	Context	Preview
 Site Root Full FTP (316)	1	All Public Content	Publish	

Figure 277: FTP Edition


## Running the FTP Edition

Once you have completed setting up to Publish the FTP Edition, you can publish it.

To publish the FTP Edition:

- 1 On the Publish tab of Content Explorer, go to the left navigation bar and under Editions, click the [By Name](#) link.

Rhythmyx displays the Editions page.

- 2** In the row of the Full Internet - FTP Edition, click the Run Edition button .  
Rhythmyx will publish the Edition as usual.
- 3** When the Edition is complete, go to your Virtual Root directory. You should see a directory structure similar to that of the Editions you have published previously.

# Database Publishing in Rhythmyx

Every Rhythmyx installation comes with a database publishing plugin that enables you to publish both raw and formatted content into a wide range of database schemas. To implement database publishing, you create a database publishing Template in which you map Content Item data to target table columns. After retrieving content from the Rhythmyx repository, Rhythmyx can insert the data directly into the columns you have mapped in the target database or assemble it into formatted outputs before inserting it into target database columns.

Database publishing enables Rhythmyx to simultaneously support content delivery through various Web servers, such as Net/IIS, BEA Weblogic Portal, or iPlanet/Apache Web Server, and custom built database driven delivery applications. Where you deliver the content published to the database depends on your company's Web presence and needs.

During the modelling and design process that was covered earlier in this document, we determined that you wanted to perform database publishing to enable users to enter a query into your Web Site and get back a page that matched the parameters of the query. In this case, the content that you publish to the database will appear on the same Web Site as the content that you publish to file system, and you will use database publishing to enhance the capabilities of your Web Site.

Database publishing could also allow you to display content on a portal, and take advantage of its requirements for both unformatted and formatted data. A portal could store a mixture of raw content (for formatting by the portal), metadata (for maintaining structure and indexing in the portal), and formatted content snippets that the portal developer could simply place within a page, or a finished document (such as a PDF) that could then be downloaded from the site. The portal's delivery application would act upon and process the content as it normally would using its own local datastore.

Install the database publishing plugin by selecting a Custom Publisher setup type and the Database Publisher feature when you install the Rhythmyx suite. (See the *Installing Rhythmyx* document for detailed installation instructions.) Rhythmyx installs the database publishing plugin into `<Rhythmyxroot>/AppServer/server/rx/deploy/RxServices.war/WEB-INF/lib/rxdbpublisher.jar`. The database publishing plugin publishes all content lists that have a `deliverytype` parameter of `database`.

---

## Special Components of Database Publishing

Before you implement database publishing, you should implement file system publishing. Database publishing runs as part of Rhythmyx's Publishing system, using the same building blocks as file system publishing, including Templates, Editions, and Content Lists. However, you must configure your components for database publishing with different settings than those used for file system publishing.

To perform database publishing, you must create an XML file that becomes the source in a database publishing Template. The database publishing plugin can only publish database publishing Templates, which are XML only, include RDBMS schema information, and conform to `sys_Database Publisher.dtd`.

## Database Publishing Specifications

In our database publishing example we will store data from our Event Content Items in a database that users can query to find matching events. Therefore, we must publish data from our Event Content Type into tables in a target database. In this example, our Rhythmyx repository and target database are MS SQL server.

Some customers who perform database publishing publish to a parent table and several child tables that duplicate child field sets in Content Items; others publish to a single table that does not include child tables. In order to demonstrate how to handle the more complex case, in this chapter, we will publish data to one parent table and one associated child table. We will publish data from the main fields in our Event Content Type to the parent table and the fields in its event\_location child field set to the child table. We use the *Event Content Type (Event) that we created earlier that includes a child field set.* (see page 242) The main fields in the Event Content Type appear in the following table and the fields in its child field set appear in the table below it.

Fields and Field Sets		
Name	Label	Control
sys_title	System Title:	sys_EditBox
displaytitle	Title:	sys_EditBox
sys_contentstartdate	Start Date:	sys_CalendarSimple
sys_contentexpirydate	Expiration Date:	sys_CalendarSimple
sys_reminderdate	Reminder Date:	sys_CalendarSimple
keywords	Keywords:	sys_TextArea
description	Description:	sys_TextArea
callout	Callout:	sys_EditLive
body	Body:	sys_EditLive
event_start	Event Start Date:	sys_CalendarSimple
event_end	Event End Date:	sys_CalendarSimple
event_location	Event Location:	sys_Table
event_type	EventType:	sys_DropDownSingle
filename	File Name:	sys_EditBox
sys_suffix	Suffix:	sys_EditBox
sys_communityid	Community:	sys_DropDownSingle
sys_workflowid	Workflow:	sys_DropDownSingle
sys_lang	Locale:	sys_DropDownSingle
sys_currentview		sys_HiddenInput
webdavowner	WebDAV Owner:	sys_TextArea
sys_hibernateVersion		sys_HiddenInput

Figure 278: Event Content Type fields

Fields in event_location		
Name	Label	Control
event_city	Event City:	sys_EditBox
event_state	Event State:	sys_EditBox
event_address	Event Address:	sys_TextArea
event_contact	Event Contact:	sys_TextArea

Figure 279: Event Location field set

Our target tables have the following characteristics:


- The parent table holds some of the main fields from the Event Content Type: displaytitle, callout, body, event\_start, event\_end, and event\_type.
- The child table holds the same information as the child field set in our Content Type: event\_city, event\_state, event\_address, and event\_contact.
- Both tables also include the Content ID and use it as a primary key/foreign key that associates the tables to one another.
- The child table also uses the seq column as a primary key. The seq column enables the database publishing plugin to move incrementally through all the rows in the child table.

---



Note that in this example, the unassembled content is published from Rhythmyx Content Items to the target database. If you want to publish assembled content to your target database, contact Rhythmyx Professional Services Organization (PSO) for assistance.

---

The target tables should resemble:

	Column Name	Data Type	Length	Allow Nulls
	CONTENTID	int	4	
	DISPLAYTITLE	nchar	512	✓
	CALLOUT	ntext	16	✓
	BODY	ntext	16	✓
	EVENT_START	datetime	8	✓
	EVENT_END	datetime	8	✓
	EVENT_TYPE	nchar	512	✓

*Figure 280: TARGET\_CONTENT table*

	Column Name	Data Type	Length	Allow Nulls
	CONTENTID	int	4	
	SEQ	int	4	
	EVENT_CITY	nchar	50	✓
	EVENT_STATE	nchar	50	✓
	EVENT_ADDRESS	nchar	255	✓
	EVENT_CONTACT	nchar	255	✓

*Figure 281: TARGET\_LOCATION table*

---

NOTE: In an actual implementation of database publishing, your target tables may already exist. If not, you would create them at this point. In our sample implementation we have created these tables in the same MS SQL Server that we use for our Rhythmyx repository, but in a different database named targetdb. If you want to follow our sample implementation, create the above tables in a database other than your Rhythmyx repository at this time.

If you did not create your target tables, the database publishing handler would create them for you, but in that case, you would have to write the source xml for storing target table data by hand (later in this chapter, we will show you how to build the source xml with a Rhythmyx utility).

---

## Checking your Database Publishing Plugin Parameter

All installations of Rhythmyx include the database publisher plugin. If you have checked Database Publishing during installation, the plugin should be listed as a parameter in your default publisher. In the graphic below, it is the first parameter listed under Configuration Parameters.

**Publisher(id): Localhost Publisher Default Port(301)** Edit Publisher

\*Publisher Name: Localhost Publisher Default Port

Description: Connects to a publisher running on default port

\*IP Address: 127.0.0.1

Port: 9992

Status: Active

CMS User Name: rxpublisher

CMS Password: ●●●●

Publisher User Name:

Publisher Password:

Configuration Parameters			Add User Param
Name	Value	Description	
<input checked="" type="checkbox"/> database	com.percussion.publisher.dient.PSDatabasePublisherHandler	database publisher handler	
<input checked="" type="checkbox"/> portal	com.percussion.portal.PSPortalPublisherPlugin	The plugin to publish to a portal site	
<input checked="" type="checkbox"/> filesystem	com.percussion.publisher.dient.PSFilePublisherHandler	component used for filesystem publishing	
<input checked="" type="checkbox"/> ftp	com.percussion.publisher.dient.PSFtpPublisherHandler	component used for FTP publishing	
<input checked="" type="checkbox"/> statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	Receives XML log from publisher and updates database	

Figure 282: Database Plugin parameter

If the database parameter is not present, click [Add User Param](#), and specify the exact Name and Value shown in the graphic.

---

## Steps for Database Publishing

To implement database publishing:

- 1 ***Determine which Rhythmyx data you will publish to your target database*** (see page 361), as well as which parent and child tables in your target database you will publish to. Make sure every table that you are publishing to has a primary key.  
  
If the tables for storing Rhythmyx data in your target database do not already exist, create them.
- 2 ***Confirm that the database plugin parameter is defined for your default Publisher.*** (see page 363)
- 3 ***Create a datasource for your target database.*** (see page 365)
- 4 ***Use Rhythmyx's runTd.bat utility*** (see page 368) to test your connection to your target database and create the XML source file for your database publishing Template.
- 5 ***Optionally, create your database publishing Site.*** (see page 373)
- 6 ***Create your database publishing Template.*** (see page 375)
- 7 ***Optionally, create your database publishing Context.*** (see page 383)
- 8 ***Create your database publishing Content List.*** (see page 385)
- 9 ***Create your database publishing Edition.*** (see page 387)
- 10 ***Publish to your database.*** (see page 389)
- 11 ***Review your results.*** (see page 390)
- 12 ***Debug any errors.*** (see page 394)



## Creating a Datasource for Your Target Database

In Rhythmyx, you must create a datasource for your target database so Rhythmyx has the correct information for connecting to it. You cannot create the database publishing datasource using the Rhythmyx Server Administrator; use one of the sample datasource files that JBOSS provides. Edit it to include the appropriate information for your database and save it with a new name.

To create a datasource for your target database:

- 1 For datasource file configurations for drivers that JBOSS supports, see the examples provided in `<Rhythmyx root>\AppServer\docs\examples\jca`. Look for a file with a prefix that matches your target database driver and ends in `-ds.xml`. Refer to the sample files that do not include `-xa` in their names.
- 2 Copy the `<database>-ds.xml` to another file that ends in `-ds.xml` into `<Rhythmyx root>/AppServer/server/rx/deploy/`. Rhythmyx looks for datasource configurations in all files in this directory that end in `-ds.xml`.

In this exercise, since we are using a jtds driver for Microsoft SQL Server we copy the closest example, `<Rhythmyx root>\AppServer\docs\examples\jca\mssql-ds.xml` to `<Rhythmyx root>/AppServer/server/rx/deploy/rxdbpub-ds.xml`

The original `mssql-ds.xml` file appears as:

```

17 <datasources>
18   <local-tx-datasource>
19     <jndi-name>MSSQLDS</jndi-name>
20
21     <connection-url>jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
22
23     <driver-class>com.microsoft.jdbc.sqlserver.SQLServerDriver</driver-class>
24     <user-name>x</user-name>
25     <password>y</password>
26     <!-- sql to call when connection is created -->
27     <new-connection-sql>some arbitrary sql</new-connection-sql>
28     <!-- sql to call on an existing pooled connection when it is obtained from pool -->
29     <check-valid-connection-sql>some arbitrary sql</check-valid-connection-sql>
30
31
32     <!-- corresponding type-mapping in the standardjbosscomp-jdbc.xml (optional) -->
33     <metadata>
34       <type-mapping>MS SQLSERVER2000</type-mapping>
35     </metadata>
36   </local-tx-datasource>
37
38 </datasources>

```

Figure 283: Rhythmyx's default datasource file

- 3 For this example, make the following edits to rxdbpub-ds.xml:
  - a) Change <jndi-name> to a new name for your new datasource. The name should begin with *jdbc/*. For this exercise, change <jndi-name> to *jdbc/TargetDB*.
  - b) The <connection-url> is not correct for jtds. We change the first part to: *jdbc:jtds:sqlserver:* and change MyDatabase to the name of our database. In this example we change it to *targetdb*.
  - c) The driver class is also incorrect for jtds. We change it to: *net.sourceforge.jtds.jdbc.Driver*
  - d) Change the <user-name> and <password> elements to have the values required for accessing your database.

The completed datasource file should appear as:

```

17 <datasources>␣
18   <local-tx-datasource>␣
19     <jndi-name>jdbc/TargetDB</jndi-name>␣
20
21     <connection-url>jdbc:jtds:sqlserver://localhost:1433;DatabaseName=ta
22     rgetdb</connection-url>␣
23     <driver-class>net.sourceforge.jtds.jdbc.Driver</driver-class>␣
24     <user-name>sa</user-name>␣
25     <password>mypass</password>␣
26     <!-- sql to call when connection is created␣
27     <new-connection-sql>some arbitrary sql</new-connection-sql>␣
28     -->␣
29     <!-- sql to call on an existing pooled connection when it is
30     obtained from pool ␣
31     <check-valid-connection-sql>some arbitrary
32     sql</check-valid-connection-sql>␣
33     -->␣
34     <!-- corresponding type-mapping in the standardjbosscomp-jdbc.xml
35     (optional) -->␣
36     <metadata>␣
37       <type-mapping>MS SQLSERVER2000</type-mapping>␣
38     </metadata>␣
39   </local-tx-datasource>␣
40 </datasources>␣

```

Figure 284: Datasource for target database

- 4 Save and close the datasource file.
- 5 Restart the Rhythmyx Server.

- 6 Test that Rhythmyx can connect to your DataSource before continuing. Open a browser and enter: `http://localhost:9992/RxServices/jnditest.jsp` in the command line. Your browser should return a page similar to the following. The portion of the datasource name that follows `jdbc/` should be listed on the page. In the following example, the listing of *TargetDB OK* indicates that the TargetDB datasource is configured correctly.

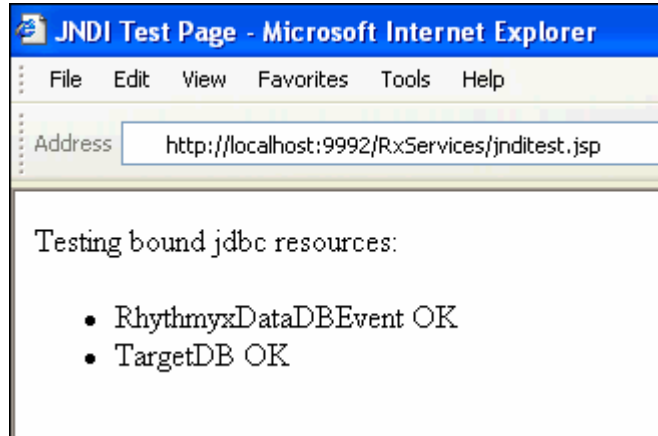


Figure 285: JNDI Test Page

---

## Creating the Source Code for Your Database Template

To check Rhythmyx's connection to your target database and to create the XML source for your database Template, use the `runTd.bat` utility in `<Rhythmyx root>/AppServer/bin`. The XML source includes a definition of your target database's parent and child tables.

- 1 Run `<Rhythmyx root>/AppServer/bin/RunTd.bat` (for Windows)  
or  
`<Rhythmyx root>/AppServer/bin/RunTd.sh` (for Unix)  
The Table Definition Builder opens to the Connection tab.
- 2 In **Database Server**, enter the name of your target database server, either the physical name of the machine or the IP address. The name is not case sensitive. In this example, the Database Server is the same Server that we use for our Rhythmyx repository.
- 3 In the **Database Type** drop list, choose the name of your database type. In this example, we choose *MSSQL*, but you would choose the database type appropriate for your system.
- 4 In the **Driver** drop list, choose the name of your database driver. In this example, we choose *jtds:sqlserver*, but you would choose the database driver appropriate for your system.
- 5 In the **Driver Class** drop list, choose the name of your driver class. In this example, we choose *net.sourceforge.jtds.jdbc.Driver*, but you would choose the driver class appropriate for your system.
- 6 In **Database Name**, enter the name you have given your database. In this example, enter *targetdb*.
- 7 In **Database Schema**, enter *dbo*.
- 8 In **User ID** enter the user ID for logging in to the target database.
- 9 In **Password**, enter the password for logging in to the target database.

Your completed Connection tab should resemble:

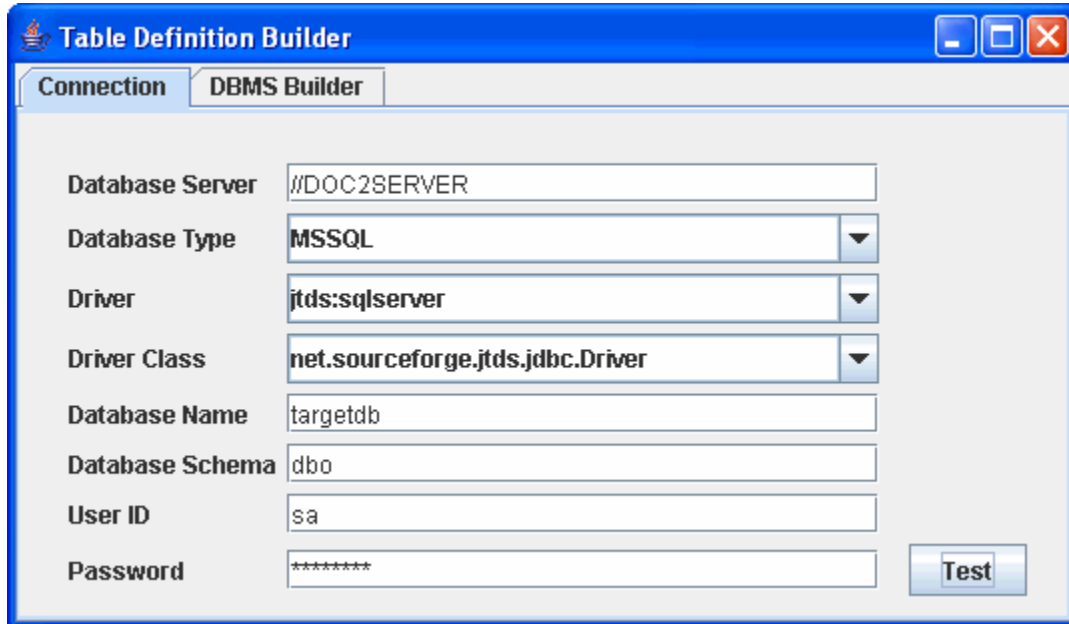


Figure 286: Table Definition Builder, Connection tab

The following tables provide examples of the information required to connect to different database types. Note that the examples under Database Server assume that you are running the Rhythmyx Server and the database server on the same machine.

DB Type	Driver	Database Server	Driver Class	Database Name	Database Schema	User ID
MSSQL	jtds:sqlserver	//localhost:1433	net.sourceforge.jtds.jdbc.Driver	targetdb	dbo	sa
MSSQL	sqlserver	//localhost:1433	com.microsoft.jdbc.sqlserver.SQLServerDriver	targetdb	dbo	sa
ORACLE	oracle:thin	@localhost:1521:UTF8	oracle.jdbc.driver.OracleDriver	N/A	target_schema	target_schema
DB2	db2	//server:port/database  (for example, //localhost:50000/RX601)	com.ibm.db2.jcc.DB2Driver	N/A	N/A	sa

**10** To test Rhythmyx's connection to the server, click **[Test]**.

A confirmation dialog similar to the following should appear:

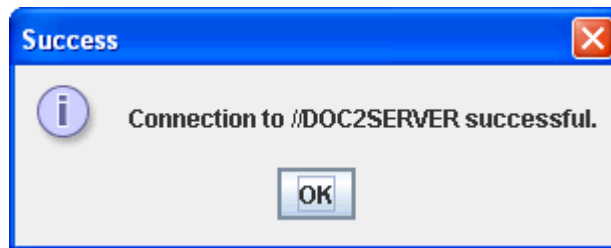


Figure 287: Success dialog

If the connection is not successful or the name of the server is not correct, review your entries in the Connection tab.

**11** To create your Template's source xml, click the DBMS Builder tab.

**12** Click [**Catalog**].

The tables that you have created for Rhythmyx database publishing are listed as well as any others in the database.

**13** Check the TARGET\_CONTENT and TARGET\_LOCATION tables (or the tables you have created for Rhythmyx database publishing).

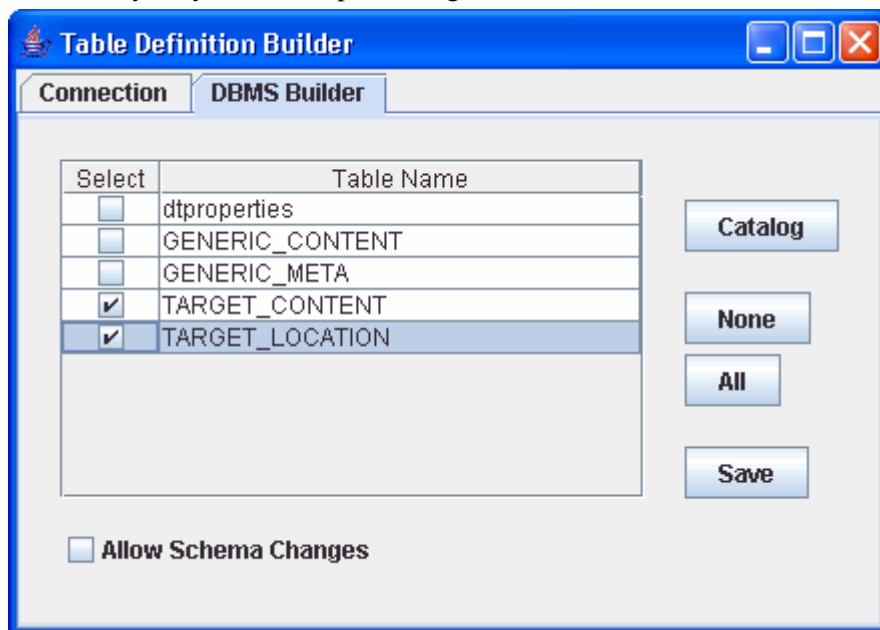


Figure 288: DBMS Builder tab

**14** Click [**Save**].

A Save dialog opens and prompts you to choose a name for the xml file and a folder to save it in. You could save the source file anywhere, but we will save ours in the Rhythmyx root. Name it DBTemplateSource.xml.

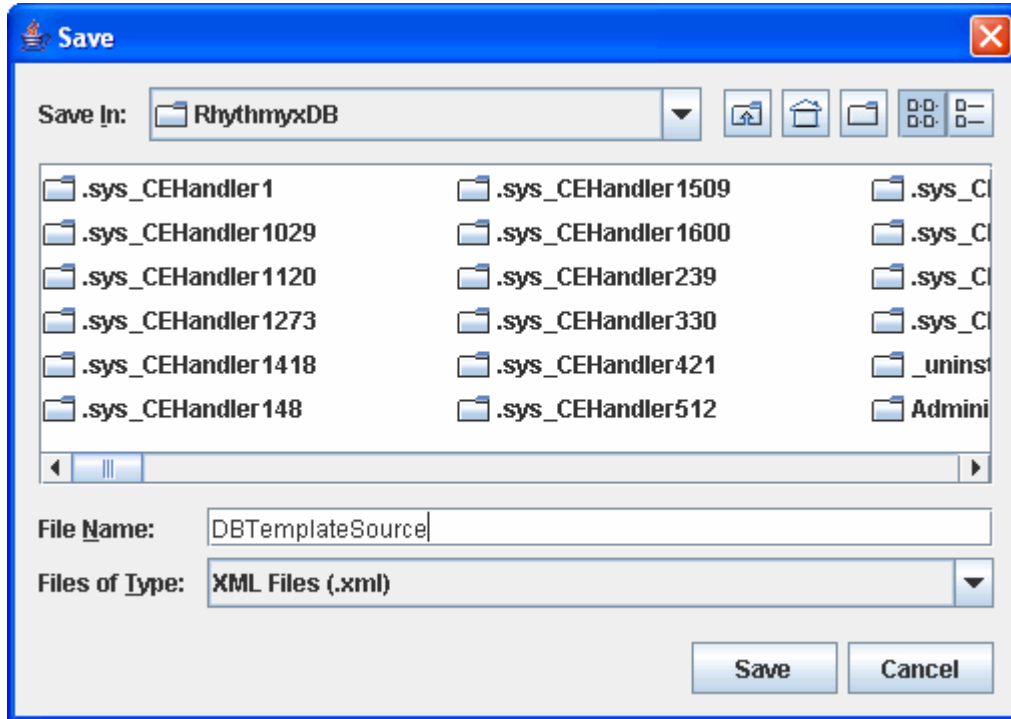


Figure 289: Save dialog

- 15 Click [Save].

A dialog tells you that the xml file is successfully created.

- 16 Click  to close the Table Definition Builder.

If you are writing to sequential columns in an Oracle database, see *Modifying the Table Definition File for Sequential Columns in Oracle* (on page 371).

## Modifying the Table Definition File for Sequential Columns in Oracle

If the table(s) defined in your table definition file has sequential columns, you must add a `<sequence>` tag inside the `<columndef>` tag to avoid a key violation error.

In Oracle, the value in the `<sequence>` tag must be the name of the sequence used to obtain the value of the column. If you add the `<sequence>` tag in other databases (MS SQL Server, UDB DB2) the value in the `<sequence>` tag is ignored.

If the row action is replace ("r"), the database server performs an update if the row exists, and an insert if it does not exist. An update requires a valid key value. The server cannot obtain the value of sequential columns; if your primary key contains sequential columns, you must add an update key (if an update key does not already exist). The update key should not contain any sequential columns.

NOTE: The `sys_Tabledef.dtd` specifies that keys and index definitions must be defined in your table definition file in the order: Primary Key, Foreign Key, Update Key, Index Definitions.

Example:

If a table has the primary key column `ID`, the Table Definition Builder creates the following column definition for `ID` in the table definition file:

```
<columndef action="c" name="ID">
  <jdbctype>INTEGER</jdbctype>
  <allowsnull>no</allowsnull>
</columndef>
```

If `ID` is a sequential column, you must manually edit its column definition in the table definition file and add a `<sequence>` tag:

```
<columndef action="c" name="ID">
  <jdbctype>INTEGER</jdbctype>
  <allowsnull>no</allowsnull>
  <sequence>ID</sequence>
</columndef>
```

If the table already exists and you want to change the action to replace `"r"`, you must have an update key. If the table does not already have an update key and you cannot use an existing field as an update key, you must add one to the table:

```
<primarykey action="r">
  <name>ID</name>
</primarykey>
<updatekey action="r">
  <name>CONTENTID</name>
</updatekey>
```



---

## Registering the Database Publishing Site

Optionally, you can register a Site for database publishing for the database publishing plugin to use. However, Rhythmyx only uses some of the fields, and they serve different purposes than they do in file system publishing.

To register the database publishing Site:

- 1 Open Content Explorer and go to the Publishing tab.
- 2 In the left navigation pane, click Sites/By Name.  
Content Explorer opens the Sites page.
- 3 Click New Site.  
Content Explorer opens the Edit Sites Properties page.
- 4 In Site Name, enter *Database Site*.
- 5 In Description, enter *Database Publishing Site*.
- 6 The database publishing plugin does not use Site Address, so leave this field blank.
- 7 Rhythmyx uses the value that you enter in Publishing Root Location as the top folder address in the Publication Details Map. Enter *doc2server:1433/targetdb/*  
where:
  - doc2server is the name of the server
  - 1433 is the server port
  - targetdb is the name of the database to which you want to publish
- 8 In the Publisher drop list choose *Localhost Publisher Default Port*.
- 9 In Folder Root enter *//Sites/EnterpriseInvestments* since the publisher will be pulling content from the *//Sites/EnterpriseInvestments* folder.

It is not necessary to enter information in any of the other Site registration fields. The completed page should appear similar to:

Site(id): (305) <span style="float: right;">Edit Site Properties</span>	
*Site Name	Database Site
Description	Database Publishing Site
Site Address (URL)	
Home Page (URL)	
Publishing Root Location	doc2server:1433/targetdb/
*Publisher	Localhost Publisher Default Port ▾
Status	Active ▾
Folder Root	//Sites/EnterpriseInvestments
Global Template	▾
Nav Theme	▾
Allowed Namespaces	
<b>FTP Information:</b>	
IP Address	
Port Number	
User ID	
Password	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Figure 290: Site Registration

**10** Click [Save].

---

# Creating the Database Publishing Template

Create your database publishing Template in the Rhythmyx Workbench.

To create the database publishing Template:

- 1 Open the Rhythmyx Workbench.
- 2 Choose *File > New > Template*.  
The New Template Wizard opens.
- 3 Under *Type*, choose *Database publishing XML*.

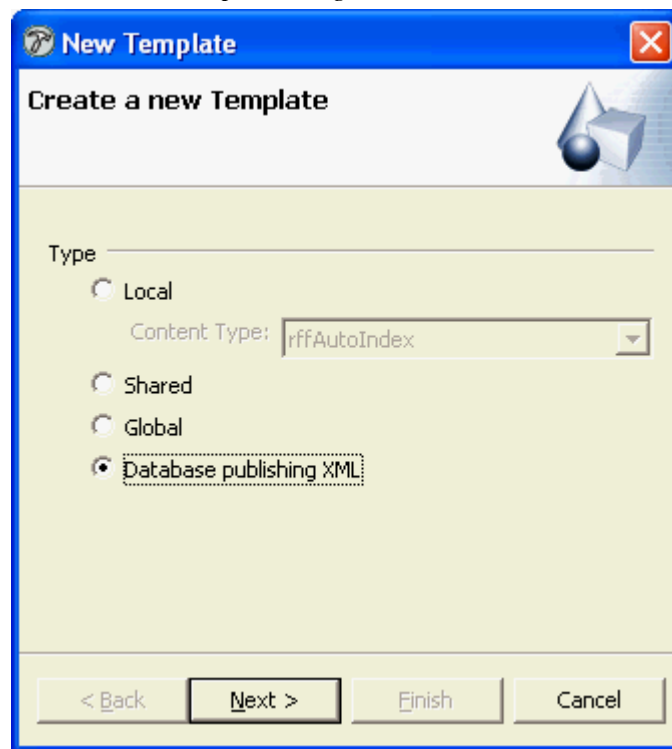


Figure 291: New Template wizard

- 4 Click [**N**ext].  
The next page of the wizard opens.
- 5 In *Template name* enter *EventDB*.  
The same name is entered in *Label*. Leave the name in *Label*.
- 6 *Description* is optional. Enter *Template for storing Event Content Type information in a database*.

- 7 In Source, navigate to the Rhythmyx root and choose the xml file that you saved using the Table Definition Builder, *DBTemplateSource.xml*.

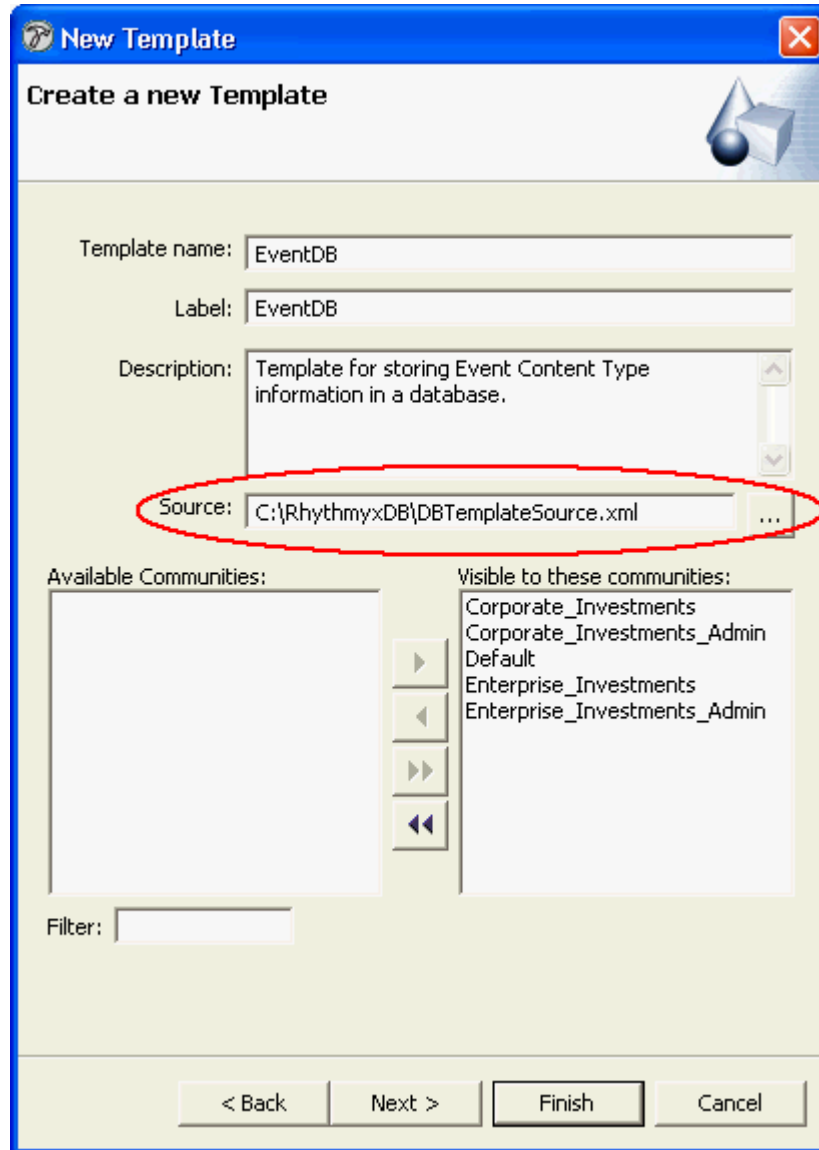


Figure 292: EventDB Template

- 8 Click [**Next**] to go to the Slots dialog.
- 9 Since you do not make Slots available to database Templates, click [**Next**] to go to the Content Types dialog.
- 10 Move *Event* from the Available Content Types list to the Associated Content Types list.
- 11 Click [**Finish**].

The New Template Wizard closes and the Workbench displays the EventDB Template in the Template editor.

The Source tab includes the XML source that you created in the Table Definition Builder. You do not have to make any changes to this tab.

- 12 Click the General tab.
- 13 Change the Active Assembly Format to *No HTML*.
- 14 Change the Mime type to *text/xml*.

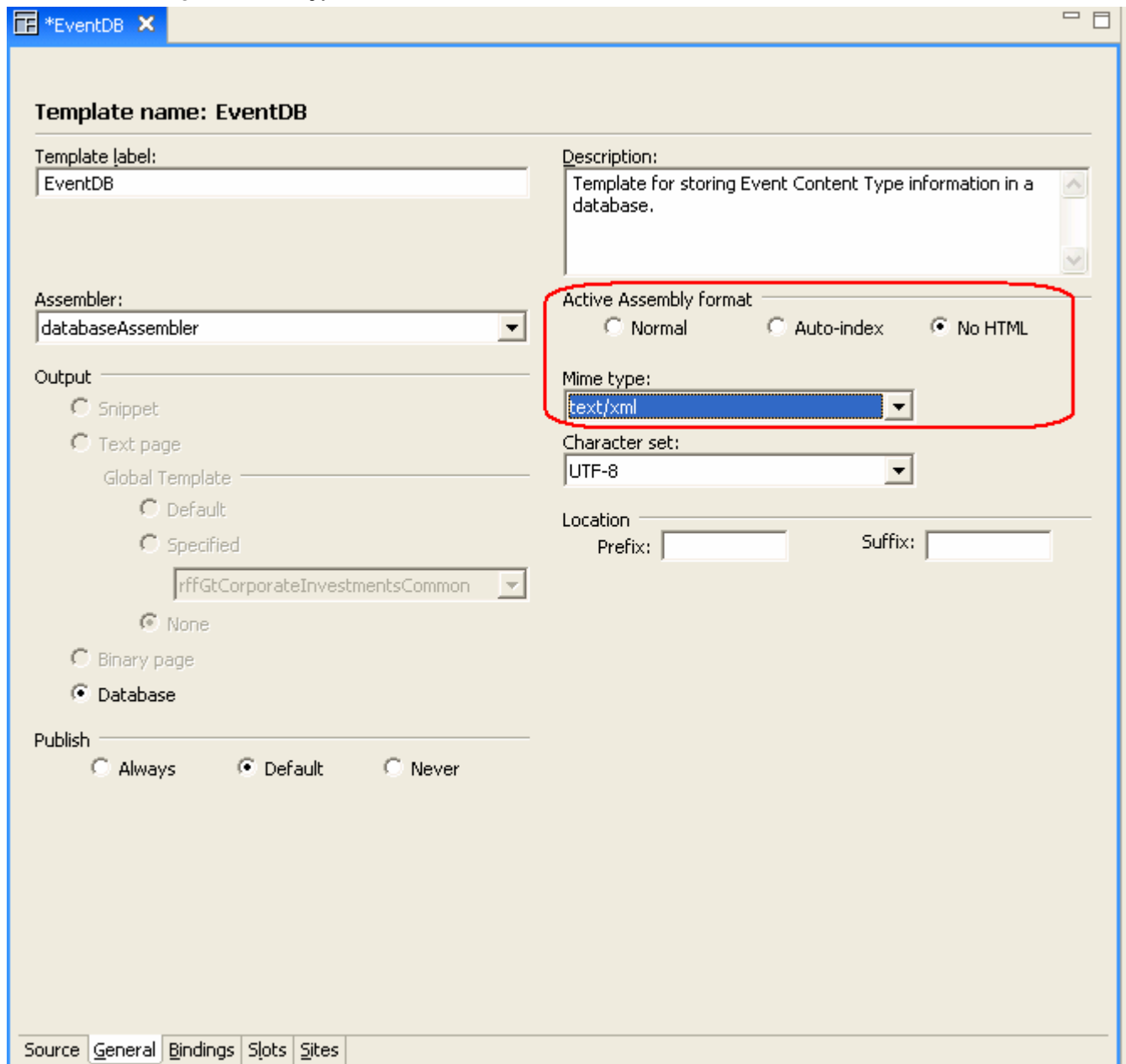


Figure 293: EventDB, General Tab

- 15 Click the Bindings tab.

On this tab, you must enter bindings that do the following:

- Map the rows in your target database tables to values. In our example, most of the values are taken from fields in the Event Content Type.

- Specify encoding for any specially formatted values, such as values stored in HTML markup.
- Specify connection information for the database plugin to use when connecting to the target database.
- Specify the action that you want performed on the content (for example, insert the row into the target database).

In this example, your bindings tab should resemble the following. In the table below the graphic, we explain each binding.

Variables:	
Variable Name	Value (JEXL expression)
\$db.action	"r"
\$db.origin	"dbo"
\$db.resource	"jdbc/TargetDB"
\$db.drivertype	"jtds:sqlserver"
\$db.database	"targetdb"
\$db.parent	"TARGET_CONTENT"
\$db.child[0]	"TARGET_LOCATION"
\$row.CONTENTID	\$sys.item.getProperty("rx:sys_contentid")
\$row.DISPLAYTITLE	\$sys.item.getProperty("rx:displaytitle")
\$row.CALLOUT	\$sys.item.getProperty("rx:callout")
\$row.\$encoding.CALLOUT	"base64"
\$row.BODY	\$sys.item.getProperty("rx:body")
\$row.\$encoding.BODY	"base64"
\$row.EVENT_START	\$sys.item.getProperty("rx:event_start")
\$row.EVENT_END	\$sys.item.getProperty("rx:event_end")
\$row.EVENT_TYPE	\$sys.item.getProperty("rx:event_type")
\$child[0].CONTENTID	\$sys.item.getProperty("rx:sys_contentid")
\$child[0].SEQ	\$rx.db.sequence(1,1)
\$child[0].EVENT_CITY	\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_city")
\$child[0].EVENT_STATE	\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_state")
\$child[0].EVENT_ADDRESS	\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_address")
\$child[0].EVENT_CONTACT	\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_contact")

Figure 294: Bindings for Database Publishing Template

Binding	Function	Value
\$db.action	Action to perform on the database with the content.	"r" - Inserts a row for the content. Deletes it first if it already exists. "n" - Inserts a row for the content if it does not already exist. "u" - Updates the row for the content if it already exists. "d" - Deletes the row for the content if it exists. If no value is entered, "r" is used as the default.
\$db.origin	Name of the target database schema.	"dbo"
\$db.resource	Name of the datasource	"jdbc/[datasource name]"

Binding	Function	Value
\$db.drivertype	Database driver type.	May be one of the following values, depending on your dbms: <ul style="list-style-type: none"> <li>▪ jtds:sqlserver</li> <li>▪ oracle:thin</li> <li>▪ db2</li> </ul>
\$db.database	Used for MS SQL Server if the database name is not specified in the datasource. Name of the database.	In this example: "targetdb". Optional in this example since it is specified in the datasource.
\$db.parent	Parent (or only) database table.	In this example: "TARGET_CONTENT"
\$db.child[0]	The first (or only) child table name. If you include additional child tables, increment the index by 1.  Do not include this variable if you are not using child tables.	In this example: "TARGET_LOCATION".
\$row.CONTENTID	Mapping for the specified column in the parent table. In this example, the mapping is for the CONTENTID column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$sys.item.getProperty("rx:sys_contentid")</code> gets the value of <code>sys_contentid</code> from the Content Item.  For more information about JEXL expressions, see the topic <i>Bindings</i> (see page 136).
\$row.DISPLAYTITLE	Mapping for the specified column in the parent table. In this example, the mapping is for the DISPLAYTITLE column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$sys.item.getProperty("rx:displaytitle")</code> gets the value of <code>displaytitle</code> from the Content Item.
\$row.CALLOUT	Mapping for the specified column in the parent table. In this example, the mapping is for the CALLOUT column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$sys.item.getProperty("rx:callout")</code> gets the value of <code>callout</code> from the Content Item.

Binding	Function	Value
\$row.\$encoding.CALLOUT	The type of encoding for the specified column.	<p>The database publishing plugin must encode special characters after retrieving data from the source database to prevent incorrect formatting when building the target database xml document. The plugin decodes the content before inserting it into the target database. The \$encoding parameter specifies how the plugin should encode and then decode each column of data from the source table.</p> <p>Currently, the only encoding type for Database Publishing is <code>base64</code>.</p> <ul style="list-style-type: none"> <li>▪ use <code>base64</code> for: <ul style="list-style-type: none"> <li>▪ binary data;</li> <li>▪ text data that includes HTML, XML, or SGML;</li> <li>▪ Data from a column in the source table that uses rich-text formatting.</li> </ul> </li> <li>▪ In other cases, do not use an encoding parameter.</li> </ul>
\$row.BODY	Mapping for the specified column in the parent table. In this example, the mapping is for the BODY column.	<p>A literal value or JEXL expression.</p> <p>In this example: the JEXL expression <code>\$sys.item.getProperty("rx:body")</code> gets the value of body from the Content Item.</p>
\$row.\$encoding.BODY	The type of encoding for the specified column.	The possible values for this column are "base64" or empty (no encoding). See the explanation of encoding in the \$row.\$encoding.CALLOUT column above.
\$row.EVENT_START	Mapping for the specified column in the parent table. In this example, the mapping is for the EVENT_START column.	<p>A literal value or JEXL expression.</p> <p>In this example: the JEXL expression <code>\$sys.item.getProperty("rx:event_start")</code> gets the value of event_start from the Content Item.</p>
\$row.EVENT_END	Mapping for the specified column in the parent table. In this example, the mapping is for the EVENT_END column.	<p>A literal value or JEXL expression.</p> <p>In this example: the JEXL expression <code>\$sys.item.getProperty("rx:event_end")</code> gets the value of event_end from the Content Item.</p>



Binding	Function	Value
\$row.EVENT_TYPE	Mapping for the specified column in the parent table. In this example, the mapping is for the EVENT_TYPE column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$sys.item.getProperty("rx:event_type")</code> gets the value of event_type from the Content Item.
\$child[0].CONTENTID	Mapping for the specified column in the child table. In this example, the mapping is for the CONTENTID column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$sys.item.getProperty("rx:sys_contentid")</code> gets the value of sys_contentid from the Content Item.
\$child[0].SEQ	Mapping of the sequence column in the child table. This column is required when child tables are used to allow the database plugin to move sequentially through all of the child table rows.	The <code>\$rx.db.sequence(start value, increment value)</code> function returns the start value the first time it is used, then adds the increment for each additional value. So <code>\$rx.db.sequence(1,1)</code> returns 1, 2, 3, 4, etc. as values.
\$child[0].EVENT_CITY	Mapping for the specified column in the child table. In this example, the mapping is for the EVENT_CITY column.	A literal value or JEXL expression. For child field set columns that have multiple rows, use the function: <code>\$rx.asmhelper.childValues(\$sys.item,"child field set name","rx:child field set field")</code> where: <code>\$sys.item</code> = the current item "child field set name" = the child field set whose column you are mapping. In this example "event_location". "rx:child field set field" = the field in the child field set that you are mapping to a column in the child table in the target database.  In this example: the JEXL expression <code>\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_city")</code> gets the value of event_city from the event_location child field set.
\$child[0].EVENT_STATE	Mapping for the specified column in the child table. In this example, the mapping is for the EVENT_STATE column.	A literal value or JEXL expression.  In this example: the JEXL expression <code>\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_state")</code> gets the value of event_state from the event_location child field set.

Binding	Function	Value
\$child[0].EVENT_ADDRESS	Mapping for the specified column in the child table. In this example, the mapping is for the EVENT_ADDRESS column.	A literal value or JEXL expression. In this example: the JEXL expression <code>\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_address")</code> gets the value of event_address from the event_location child field set.
\$child[0].EVENT_CONTACT	Mapping for the specified column in the child table. In this example, the mapping is for the EVENT_CONTACT column.	A literal value or JEXL expression. In this example: the JEXL expression <code>\$rx.asmhelper.childValues(\$sys.item,"event_location","rx:event_contact")</code> gets the value of event_contact from the event_location child field set.
<p>Notes on bindings and values:</p> <ul style="list-style-type: none"> <li>▪ To set encoding for a child column, use <code>\$child[0].\$encoding.&lt;column&gt;</code> in the Variable Name column.</li> <li>▪ In our example, we only used fields with single values in the child field set. If any of the fields had held multiple values, the binding value would have been different. In general, fields with the controls <code>sys_DropDownMultiple</code>, <code>sys_CheckBoxTree</code>, and <code>sys_CheckBoxGroup</code> may hold multiple values. For example, if EVENT_CONTACT used a <code>sys_DropDownMultiple</code> control and could hold more than one contact values, the binding and its value would be as follows: <b>Binding</b> <code>\$child[0].EVENT_CONTACT</code>    <b>Value</b> <code>\$sys.item.getProperty("rx:event_contact").getValues()</code></li> <li>▪ See the section <i>Binding Variables</i> (on page 409) for more detailed information about available binding variables and their syntax.</li> <li>▪ Note: If you want to query a non-Rhythmyx database table for values to insert into a target child table, use <i>\$rx.db.get</i> (see "\$rx.db" on page 420) and <i>\$rx.asmhelper.mapValues</i> (see "\$rx.asmhelper" on page 415).</li> </ul>		

- 1 Do not click the Slots tab. You do not make Slots available to database Templates.
- 2 Click the Sites tab.
- 3 Move *Database Site* from the **Hidden on these Sites** box to the **Visible on these Sites** box.
- 4 Save and close the EventDB Template.

---

## Registering the Database Publishing Context

Database publishing does not use a Context to determine where to publish content; it uses the information in a Context to list where database content has been published in the Publication Map when a user reviews a database publishing run. The absence of a database publishing Context and Location Scheme will not prevent you from performing database publishing.

To register a database publishing context:

- 1 In the Content Explorer Publishing tab create a new Context. See "Adding Contexts and Location Schemes" in the CMS Help for instructions.
- 2 Click the **Context Name** to reopen the Edit Context page.
- 3 Click New Location Scheme to open the Edit Location Scheme page.
- 4 Enter the following values:
  - Enter a **Name and Description**.
  - In the **Generator** drop list, choose *sys\_JexlAssemblyLocation*.
  - In the **Content Type** drop list, choose *Event*.
  - In the **Template Type** drop list, choose *EventDB*.
- 5 Click [**Save**] to save the Location scheme and open the Edit Context page.
- 6 Under **Location Scheme (id)**, click *Event* to open the Edit Location Scheme page.
- 7 Click New Location Scheme Parameter.  
The Edit Location Scheme Parameter page opens.
- 8 Enter the following values:
  - In **Name** enter *expression*.
  - In **Type** enter *String*.
  - In **Sequence** enter *1*.
  - In **Value** you could enter anything that would identify to the user publishing the data what has been published. One possibility is to show each Content Item's id number with the suffix *xml* to indicate that the data published was passed in an XML file. To do this, enter a variation on the Location Scheme Parameter Value that is used for Site Folder publishing, but only include the word *item*, the content ID, and the suffix *xml*. Enter:
    - `'item' + $sys.item.getProperty('rx:sys_contentid').String + '.xml'`
- 9 Click [**Save**].

The completed Edit Location Scheme Parameter page should appear similar to:

**Edit Location Scheme**

Location Scheme(id): Event(338)

\*Name: Event

Description: Location Scheme for Event Content Item data.

\*Generator: sys\_JexlAssemblyLocation

\*Content Type: Event

\*Template Type: EventDB

Save Cancel

**Location Scheme Parameters**

**New Location Scheme Parameter**

Name(id)	Type	Sequence	Value
✘ expression	String	1	'item' + \$sys.item.getProperty('rx:sys_contentid').String + '.xml'

Figure 295: Event Location Scheme

**10** Click [Save].

The Edit Location Scheme Parameter page closes and the completed Edit Context page opens.

**11** Click [Save].

---

# Creating the Database Publishing Content List

Create your database publishing Content List in Content Explorer. A few of the settings in the Database Publishing Content List will differ from that of a file system Content List.

To create the database publishing Content List:

- 1 Open Content Explorer.
- 2 Click the Publishing tab.
- 3 In the left navigation pane, click Content Lists/By Name.  
Content Explorer displays the Content List Administration page.
- 4 Click New Content List.  
Content Explorer displays the Edit Content List page.
- 5 In Name, enter *EventDBContentList*.
- 6 In Description, enter *Content List that selects items to publish to a database*.
- 7 In Url, change *sys\_deliverytype=filessystem* to *sys\_deliverytype=database*
- 8 Leave Type as *Normal*.
- 9 Leave Edition Type as *Automatic*.
- 10 In the Generator drop list, choose *sys\_searchGenerator*.  
The screen displays a field for entering a query.
- 11 In the query Value field, enter:  

```
select rx:sys_contentid, rx:sys_folderid from rx:Event where  
jcr:path like '//Sites/EnterpriseInvestments%'
```

  
This is a **JSR-170 query** (see page 187) that tells Rhythmyx to choose all Event Content Items in the Sites/EnterpriseInvestments folder path.
- 12 In the Template Expander field, choose *sys\_ListTemplateExpander*. In general, use this template expander whenever you perform database publishing. It enables you to specify the Template(s) that you want to use.  
The screen displays a field for entering Templates.
- 13 In the template Value field, enter *EventDB*.
- 14 Leave Item Filter as *Public*.

The Edit Content List page should appear similar to:

Edit Content List		Errors				
<b>Id</b>	0-21-339					
<b>Name</b>	EventDBContentList					
<b>Description</b>	Content List that selects items to publish to a database.					
<b>Url</b>	/Rhythmyx/contentlist?sys_deliverytype=database&sys_contentlist=EventDBContentList					
<b>Type</b>	<input checked="" type="radio"/> Normal <input type="radio"/> Incremental					
<b>Edition Type</b>	Automatic					
<b>Generator</b>	Java/global/percussion/system/sys_SearchGenerator					
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>query</td> <td>           select rx:sys_contentid, rx:sys_folderid from rx:Event where jcr:path like '/Sites/Er  <small>(Required) The JSR-170 sql style query to use in finding the base items for the content list.</small> </td> </tr> </tbody> </table>		Name	Value	query	select rx:sys_contentid, rx:sys_folderid from rx:Event where jcr:path like '/Sites/Er <small>(Required) The JSR-170 sql style query to use in finding the base items for the content list.</small>
Name	Value					
query	select rx:sys_contentid, rx:sys_folderid from rx:Event where jcr:path like '/Sites/Er <small>(Required) The JSR-170 sql style query to use in finding the base items for the content list.</small>					
<b>Template Expander</b>	Java/global/percussion/system/sys_ListTemplateExpander					
	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>template</td> <td>EventDB</td> </tr> </tbody> </table> <small>One or more template names, separated by commas. Templates must be registered as non-HTML.</small>		Name	Value	template	EventDB
Name	Value					
template	EventDB					
<b>Item Filter</b>	public					
<input type="button" value="Save"/> <input type="button" value="Cancel"/>						

Figure 296: Database Content List

**15** Click [Save].

Content Explorer closes the Content List Editor page. Your new Content List is now listed on the Content List Administration page.

---

# Creating the Database Publishing Edition

Create your database publishing Edition in Content Explorer.


To create your database publishing Edition:

- 1 Open Content Explorer.
- 2 Click the Publishing tab.
- 3 In the left navigation pane, click any of the Editions links.  
Content Explorer displays the Editions page.
- 4 Click New Edition.  
Content Explorer displays the Edit Edition Properties page.
- 5 In **Edition Name**, enter *EventDBEdition*.
- 6 In **Description**, enter *Edition for publishing a DB Content List*.
- 7 In the **Destination Site** drop list, choose *Database Site*.
- 8 Leave the rest of the fields at their default values.
- 9 Click [Save].  
Content Explorer closes the Edit Edition Properties page. The Editions page displays *EventDBEdition*.
- 10 Click the *EventDBEdition* link to open the Edit Edition Properties page again.  
Now the page displays a section for adding Content Lists.
- 11 Click Add Content List.  
Content Explorer displays the Edition Content List page.
- 12 In the **Content List** drop list, choose *EventDBContentList*.
- 13 In **Sequence**, enter *1*, since this is the only Content List that you will be publishing.
- 14 In **Authorization Type**, choose *All Public Content*.
- 15 In **Context**, choose *Database*.
- 16 Click [Save].

Content Explorer closes the Edition Content List page and displays the Edit Editions Properties page. The Edit Editions Properties page should appear similar to:

Content List Name (id)	Sequence	Authorization Type	Context	Preview
EventDBContentList (339)	1	All Public Content	Database	

Figure 297: EventDBEdition

- 17** To make sure that your Content List is configured correctly, click the Preview button  at the end of the Content List's row. You should see an XML file that includes all of your Public Event Content Items. The <location> element stores the value that you entered in your Location Scheme parameter.

```
<?xml version="1.0" encoding="utf-8" ?>
- <contentlist context="303" deliverytype="database">
- <contentitem contentid="706" revision="1" unpublish="no" variantid="555">
  <title>Trade Show 2006</title>
  <contenturl>http://10.10.10.187:9911/Rhythmyx/assembler/render?
    sys_revision=1&sys_siteid=305&sys_template=555&sys_itemfilter=public&sys_contentid=706&sys_folderid=306
  - <delivery>
    <location>item706.xml</location>
  </delivery>
  <modifydate>2006-10-12 09:11:34</modifydate>
  <modifyuser>admin1</modifyuser>
  <contenttype>318</contenttype>
</contentitem>
- <contentitem contentid="707" revision="1" unpublish="no" variantid="555">
  <title>User's Conference</title>
  <contenturl>http://10.10.10.187:9911/Rhythmyx/assembler/render?
    sys_revision=1&sys_siteid=305&sys_template=555&sys_itemfilter=public&sys_contentid=707&sys_folderid=306
  - <delivery>
    <location>item707.xml</location>
  </delivery>
  <modifydate>2006-10-12 09:11:42</modifydate>
  <modifyuser>admin1</modifyuser>
  <contenttype>318</contenttype>
</contentitem>
</contentlist>
```

Figure 298: EventDBContentList


- 18** Close the Content List and click [Save].



---

## Publishing to the Database

To regularly publish your database Edition, schedule it to publish like any other Edition.

To test that the Edition you just created publishes correctly, create some Content Items of the Event Content Type and transition them to a Public State. Then go to the Editions page and click the Publish button  next to *EventDBEdition*. If the Edition is publishing successfully, the same *progress messages* (see page 339) that appear when you manually publish any Edition appear on your screen now.

## Reviewing the Database Publication

To review your test database publishing run, review the Publication Log and check your database tables to make sure the data was written to them.

To check the Publication Log:

- 1 Click the Publishing tab in Content Explorer and click one of the Publication Log links. Locate the entry for your publication. Since your database Edition included one Content List, the Publication Log should include one entry for the database publication.

Database Site (395)					
Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status	
2006-10-12 09:11:54.0 (321:321)	EventDBContentList (329)	EventDBEdition		Success	

Figure 299: Database Publication Log Entry

Since this is the first time you have published to the database, the number of Content Items that you have published should be listed under the Inserted Content Items icon . The Status column either displays *Success* or *Error*. You can click on *Success* or *Error* to view a detailed log of the events that occurred during publishing.

- 2 Click the Date/Time link to see a list of the content published.

Filename	Operation	Status	Publication Details CMS Link
doc2server:1433/ targetdb/ item706.xml	publish	success	Trade Show 2006
item707.xml	publish	success	User's Conference

Figure 300: Publication Details

The path at the top of the Filename list, doc2server:1433/targetdb/ is the value you entered as your database Site's **Publishing Root Location**. The filenames listed use the pattern you specified for the Location Scheme. No real files with these names exist (if you click on the links, the system returns a page saying that the resource cannot be found); the filenames actually provide a list of the Content Items whose data was published to the database tables.

- 3 If you want to see the XML page that stored the datasource information, table definitions, and data published to the tables, click a title under CMS Link. You may have to enter your Rhythmyx username and password to open the file. In the <tabledataset> element at the bottom of the xml file, you should see the Content Item values inside tags with the correct target table column names:

```

<?xml version="1.0" encoding="utf-8" ?>
- <datapublisher drivertype="jtds:sqlserver" origin="dbo" resourceName="jdbc/TargetDB">
+ <tabledefset>
- <tabledataset>
  - <table name="TARGET_CONTENT">
    - <row action="r">
      <column name="EVENT_END" />
      <column name="CALLOUT"
        encoding="base64">PGRpdiBjbGFzcz0icnhib2R5ZmllbGQiPjxwPIRyYWRIIFNob
      <column name="DISPLAYTITLE">Trade Show 2006</column>
      <column name="BODY"
        encoding="base64">PGRpdiBjbGFzcz0icnhib2R5ZmllbGQiPjxwPkF0IFRyYWRII
      <column name="EVENT_START" />
      <column name="CONTENTID">706</column>
      <column name="EVENT_TYPE" />
    - <childtable name="TARGET_LOCATION">
      - <row action="r">
        <column name="EVENT_ADDRESS">20 Magnolia Drive</column>
        <column name="EVENT_STATE">CA</column>
        <column name="SEQ">1</column>
        <column name="EVENT_CITY">San Francisco</column>
        <column name="CONTENTID">706</column>
        <column name="EVENT_CONTACT">Jolie Cameron</column>
      </row>
      - <row action="r">
        <column name="EVENT_ADDRESS">100 Madison Avenue</column>
        <column name="EVENT_STATE">NY</column>
        <column name="SEQ">2</column>
        <column name="EVENT_CITY">New York</column>
        <column name="CONTENTID">706</column>
    - </table>
  - </tabledataset>
- </tabledefset>
- </datapublisher>

```

Figure 301: Data published

- 4 As a final check, look in the database tables that you published to and check that they are storing the information you just published.

	CONTENTID	DISPLAYTITLE	CALLOUT	BODY	EVENT_START	EVENT_END	EVENT_TYPE
▶	706	Trade Show 2006	<div class="rxt	<div class="rxbc	<NULL>	<NULL>	<NULL>
	707	User's Conference	<div class="rxt	<div class="rxbc	<NULL>	<NULL>	<NULL>
*							

Figure 302: TARGET\_CONTENT Table

	CONTENTID	SEQ	EVENT_CITY	EVENT_STATE	EVENT_ADDRESS	EVENT_CONTACT
▶	706	1	San Francisco	CA	20 Magnolia Drive	Jolie Cameron
	706	2	New York	NY	100 Madison Avenue	Dirk Anthony
	707	1	Phoenix	AZ	2000 Mesa Drive	Ariana Cabrio
	707	2	Boston	MA	100 Commonwealth	Steve Smith
*						

*Figure 303: TARGET\_LOCATION table*

---

## Unpublishing from the Database

To unpublish from your target database:

- 1 **Create a Template** (see page 375) that exactly resembles the Template that you created for publishing to the database.
- 2 In the Bindings tab, set \$db.action to "d" instead of "r".
- 3 **Create a Database Publishing Content List** (see page 385) exactly like the one you created for publishing, but specify your unpublish Template.
- 4 **Create an Edition** (see page 387) that includes the Unpublish Content List.
- 5 Publish the unpublish Edition.

When you check your logs and tables, all of the content included in the Content List should be deleted.

---

## Debugging Database Publishing

This page lists common types of errors that you may encounter during database publishing and their meanings.

- 1 An Error or Fatal Error link in the publications page indicates that your database publishing edition did not publish correctly. If you click the Error or Fatal Error link, a log opens.



*Figure 304: Link to Error Log*

You can scroll down the log until you find information about your error.

If the publisher was able to connect to the target database and attempt to publish data, you may see an error message in the following format:

**Rhythmyx**  
CONTENT MANAGEMENT

## Problem during the assembly of item

[Click here to view velocity log](#)

### Parameters passed

Name	Value
sys_revision	1
sys_siteid	305
sys_template	585
sys_itemfilter	public
sys_contentid	708
sys_folderid	306
sys_context	303
sys_publish	publish

### Error reported

Unexpected exception while assembling one or more items: java.lang.IllegalArgumentException: var must start with a dollar sign

*Please note: More information may be available on the console*

Figure 305: Assembly Error Message

The text under *Error reported* explains the error. In this case, a binding is missing a \$. You would return to your Template's Bindings tab and add the missing \$.

If your error occurs when the publisher attempts to connect to the target database, you must scroll down to the following type of messages to look for error information:

09:34:04 359ms	<b>ERROR</b>	Unexpected error:
<pre> javax.naming.NameNotFoundException: Targetdb not bound   at org.jnp.server.NamingServer.getBinding(NamingServer.java:514)   at org.jnp.server.NamingServer.getBinding(NamingServer.java:522)   at org.jnp.server.NamingServer.getObject(NamingServer.java:528)   at org.jnp.server.NamingServer.lookup(NamingServer.java:281)   at org.jnp.server.NamingServer.lookup(NamingServer.java:255)   at org.jnp.interfaces.NamingContext.lookup(NamingContext.java:610)   at org.jnp.interfaces.NamingContext.lookup(NamingContext.java:572)   at com.percussion.tablefactory.PSJdbcDbmsDef.lookupDatasource(Unknown Source)   at com.percussion.tablefactory.PSJdbcDbmsDef.getDataSource(Unknown Source)   at com.percussion.tablefactory.PSJdbcDbmsDef.&lt;init&gt;(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler\$DbmsConnection.&lt;init&gt;(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.connectToDatabase(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.performAction(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.publish(Unknown Source)   at com.percussion.publisher.client.PSContentItem.publish(Unknown Source)   at com.percussion.publisher.client.PSContentItem.process(Unknown Source) </pre>		

*Figure 306: Connection Error Message*

The following is a list of some common error listings in the log and their meanings.

a) **Name Not Found Exception**

You get a fatal error, and the first line of error text reads  
*javax.naming.NameNotFoundException: <dbname> not bound*



Meaning: For your Template's \$db.resource binding, the value you have entered does not match the value entered in the <jndi-name> element of your datasource file. The value is case sensitive, so check for differences in case sensitivity as well as spelling. In addition, enter `http://localhost:9992/RxServices/jnditest.jsp` in a browser to see additional messages about errors in the binding. See *Creating a Datasource for your Target Database* (see page 365) for more information about entering this command.

09:34:04 359ms	ERROR	Unexpected error:
<pre> javax.naming.NameNotFoundException: Targetdb not bound   at org.jnp.server.NamingServer.getBinding(NamingServer.java:514)   at org.jnp.server.NamingServer.getBinding(NamingServer.java:522)   at org.jnp.server.NamingServer.getObject(NamingServer.java:528)   at org.jnp.server.NamingServer.lookup(NamingServer.java:281)   at org.jnp.server.NamingServer.lookup(NamingServer.java:255)   at org.jnp.interfaces.NamingContext.lookup(NamingContext.java:610)   at org.jnp.interfaces.NamingContext.lookup(NamingContext.java:572)   at com.percussion.tablefactory.PSJdbcDbmsDef.lookupDatasource(Unknown Source)   at com.percussion.tablefactory.PSJdbcDbmsDef.getDataSource(Unknown Source)   at com.percussion.tablefactory.PSJdbcDbmsDef.&lt;init&gt;(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler\$DBmsConnection.&lt;init&gt;(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.connectToDatabase(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.performAction(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler.publish(Unknown Source)   at com.percussion.publisher.client.PSContentItem.publish(Unknown Source)   at com.percussion.publisher.client.PSContentItem.process(Unknown Source) </pre>		

Figure 307: Name Not Found Exception

#### b) Column Not Found in Table Schema

You get an error, and the first line of the error text reads:

*com.percussion.tablefactory.PSJdbcTableFactoryException: The column definition for the column (TEST) in the data for table (GENERIC\_CONTENT) was not found in the table schema.*

Meaning: In the Template definition, a column name in the Source tab does not match a binding variable column name in the Bindings tab. Return to the Template's binding tab in the Workbench, and make sure the column names match what is in the target database.

10:41:10 064ms	<b>ERROR</b>	Database item publishing failed. The error was:
<pre>com.peroussion.tablefactory.PSJdbcTableFactoryException: The column definition for the column (TEST) in the data for table (GENERIC_CONTENT) was not found in the table schema. at com.peroussion.tablefactory.PSJdbcTableSchema.validateTableData(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableSchema.setTableData(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableSchemaCollection.setTableData(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableFactory.processTables(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableFactory.processTables(Unknown Source) at com.peroussion.publisher.client.PSDatabasePublisherHandler.performAction(Unknown Source) at com.peroussion.publisher.client.PSDatabasePublisherHandler.publish(Unknown Source) at com.peroussion.publisher.client.PSContentItem.publish(Unknown Source) at com.peroussion.publisher.client.PSContentItem.process(Unknown Source) at com.peroussion.publisher.client.PSContentPublisher.process(Unknown Source) at com.peroussion.publisher.client.PSContentPublisher.execute(Unknown Source) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)</pre>		

Figure 308: Column not found in table schema

c) Data truncation

You get an error, and the first line of the error text reads: *java.sql.DataTruncation: Data truncation.*

Meaning: Column length in target table is not big enough (or is the wrong type) to accommodate the length of data being inserted. Either change the length of the column in the target table (or the data type) or make the length of the source data shorter.

11:03:31 991ms	<b>ERROR</b>	Database item publishing failed. The error was:
<pre>java.sql.DataTruncation: Data truncation at net.sourceforge.jtds.jdbc.SQLDiagnostic.addDiagnostic(SQLDiagnostic.java:379) at net.sourceforge.jtds.jdbc.TdsCore.tdsErrorToken(TdsCore.java:2781) at net.sourceforge.jtds.jdbc.TdsCore.nextToken(TdsCore.java:2224) at net.sourceforge.jtds.jdbc.TdsCore.getMoreResults(TdsCore.java:628) at net.sourceforge.jtds.jdbc.JtdsStatement.processResults(JtdsStatement.java:525) at net.sourceforge.jtds.jdbc.JtdsStatement.executeSQL(JtdsStatement.java:487) at net.sourceforge.jtds.jdbc.JtdsPreparedStatement.execute(JtdsPreparedStatement.java:475) at org.jboss.resource.adapter.jdbc.WrappedPreparedStatement.execute(WrappedPreparedStatement.java:183) at com.peroussion.util.PSPreparedStatement.execute(Unknown Source) at com.peroussion.tablefactory.PSJdbcPreparedStatement.execute(Unknown Source) at com.peroussion.tablefactory.PSJdbcReplaceExecutionPlan.execute(Unknown Source) at com.peroussion.tablefactory.PSJdbcReplaceExecutionPlan.reverseExecute(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableFactory.processTable(Unknown Source) at com.peroussion.tablefactory.PSJdbcTableFactory.processTables(Unknown Source)</pre>		

Figure 309: Data Truncation Error

d) Class not found

You get a fatal error, and the first line of the error text reads:

```
java.lang.ClassNotFoundException:
com.percussion.publisher.client.PSDatabasePublisherHandle
```

Meaning: When you registered the Publisher parameter *database*, you entered either the parameter Name *database* or its value `com.percussion.publisher.client.PSDatabasePublisherHandler` incorrectly. In the example below, the final *r* is missing on *Handler*.

11:33:57 520ms	<b>ERROR</b>	Publisher failed, error: com.percussion.publisher.client.PSDatabasePublisherHandle	publish siteId="" clientId="" skipped
<pre>java.lang.ClassNotFoundException: com.percussion.publisher.client.PSDatabasePublisherHandle at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1332) at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1181) at java.lang.ClassLoader.loadClassInternal(Unknown Source) at java.lang.Class.forName0(Native Method) at java.lang.Class.forName(Unknown Source) at com.percussion.publisher.client.PSContentItem.&lt;init&gt;(Unknown Source) at com.percussion.publisher.client.PSContentPublisher.process(Unknown Source) at com.percussion.publisher.client.PSContentPublisher.execute(Unknown Source) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source) at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source) at java.lang.reflect.Method.invoke(Unknown Source)</pre>			

Figure 310: Class Not Found Exception

e) Could not create connection - Login failed.

You get a fatal error, and the first line of the error text reads  
*org.jboss.util.NestedSQLException: Could not create connection; - nested throwable:*  
*(java.sql.SQLException: Login failed for user 'test'.); - nested throwable:*  
*(org.jboss.resource.JBossResourceException: Could not create connection; - nested*  
*throwable: (java.sql.SQLException: Login failed for user 'test'.))*

Meaning: In your datasource file (<Rhythmyx root>/AppServer/server/rx/deploy/<filename>-ds.xml file), you did not enter a correct username in the <connection-url> element's *user* parameter. In the following example, *test* is incorrect as the user:

```
<connection-
url>jdbc:jtds:sqlserver://doc2server:1433/targetdb;user=test;password=mypass</connecti
on-url>
```

12:00:16 754ms	<b>ERROR</b>	Unexpected error:	publisherId="301" publicationId="391" siteId="305" editionId="312" clientId="329" contentType="311" contentId="335" action="publish"	com.percussion.s
<pre>org.jboss.util.NestedSQLException: Could not create connection; - nested throwable: (java.sql.SQLException: Login failed for user 'test'.); - nested throwable: (org.jboss.resource.JBossResourceException: Could not create connection; - nested throwable: (java.sql.SQLException: Login failed for user 'test'.)) at org.jboss.resource.adapter.jdbc.WrapperDataSource.getConnection(WrapperDataSource.java:79) at com.percussion.tablefactory.PSJdbcDataSource.getConnection(Unknown Source) at com.percussion.tablefactory.PSJdbcTableFactory.getConnection(Unknown Source) at com.percussion.publisher.client.PSDatabasePublisherHandler\$DataSource.getConnection(Unknown Source) at com.percussion.publisher.client.PSDatabasePublisherHandler.connectToDatabase(Unknown Source) at com.percussion.publisher.client.PSDatabasePublisherHandler.performAction(Unknown Source)</pre>				

Figure 311: Could not create connection

f) Could not create connection - SSO failed.

You get a fatal error, and the first line of the error text reads *org.jboss.util.NestedSQLException: Could not create connection; - nested throwable: (java.sql.SQLException: I/O Error: SSO Failed: Native SSPI library not loaded. Check the java.library.path system property.);*

Meaning: In your datasource file (<Rhythmyx root>/AppServer/server/rx/deploy/<filename>-ds.xml file), you do not have <user-name> and <password> elements. See the topic **Creating a Datasource for your Target Database** (see page 365) for an example of where these elements belong.

13:18:14 082ms	ERROR	Unexpected error:	siteId="305" editionId="312" clientId="329" contentType="311" contentId="335" action="publish"
<pre>org.jboss.util.NestedSQLException: Could not create connection; - nested throwable: (java.sql.SQLException: I/O Error: SSO Failed: Native SSPI library not loaded. Check the java.library.path system property.); - nested throwable: (java.sql.SQLException: I/O Error: SSO Failed: Native SSPI library not loaded. Check the java.library.path system property.);     at org.jboss.resource.adapter.jdbc.WrapperDataSource.getConnection(WrapperDataSource.java:79)     at com.percussion.tablefactory.PSjdbcDbmsDef.getConnection(Unknown Source)     at com.percussion.tablefactory.PSjdbcTableFactory.getConnection(Unknown Source)     at com.percussion.publisher.client.PSDatabasePublisherHandler\$DbmsConnection.&lt;init&gt;(Unknown Source)     at com.percussion.publisher.client.PSDatabasePublisherHandler\$DbmsConnection.&lt;init&gt;(Unknown Source)</pre>			

Figure 312: Could not create connection

g) **SecurityException - Invalid Authentication Attempt**

You get a fatal error, and the first line of the error text reads *java.lang.SecurityException: Invalid authentication attempt, principal=null.*

Meaning: In your datasource file (<Rhythmyx root>/AppServer/server/rx/deploy/<filename>-ds.xml file), you have a <security-domain> element. Open your datasource file and comment out the element. See *Creating a Datasource for your Target Database* (see page 365) for information on the datasource file.

13:41:00 363ms	ERROR	Unexpected error:
<pre> java.lang.SecurityException: Invalid authentication attempt, principal=null   at org.jboss.resource.connectionmanager.BaseConnectionManager2.getSubject(BaseConnectionManager2.java:665)   at org.jboss.resource.connectionmanager.BaseConnectionManager2.allocateConnection(BaseConnectionManager2.java:461)   at org.jboss.resource.connectionmanager.BaseConnectionManager2\$ConnectionManagerProxy.allocateConnection(BaseConnectionManager2.java:894)   at org.jboss.resource.adapter.jdbc.WrapperDataSource.getConnection(WrapperDataSource.java:73)   at com.percussion.tablefactory.PSJdbcDbmsDef.getConnection(Unknown Source)   at com.percussion.tablefactory.PSJdbcTableFactory.getConnection(Unknown Source)   at com.percussion.publisher.client.PSDatabasePublisherHandler\$DbmsConnection.&lt;init&gt;(Unknown Source) </pre>		

Figure 313: Security Exception

- 2 Your log status is *Success*, but when you check your target database, the data is not present. If you click on Success and go through the log, you can find messages indicating that your tables are being created:

```

SQL[1]: CREATE TABLE dbo.GENERIC_CONTENT (ID INT NOT NULL, TITLE NCHAR(512) NULL, CALLOUT NTEXT NULL, BODY NTEXT NULL,
CONSTRAINT PK_GENERIC_CONTENT PRIMARY KEY (ID))

```

Figure 314: Create table message

If you look in the *master* database (In MS SQL Server) you can see that the tables have been created in that database, and the publisher has written the data to those tables.

Meaning: In your datasource file (<Rhythmyx root>/AppServer/server/rx/deploy/<filename>-ds.xml file), you have not added the the non-default port and the database name to the <connection-url> element. For an example of this element, see the topic *Creating a Datasource for your Target Database* (see page 365). By default, if the port and database name are not indicated, the publisher writes to the *master* database.



# Specialized Implementations

The Rhythmyx Implementation Guide documents the basic procedures for implementing a Rhythmyx Content Management System. A number of special implementation options are also available.

You may want to customize the Content Explorer interface, such as:

- adding new Display Formats;
- adding new Views and pre-defined Searches;
- customizing Lifecycle Analysis;
- customizing DocuComp (interface for comparing Content Items);
- enhancing accessibility;
- setting up the Active Assembly tutorial.

For details about performing these tasks, see *Customizing Content Explorer*.

Rhythmyx has been internationalized and can be localized. For details, see *Internationalizing and Localizing Rhythmyx*.

If you need to provide alternate clients for entering and maintaining content in Rhythmyx, see the *Web Services Development Kit*.

Rhythmyx can use WebDAV (Web Distributed Authoring and Versioning) to provide virtual access to Rhythmyx Folders. For details, see *Implementing WebDAV*.

Microsoft Word can be used as an alternative editing interface. For details, see *Implementing Word in Rhythmyx*.





## CHAPTER 12

## Next Steps

Once you have completed implementing your Rhythmyx design objects, you are ready to set up your production server (and any staging or quality assurance servers that you want to implement as well). For details about this process, see *Setting Up the Rhythmyx Production Environment*.



# Appendices



## APPENDIX I

# Binding Variables

Rhythmyx includes an extensive array of variables to use when creating bindings. Note that all binding variables must begin with the character "\$" Variables fall into the following categories:

Prefix	Category	Description
\$sys	<i>System Variables</i> (on page 410)	General system variables
\$rx	<i>System Functions</i> (on page 414)	System methods implemented by system-defined extensions
\$nav	<i>Navon Properties</i> (on page 290)	Variables used in Managed Navigation Templates.
\$db	<i>Database Publishing Variables</i> (on page 431)	Variables used to implement Database Publishing.
\$user	User-defined Variables and Functions	Prefix used for custom variables and functions developed by customers.
\$tools	<i>Velocity Tools Extensions</i> (see "Velocity Tool Extensions" on page 432)	Prefix used for Velocity tools extensions.

## System Variables

Simple system variables all begin with the string "\$sys".

Some of these variables come with pre-defined bindings to values. Others require the user to define the value for the variable. In several cases, (such as \$sys.mimetype and \$sys.charset) you can map an item property directly to the binding; these bindings do not execute a function.

For details about the variable type, see the *JavaDoc* (../JavaDocs/index.html).

Variable	Type	Description
\$sys.activeAssembly	Boolean	Bound to the value true if the Template is invoked for Active Assembly Preview.  Used in: Templates
\$sys.assemblyItem	<i>IPSAssemblyItem</i>	The current Content Item, assembled. This variable is required to call Slot Content Finders and to generate locations.  Used in: Templates
\$sys.binary	javax.jcr.Property or Byte Array	Value of the binary field for a template. Required for Binary Templates.  Used in: Templates
\$sys.charset	String	Value of the Characterset field of the Template. The bindings can override the value specified in the Template. Default value is <i>UTF-8</i> .  Used in: Templates
\$sys.crossSiteLink	Boolean	<i>True</i> if the Dependent Content Item in the ActiveAssembly Relationship is on a different Site.  Used in: Location Schemes
\$sys.currentslot	java.util.Map	Container for data calculated in the #initslot macro.  Used in: Templates
\$sys.ctx.Class.Name	String	Bound to the Velocity context. Required for the use of certain Velocity tools functions, such as \$tools.render.  Used in: Templates and Location Schemes
\$sys.index	Integer	A loop counter for Slot contents; can only be used in a Slot. The index of the Content Items in the Slot. Indexing begins at 1. (In other words, the first Content Item in the Slot is index1, the second index 2, and so on.) This variable can be used to modify the rendering of different rows in the Slot, or to add more data to the Snippet rendering.  Used in: Templates

Variable	Type	Description
\$sys.item	javax.jcr.Node	<p>Contains the fields and children of the current Content Item. Each field, and each child, is considered a property of the \$sys.item variable.</p> <p>Use the <code>getProperty</code> method to refer to a field of the Content Item. Usually, you will use the <code>getString</code> method to return a string, although you can also return other Java data types. For example, <code>\$sys.item.getProperty(displaytitle).getString</code> refers to the value of the <code>displaytitle</code> field of the current Content Item as a string value. For details see the JavaDoc for <code>javax.jcr.Property</code> in the JSR-170 JavaDoc.</p> <p><b>Used in:</b> Templates and Location Schemes</p>
\$sys.mimetype	String	<p>Value of the <code>MIMETYPE</code> field of the Template. The bindings can override the value specified in the Template. Required for Binary Templates. Default value is <i>text/html</i>.</p> <p><b>Used in:</b> Templates</p>

Variable	Type	Description																
\$sys.params	Map<String,String>[]	<p>Provides access to HTML parameters passed from the Content Item being assembled to the Template. An assembly request includes the following parameters:</p> <table border="1" data-bbox="779 420 1372 1669"> <thead> <tr> <th data-bbox="779 420 941 451">Name</th> <th data-bbox="941 420 1372 451">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="779 451 941 567">sys_path</td> <td data-bbox="941 451 1372 567">Path to the Content Item being assembled. Required.</td> </tr> <tr> <td data-bbox="779 567 941 672">sys_siteid</td> <td data-bbox="941 567 1372 672">The Site identifier. Optional</td> </tr> <tr> <td data-bbox="779 672 941 787">sys_context</td> <td data-bbox="941 672 1372 787">The name of the Context used to assemble the Content Item. Required.</td> </tr> <tr> <td data-bbox="779 787 941 934">sys_itemfilter</td> <td data-bbox="941 787 1372 934">The name of the Item Filter to use to filter Related Content when assembling the Content Item. Required.</td> </tr> <tr> <td data-bbox="779 934 941 1050">sys_template</td> <td data-bbox="941 934 1372 1050">The name of the Template to use to assemble the Content Item. Required.</td> </tr> <tr> <td data-bbox="779 1050 941 1333">sys_command</td> <td data-bbox="941 1050 1372 1333">If specified with the value <i>editrc</i>, the Content Item is assembled in Active Assembly mode, including Active Assembly decorations. If specified as null or any other value, the Content Item is assembled without Active Assembly decorations. Required.</td> </tr> <tr> <td data-bbox="779 1333 941 1669">sys_reinit</td> <td data-bbox="941 1333 1372 1669">If specified with the value <i>true</i>, the Velocity engine is re-initialized before assembling the Content Item, which causes all macros to be reloaded. If not included or specified with the value <i>false</i>, the Velocity engine is not re-initialized before assembling the Content Item and cached Templates are used. Optional.</td> </tr> </tbody> </table> <p>Used in: Templates and Location Schemes</p>	Name	Description	sys_path	Path to the Content Item being assembled. Required.	sys_siteid	The Site identifier. Optional	sys_context	The name of the Context used to assemble the Content Item. Required.	sys_itemfilter	The name of the Item Filter to use to filter Related Content when assembling the Content Item. Required.	sys_template	The name of the Template to use to assemble the Content Item. Required.	sys_command	If specified with the value <i>editrc</i> , the Content Item is assembled in Active Assembly mode, including Active Assembly decorations. If specified as null or any other value, the Content Item is assembled without Active Assembly decorations. Required.	sys_reinit	If specified with the value <i>true</i> , the Velocity engine is re-initialized before assembling the Content Item, which causes all macros to be reloaded. If not included or specified with the value <i>false</i> , the Velocity engine is not re-initialized before assembling the Content Item and cached Templates are used. Optional.
Name	Description																	
sys_path	Path to the Content Item being assembled. Required.																	
sys_siteid	The Site identifier. Optional																	
sys_context	The name of the Context used to assemble the Content Item. Required.																	
sys_itemfilter	The name of the Item Filter to use to filter Related Content when assembling the Content Item. Required.																	
sys_template	The name of the Template to use to assemble the Content Item. Required.																	
sys_command	If specified with the value <i>editrc</i> , the Content Item is assembled in Active Assembly mode, including Active Assembly decorations. If specified as null or any other value, the Content Item is assembled without Active Assembly decorations. Required.																	
sys_reinit	If specified with the value <i>true</i> , the Velocity engine is re-initialized before assembling the Content Item, which causes all macros to be reloaded. If not included or specified with the value <i>false</i> , the Velocity engine is not re-initialized before assembling the Content Item and cached Templates are used. Optional.																	
\$sys.path	String	<p>The path of the Folder where the Content Item resides. The path returned by this variable only uses the name specified for the Folder, not the alternative name specified in the Folder Properties.</p> <p>Used in: Location Schemes</p>																



Variable	Type	Description
\$sys.pub.path	String	The publication path of the Content Item. If any Folder in the path has an alternative name specified in the Folder Properties, that name is used in the path returned by this variable.  Used in: Location Schemes
\$sys.site.globalTemplate	String	Name of the Global Template of the Site. If the Global Template is undefined, returns null.  Used in: Location Schemes
\$sys.site.id	<i>IPSGuid</i>	ID of the Site, as a GUID  Used in: Templates and Location Schemes
\$sys.site.path	String	Path in the Folder tree from the current Content Item to the Site Folder  Used in: Templates and Location Schemes
\$sys.site.url	String	URL of the Site defined in the Site Address field in the Site registration.  Used in: Location Schemes
\$sys.template	String	The input text of the Velocity template. Used to override the default text of the template. (NOTE: overriding the default text requires a custom extension.).  Used in: Templates
\$sys.template.prefix	String	Location prefix from the Template definition.  Used in: Location Schemes
\$sys.template.suffix	String	Location suffix from the Template definition.  Used in: Location Schemes
\$sys.variables	Map<String,String>[]	Provides access to context variables for the current Site.  Used in: Templates and Location Schemes

---

## System Functions

Binding variables that begin with the prefix \$rx are system functions.

Note that you can also write your own binding functions, which must use the prefix \$user. For details, see the *Rhythmyx Technical Reference Manual*.

- ***\$rx.asmhelper*** (on page 415): provide data for use in assembly
- ***\$rx.codec*** (on page 418): encode and decode data
- ***\$rx.cond*** (on page 419): evaluate conditional statements (Note: This binding function is deprecated)
- ***\$rx.db*** (on page 420): Database Publishing
- ***\$rx.doc*** (on page 420): process XML and HTML documents
- ***\$rx.ext*** (on page 421): call a Rhythmyx extension
- ***\$rx.guid*** (on page 422): retrieve GUIDs
- ***\$rx.i18n*** (on page 422): internationalization
- ***\$rx.keyword*** (on page 422): provide access to Keyword data
- ***\$rx.link*** (on page 424): manipulate links
- ***\$rx.location*** (on page 425): generate hypertext links
- ***\$rx.nav*** (on page 427): Managed Navigation processing
- ***\$rx.session*** (on page 428): returns session IDs
- ***\$rx.string*** (on page 428): manipulate string values

## \$rx.asmhelper

The methods of this function provide data for use in assembly. The following methods are available:

- `$rx.asmhelper.assemble`
- `$rx.asmhelper.isAASlot (slot)`
- `$rx.asmhelper.getPopupMenu`
- `$rx.asmhelper.getSingleParamValue`
- `$rx.asmhelper.getTitle ($sys.item.guid)`
- `$rx.asmhelper.combine`
- `$rx.asmhelper.childValues`
- `$rx.asmhelper.mapValues`

### \$rx.asmhelper.assemble

Returns `List<IPSAsemblyResult>`

Used in Templates

Assembles the related Content Items in the specified Slot of the specified Content Item using the specified parameter values as overrides.

Name	Type	Description
item	<i>IPSAsemblyItem</i>	The Content Item whose related Content Items to assemble. Use <code>\$sys.item</code> .
slot	<i>IPSTemplateSlot</i>	The Slot whose related Content Items to assemble.
params	<code>Map&lt;String, Object&gt;</code>	Set of assembly parameters to use as overrides.

Example: `$rx.asmhelper.assemble($sys.item, rffList, "template='rffSnTitleCalloutLink' ")` returns the contents of the `rffList` Slot assembled using the `rffSnTitleCalloutLink` Template.

### \$rx.asmhelper.isAASlot (slot)

Returns `boolean`

Used in Templates

Returns true if the specified Slot is an Active Assembly Slot; otherwise returns false. The Slot Content Finder defines whether the Slot is an Active Assembly Slot. Of the Content Finders shipped with Rhythmyx, only Slots that use the `sys_RelationshipContentFinder` are considered Active Assembly Slots and will return true.

**\$rx.asmhelper.getPopupMenu**

Returns String

Used in Templates

Used internally. Returns the popup menus used in Active Assembly. The menuname parameter is optional; if not included, the menu is named using the Content ID of the specified Content Item.

Name	Type	Description
itemguid	<i>IPSGuid</i>	The GUID of the Content Item whose menus to retrieve. Use <code>\$sys.item.guid</code> .
mode	String	The operational Mode of Content Explorer for which to display the Menu. An operational Mode is a portion of Content Explorer for which unique menus can be displayed. For additional details, see "Action Menus" in <i>Implementing the Business User Interface</i> .
uicontext	String	The Visibility Context for which to display the Menu.
menuname	String	Name to display for the menu.

**\$rx.asmhelper.getSingleParamValue**

Returns String

Used in Templates

Returns the value of the specified parameter. If the specified parameter does not have a value, returns null.

Name	Type	Description
params	Map<String,String>	Set of Content Item parameters. Use <code>\$sys.params</code> .
key	String	name of the parameter whose value you want to return.

Example: `$rx.asmhelper.getSingleParamValue ($sys.params, sys_title)` returns the value in the `sys_title` field for the current Content Item.

**\$rx.asmhelper.getTitle (\$sys.item.guid)**

Returns String

Used in Templates

Returns the title of the specified Content Item.

**\$rx.asmhelper.combine**

Returns Map&lt;String, Object&gt;

Used in Templates

Generates a new parameter map with the the parameters from the urlquery "overlaid" over the parameters in the input parameter map. The original input parameters still exist and are unchanged. If a later function uses a parameter from the map that exists in both the original parameter map and the urlquery, the value of the parameter from the urlquery supersedes the value from the original parameter map. If a later function uses a parameter that does not exist in the urlquery, it takes the value from the original parameter map.

The value of the urlquery parameter can refer to another function. In that case, you must develop a function that returns a parameter string. For example, you could develop the function `$user.example` to return the set of overlay parameters. To use the output of this function, you would define the binding value `$rx.asmhelp.combine($sys.params, $user.example)`.

Name	Type	Description
params	Map<String, String[]>	Set of Content Item parameters. Use \$sys.params.
urlquerystring	String	URL-query-style string, with names and values separated by equals, and each name and value pair separated by ampersands

**\$rx.asmhelper.combine**

Returns Map&lt;String, Object&gt;

Used in Templates

Generates a new parameter map with the the parameters from the added parameter "overlaid" over the parameters in the input parameter map. The original input parameters still exist and are unchanged. If a later function uses a parameter from the map that exists in both the original parameter map and the added parameter, the value of the parameter from the added parameter supersedes the value from the original parameter map. If a later function uses a parameter that does not exist in the added parameter, it takes the value from the original parameter map.

Name	Type	Description
params	Map<String, String[]>	Set of Content Item parameters. Use \$sys.params.
added	Map<String, String[]>	Map of additional parameters,

**\$rx.asmhelper.childValues**

Returns List&lt;Object&gt;

Used in Templates

Used for database publishing. Returns a list of the values of the specified child Fieldset.

Name	Type	Description
parentNode	javax.jcr.Value	The Content Item whose child Fieldset children to return. Use \$sys.item.
childName	javax.jcr.Value	Name of the child Fieldset whose values you want to return.
PropertyName	javax.jcr.Value	Name of the fields to return from the child Fieldset

Example: `$rx.asmhelper.childValues($sys.item, Location, "city")` would return a list of the values of the field city in the Location child Fieldset of the current Content Item.

### **\$rx.asmhelper.mapValues**

Returns List<Object>

Used in Templates

Used for database publishing. Takes the output of `$rx.db.get` (a list of values of specified columns in a database returned as a list of maps) as input. Returns the list of the values of the specified columns for inserting into the child table.

## **\$rx.codec**

The methods of this function encode and decode data.

- `$rx.codec.base64Decoder`
- `$rx.codec.base64Encoder`
- `$rx.codec.escapeForXml`
- `$rx.codec.decodeFromXml`

### **\$rx.codec.base64Decoder(string)**

Returns String

Used in Templates

Decodes the string passed from base-64 encoding. The value to be decoded must be passed as a string.

Example:

```
$rx.codec.base64Decoder($sys.item.getProperty("body").String())  
would decode the value of the body field from base-64 coding to UTF-8 coding and return the resulting string.
```

### **\$rx.codec.base64Encoder(string)**

Returns String

Used in Templates

Encodes the string passed to base-64 coding. The value to be encoded must be passed as a string.

Example:

```
$rx.codec.base64Encoder($sys.item.getProperty("body").String())  
would encode the value of the body field to base-64 and return the resulting string.
```

**\$rx.codec.escapeForXml(string)**

Returns String

Used in Templates

Escapes the any characters in the input string specified that must be escaped for XML, including:

- & (ampersand)
- " (double quotation marks)
- < (greater than symbol)
- > (less than symbol)
- ' (single quotation mark)

. Example:

```
$rx.codec.base64Encoder($sys.item.escapeForXml($sys.item.getProperty("body").String()))
```

would escape any of the characters specified above that occurred in the body field and return the string so it could be used in an XML document.

**\$rx.codec.decodeFromXml(string)**

Returns String

Used in Templates

Converts the embedded XML entities in the input string to characters.

**\$rx.cond**

NOTE: This function is deprecated. JEXL expressions that use this binding should be rewritten to use the JEXL `if...else` conditional function instead.

The method of this function is used to evaluate conditional statements.

**\$rx.cond.choose**

Returns Object

If the boolean condition evaluate to *true*, returns the first value; otherwise, returns the second value.

Name	Type	Description
boolean	Object	The boolean expression to evaluate
truevalue	Object	The value to return if the boolean expression evaluates to <i>true</i> .
falsevalue	Object	The value to return if the boolean expression evaluates to <i>false</i> .

Example:

```
$rx.cond.choose($location=="above", "rffSnTitleAndImage", "rffSnImageAndTitle")
```

returns *rffSnTitleAndImage* if the value of *\$location* is *above*, otherwise it returns *rffSnImageAndTitle*.

## \$rx.db

The method of this function is used in database publishing.

Note: Be careful not to confuse this function with the Database Publishing variables. Note that those variables all begin with the prefix \$db, while this function begins with \$rx.db. If you attempt to specify this function as \$db.get, the system will return an error.

### \$rx.db.get

Returns List<Map<String, Object>>

Used in Templates

Executes the specified SQL query on the specified datasource and returns the results as a list of entries. Each entry consists of a column name and a value.

Name	Type	Description
datasource	String	Name of the datasource on which to execute the query. To specify the Rhythmyx datasource, specify an empty string.
selectStatement	String	SQL query to execute on the specified datasource.

Example: `$rx.db.get("RhythmyxData", "select LANGUAGESTRING, DISPLAYNAME from RXLOCALES where LOCALE_ID<10")` would return a list of the values of the LANGUAGESTRING and DISPLAYNAME columns in the database specified by the Rhythmyx Data datasource. The results are returned as a map array.

## \$rx.doc

The methods of this function process XML and HTML documents.

### \$rx.doc.getDocument(url)

Returns String

Used in Templates

Calls the URL specified as the parameter and returns the result document data. Use this method to call an existing Rhythmyx application. If an error occurs, returns an empty string. If the URL is defined as relative (../), the request is submitted internally.

Name	Type	Description
url	String	URL to query for the document.

### \$rx.doc.getDocument(url,user,password)

Returns String

Used in Templates

Alternate signature of the `$rx.doc.getDocument` function that includes the user and password parameters to access an external resource that requires authentication.



Name	Type	Description
url	String	URL to query for the document.
user	String	User name to use to access external resource. (Optional)
password	String	Password to use to access external resource. (Optional)

### **\$rx.doc.extractBody**

Returns String

Used in Templates

Extracts the body from the submitted data.

Name	Type	Description
rval	<i>IPSAsemblyResult</i> (Javadocs/com/perc ussion/services/asse mby/IPSAsembly Result.html)	The assembled result from which to extract the body. Optional; not used if the text parameter is used.
text	String	The HTML document from which to extract the body. Optional; not used if the rval parameter is used.

### **\$rx.ext**

The method of this function allows you to call an extension.

#### **\$rx.ext.call**

Returns Object

Used in Templates and Location Schemes

Calls the specified extension, with the specified parameters. Returns whatever the requested extension returns.

Name	Type	Description
name	String	Name of the extension to call
parameters	String	Parameters to submit to the requested extension. Up to 20 parameters can be submitted as strings.

## \$rx.guid

The method of this function allows you to retrieve GUIDs.

### \$rx.guid.getContentId(\$sys.item.guid)

Returns int

Used in Templates and Location Schemes

Extracts the Content Item ID from the submitted GUID. The GUID must be the ID of a Content Item. If the GUID is any other type of object, the method fails.

## \$rx.i18n

The method of this function is used to retrieve internationalized and localized data.

### \$rx.i18n.getString

Returns String

Used in Templates

For the specified key, looks up the localized text for the specified Locale.

Name	Type	Description
key	String	The key whose translated text to look up.
locale	String	The Locale for which to look up the text of the key..

Example: `$rx.i18n.getString ("psx.role@admin", "fr-fr")` returns the French translation of the Role admin for France.

## \$rx.keyword

The methods of this function provide access to Keyword data.

- `$rx.keyword.keywordSelectChoices`
- `$rx.keyword.keywordChoices`
- `$rx.keyword.getLabel`

### \$rx.keyword.keywordSelectChoices

Returns String

Used in Templates

Returns the set of Choices and Values for the specified Keyword formatted as a set of HTML `<SELECT>` elements.

Name	Type	Description
keywordname	String	Name of the Keyword whose Choices and Values to return.
currentchoice	String	Specifies the currently selected Choice of the Keyword

### **\$rx.keyword.keywordChoices**

Returns List<String[]>

Used in Templates

.Returns a list of results. Each result is an array consisting of two elements. The first element of this array is the label of the keyword choice, the second is the value of the keyword choice.

Name	Type	Description
keywordname	String	Name of the Keyword whose Choices and Values to return.

### **\$rx.keyword.getLabel**

Returns String

Used in Templates

Returns the Label of the specified Value of the specified Keyword.

Name	Type	Description
keywordname	String	Name of the Keyword for which to return a Label.
keywordvalue	String	Name of the Keyword Value for which to return the Label.

### **\$rx.keyword.getLabel**

Returns String

Used in Templates

Alternative signature of the \$rx.keyword.getLabel function that can return the localized value of the label. The locale for which to return the label is specified in the locale parameter. The local parameter cannot be null or empty.

Name	Type	Description
keywordname	String	Name of the Keyword for which to return a Label.
keywordvalue	String	Name of the Keyword Value for which to return the Label.
locale	String	Locale code of the Locale for which to return the localized Label.

### **\$rx.keyword.getChoiceLabel**

Returns String; the string may be empty if

- the specified Content Type does not exist;

- the specified field does not exist or does not contain a choice list; or
- the specified value does not exist in the choice list.

Used in Templates

Retrieves the specified field for the specified Content Type and returns the label for the specified choice. If any problems occur, an error is logged and a runtime exception is thrown.

Name	Type	Description
contenttypename	String	Name of the Content Type whose field is being specified. Cannot be null or empty.
fieldname	String	Name of the field of the specified Content Type for which to return the choice label.
choicevalue	String	The choice value whose name to return.

## **\$rx.link**

The methods of this function allow you to manipulate links.

- `$rx.link.addParams`
- `$rx.link.getAbsUrl`
- `$rx.link.getRelUrl`

### **\$rx.link.addParams**

Returns String

Used in Templates

Adds up to five name-value pairs to the specified URL.

Name	Type	Description
url	String	URL to which to add the additional parameters.
param1... param5	String	Parameter to add to the URL. Up to five parameters can be added. Each parameter must be match with a value (e.g., param1, value 1, param2,value2...param5,value5). If any parameter is missing a corresponding value, the system returns an error.
value1... value5	String	Value for the parameter.

### **\$rx.link.getAbsUrl**

Returns String

Used in Templates

Returns the absolute URL of the request path. Can be used with the addParams method to extend the URL with additional parameters.

Name	Type	Description
path	String	Partial path that will be appended to the Rhythmyx URL to produce an absolute URL.
secure	boolean	If specified as true, generates a secure link. If omitted, defaults to false.

### **\$rx.link.getRelUrl**

Returns String

Used in Templates

Creates a fully-qualified URL through the Rhythmyx server for the specified request root. For example, if alpha/beta were submitted, the URL `http://127.0.0.1:9992/Rhythmyx/alpha/beta` would be returned, where 127.0.0.1 was the IP address of the Rhythmyx server and 9992 was its port.

Name	Type	Description
path	String	Path whose URL base you want to retrieve.
addParams	String	HTML parameters to add to the output URL.

## **\$rx.location**

The methods of this function allow you to generate hypertext links.

- `$rx.location.generate`
- `$rx.location.getFirstDefined`
- `$rx.location.siteBase($sys.site)`

### **\$rx.location.generate**

Returns String

Used in Templates and Location Schemes

Generates the target URL for the assembly item using the default Template.

Can be specified with the template parameter, in which case the link will be generated to the specified Page Template. The value of the Template parameter must be the name of a page Template.

Name	Type	Description
targetItem	<i>IPSAsemblyItem</i> (../Javadocs/com/percussion/services/assembly/IPSAssemblyItem.html)	The target Content Item to which the URL will point.
targetTemplate	String	The specific Template to which the URL will point.

**\$rx.location.generate**

Returns String

Used in Templates and Location Schemes

Additional signature of the \$rx.location.generate function that allows the user to specify the data used to generate the URL.

Name	Type	Description
templateinfo	String	The Page Template to which the URL will point.
item	Node	The target Content Item to which the URL will point.
folderPath	String	The Folder location to which the URL will point.
filter	String	The Item Filter to use when generating the URL.
siteid	Number	The ID of the Site to which the URL will point.
context	Number	The publishing Context for which to generate the URL.

**\$rx.location.getFirstDefined**

Returns String

Used in Templates and Location Schemes

Looks through the set of fields specified to find a field that contains a value. The first field in the list specified that contains a value will be used. If none of the specified fields contains a value, the default value will be used.

Name	Type	Description
item	Node	The Content Item whose fields to search.
listofproperties	String	Comma-separated list of fields to search for a value.
defaultvalue	String	default value to use if none of the specified fields contains a value.

**\$rx.location.siteBase**

Returns String

Used in Templates and Location Schemes

Returns the base URL for the specified Site. Can include the modify parameter. If the modify parameters is set to yes [`siteBase($sys.assemblyitem, yes)`], the protocol, host, and port are not included in the returned URL.

Name	Type	Description
siteid	String	The Site whose base URL to return
modify	String	If the value of this parameter is "yes", the protocol, host, and port will be stripped from the base URL. Otherwise, the protocol, host, and port are included. (Optional)

## \$rx.nav

The methods of this function are used in processing Managed Navigation. They are only valid when applied to nodes returned from the Managed Navigation Slot Content Finder.

Note: Be careful not to confuse the methods of this function with the Managed Navigation variables, which begin with the prefix \$nav. If you attempt to specify this function as \$nav, the system will return an error.

### \$rx.nav.findProperty

Returns Property

Used in Templates

Starting at the specified node, searches up the navigation hierarchy to find a node that has a value for the specified property.

Name	Type	Description
node	Node	The node from which to start the search
propertyname	String	The name of the property to search for.

### \$rx.nav.findNode

Returns Node

Used in Templates

Starting at the specified node, searches up the navigation hierarchy until it finds a node that has a child node with the specified name.

Name	Type	Description
node	Node	The node from which to start the search
childname	String	The name of the child node to search for.

## \$rx.session

The methods of this function return session IDs that can be returned to Rhythmyx when calling Rhythmyx applications or other URLs via HTTP.

- `$rx.session.getSessionID`
- `$rx.session.getJSessionID`

### \$rx.session.getJSessionID

Returns String

Used in Templates

Returns the JSession ID. The ID must be passed back to Rhythmyx in one of the following ways:

- a cookie called JSESSION
- as part of the URL using the required syntax for JSESSION IDs.

### \$rx.session.getSessionID

Returns String

Used in Templates

Returns the session ID. The ID must be passed back to Rhythmyx in a HTTP parameter called PSESSIONID.

## \$rx.string

The methods of this function allow you to allow you to manipulate string values.

- `$rx.string.stringTo Map`
- `$rx.string.equalNumbers`
- `$rx.string.extractNumber`

### \$rx.string.stringTo Map

Returns Map<String, String>

Used in Templates

Converts a URL-style parameter string to a Java map. Used to pass parameters to Slot Content Finders.

Name	Type	Description
paramstring	String	Parameter string to convert. If null, an empty map is returned.



**\$rx.string.equalNumbers**

Returns boolean

Used in Templates and Location Schemes

Compares two parameters, each a number or a string with a numeric value (which will be converted to a number; if a parameter value cannot be converted to a number or if a parameter is of an unknown type, it is converted to 0). If the two numbers are equal, returns true.

Name	Type	Description
a	Object	The first object to compare
b	Object	The second object to compare.

**\$rx.string.extractNumber**

Returns long

Used in Templates and Location Schemes

Extracts the numerical value of the parameter. If the object is of the type `java.lang.Number`, returns the Long value of that number. If the object is of the type `java.lang.String`, the string is converted to a number; the default value in this case is 0. If the data type of the object is unknown, returns 0.

Name	Type	Description
a	Object	The object from which to extract the number.

**\$rx.string.stripSpace**

Returns string

Used in Templates and Location Schemes

Removes leading and trailing whitespace, and replace any embedded sequence of whitespace characters with a single space each.

Name	Type	Description
src	String	The source string to process for extraneous whitespace. May be empty but never null.

## Navon Properties

Navons have several unique properties and child nodes that play an important role in implementing Managed Navigation.

Navon properties are only used in Templates.

Variable	Data Type	Description
nav:axis	String	The axis of the Navon being processed in relation to the Navon from which processing was initiated. Available options include: <ul style="list-style-type: none"> <li>▪ ANCESTOR: a node in in the path of the Navon higher than the PARENT Navon node</li> <li>▪ DESCENDANT: A node in the path after the Navon</li> <li>▪ NONE: No other category applies</li> <li>▪ PARENT: The immediate predecessor of the Navon in its path.</li> <li>▪ SELF: The Navon itself.</li> <li>▪ SIBLING: Another Navon that shares the PARENT of the Navon.</li> </ul>
nav:url	String	The URL of the Navon's landing page.
nav:landingPage	Node	The landing page Content Item.
nav:leaf	Boolean	Boolean specifying whether the Navon is a leaf (in other words, has no children) <ul style="list-style-type: none"> <li>▪ True: The Navon is a leaf (has no children).</li> <li>▪ False: The Navon is not a leaf (has children).</li> </ul>
nav:submenu	Node Iterator	The variable contains all Navon children of the Navon being processed.
nav:image	Node Iterator	This variable contains all NavImage children of the Navon being processed.
nav:selectedImage	Node	The NavImage selected by the Selector

## Database Publishing Variables

The following variables have been defined specifically for use in Database Publishing. For additional details, see the *Rhythmyx JavaDoc*.

Note: Be careful not to confuse these variables with the Database function, \$rx.db. If you specify these variables with the prefix \$rx.db, the system will return an error.

Database Publishing Variables are used only in Templates.

Variable	Type	Description										
\$db.action	String	The action to perform. Always a single alphabetic character: <table border="1" data-bbox="808 764 1430 1052"> <thead> <tr> <th>Value</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>r</td> <td>Inserts the row. If the row already exists, it is deleted first.</td> </tr> <tr> <td>n</td> <td>Inserts the row if it does not already exist.</td> </tr> <tr> <td>u</td> <td>Updates the row if it exists.</td> </tr> <tr> <td>d</td> <td>Deletes the row if it exists.</td> </tr> </tbody> </table>	Value	Action	r	Inserts the row. If the row already exists, it is deleted first.	n	Inserts the row if it does not already exist.	u	Updates the row if it exists.	d	Deletes the row if it exists.
Value	Action											
r	Inserts the row. If the row already exists, it is deleted first.											
n	Inserts the row if it does not already exist.											
u	Updates the row if it exists.											
d	Deletes the row if it exists.											
\$db.origin	String	The schema or origin of the target database.										
\$db.resource	String	The JNDI resource on the target server that contains the database.										
\$db.drivertype	String	The type of the driver. The driver type is the portion of the JDBC URL after <i>jdbc:</i> and before the database specification.										
\$db.parent	String	The name of the parent table.										
\$db.child [ <i>n</i> ]	String	The name of the child table.										
\$row. <i>columnname</i>	String or Number	The name of a specific column in a row of the parent database.										
\$child[ <i>n</i> ]. <i>columnname</i> [ <i>i</i> ]	String or Number	Each child row maps to an index on the child. Each indexed row has a series of column names. This variable specifies the name of the specific column in the specified index row of the child.										
\$row.\$encoding. <i>columnname</i>	String	The encoding of the specified column. Options are empty or <i>base64</i> .										
\$child[ <i>n</i> ].\$encoding. <i>columnname</i>	String	The encoding of the specified child column. Options are empty or <i>base64</i> .										

## Velocity Tool Extensions

Velocity tools are available using the \$tool prefix. For details, see the Velocity tools documentation (<http://jakarta.apache.org/velocity/tools/index.htm>).

Velocity tools may be used in either Templates or in Location Schemes.

The following tools are available:

<b>Tool</b>	<b>Class</b>
\$tools.alternator	org.apache.velocity.tools.generic.AlternatorTool
\$tools.date	org.apache.velocity.tools.generic.DateTool
\$tools.esc	org.apache.velocity.tools.generic.EscapeTool
\$tools.mill	org.apache.velocity.tools.generic.IteratorTool
\$tools.list	org.apache.velocity.tools.generic.ListTool
\$tools.math	org.apache.velocity.tools.generic.MathTool
\$tools.number	org.apache.velocity.tools.generic.NumberTool
\$tools.render	org.apache.velocity.tools.generic.RenderTool
\$tools.sorter	org.apache.velocity.tools.generic.SortTool
\$tools.parser	org.apache.velocity.tools.generic.ValueParser

---

## Assembly Items and Assembly Nodes

When working with Templates and with bindings, it is important to understand the differences between Assembly Items and Assembly Nodes.

An Assembly Item is a container for all data required to assemble a Content Item output, such as the Site and the Template, including the Assembly Node.

The Node is the Content Item itself. The node consists of Content Item data as a set of node properties. A child item edited in a Detail Editor is represented in the Content Item node as a child node with its own set of properties.



## APPENDIX II

# Accessing Object Properties

When implementing Rhythmyx, you may sometimes need to know an object's ID or some other property of the object. You can view and access these properties on the Properties view in the Rhythmyx Workbench. The Properties view is located in the lower left corner of the Workbench, under the Navigation views, with the Snippet Drawer:

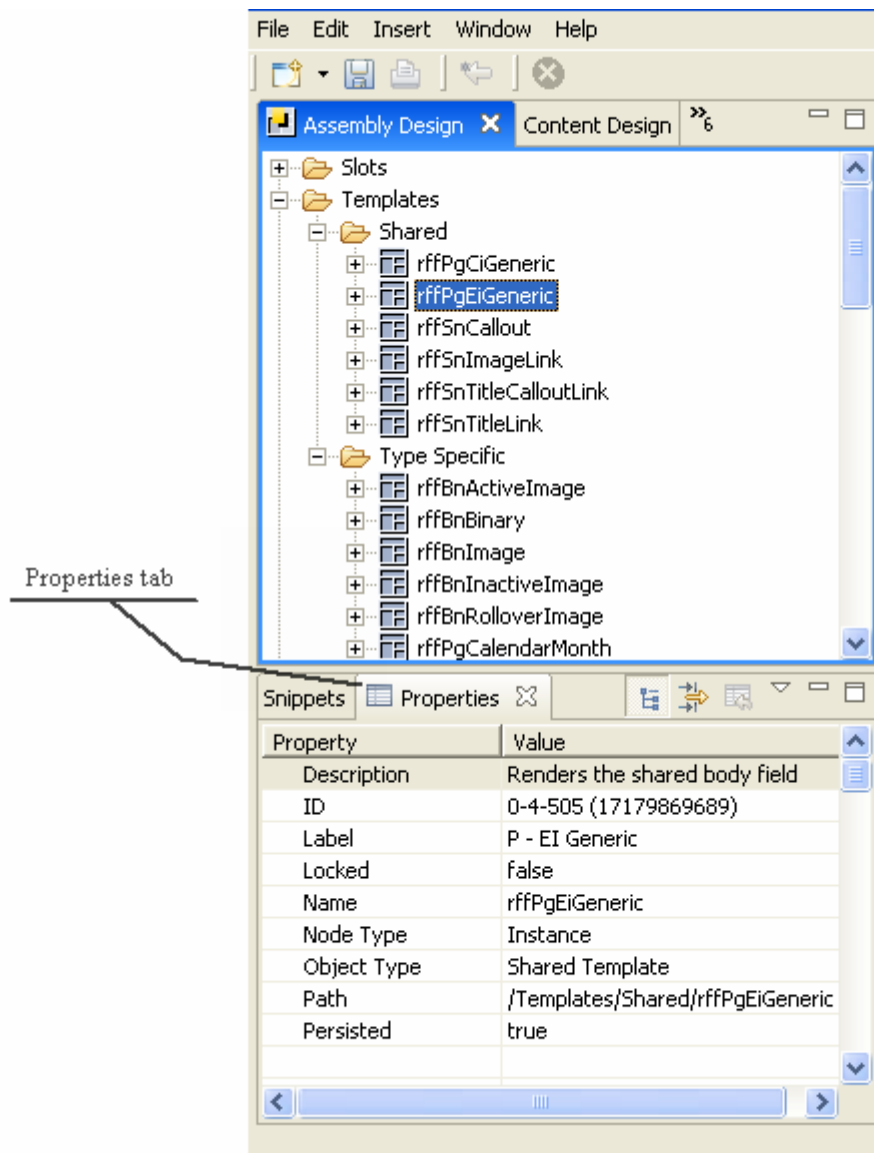


Figure 315: Properties tab

You can copy these properties to the clipboard for use elsewhere in your implementation. To copy a property, select the property you want to copy, right-click, and from the popup menu choose *Copy*. This action copies the entire row. For example, if you copied the ID property in the graphic above, when you pasted the value, the result would be:

```
ID 0-4-505 (17179869689)
```

You should remove extraneous data from the pasted value (for example, *ID* would not be necessary).

The most common property to use is the ID property. In Rhythmyx, IDs are Generic Universal IDs (GUIDs). A Rhythmyx GUID consists of three parts separated by hyphens:

- host ID  
The host ID is a randomly-generated value that identifies the host machine on which the object was created. Each machine on which Rhythmyx is installed has a unique host ID. In the example above, the host ID is *0*.
- object type ID  
The object type ID specifies the type of object. In the example above, the object ID (*4*) indicates that the object is a Template. Object Type ID values are defined by Percussion Software.
- object ID  
The object ID is a unique identifier of the specific object of that type on that host. In the example above, the object ID is *505*.

The numeric value in parentheses following the GUID is a binary representation of the GUID. This value is usually also extraneous data and can be removed.

When using a GUID, you may use either the complete GUID (0-4-505 in this example) or the object ID (505 in this example). When a ID is used, Rhythmyx documentation will specify whether to use the GUID or the object ID. In no case should you use the binary representation of the GUID.



## APPENDIX III

# Naming Conventions

Percussion Software has defined two sets of naming conventions. The first set of naming conventions applies to files and XML applications, the second to design objects (such as Content Types, Templates, Slots, and so forth).

---

## File and Application Naming Conventions

The naming conventions for files and XML applications are mandatory. Percussion Software uses these naming conventions to determine which files and applications will be upgraded when Rhythmyx is upgraded and which are eligible to be customized by customers during implementations. Failure to observe these naming conventions may result in loss of customization when the system is upgraded, failure to upgrade, or failure of the upgraded system.

Any application or file with the prefix "sys" is a Rhythmyx system file or application. These files and applications may be modified during upgrades. **DO NOT** modify or customize these files and applications. Modification or customization of these files and applications may cause the system to fail. Any modifications or customizations to a file or application with the prefix "sys" will be lost on upgrade, and may cause the upgrade to fail.

Files and applications with the prefix "rx" are eligible for customization and modification. These files and applications will not be modified during upgrade and changes to them will not be lost.

In many cases, two files or applications may exist with nearly the same name, differing only in the prefix (one with the prefix "sys", the other with the prefix "rx"). For example, two Velocity macro files are provided with Rhythmyx, `sys_assembly.vm` and `rx_assembly.vm`. The macro file `sys_assembly.vm` is the system file and may be changed on upgrade as Percussion Software changes existing macros and adds new ones. The macro file `rx_assembly.vm` is provided for you to define your own Velocity macros if needed. This file will not be changed by Percussion Software.

---

**REMEMBER:** You can modify or customize files and applications with the prefix "rx". **DO NOT** modify files and applications with the prefix "sys".

---

---

## Design Object Naming Conventions

These naming conventions are designed for the Content Types, Templates, Workflows, and other design objects that constitute the CMS implementation. Use of these naming conventions is not required, but it is strongly recommended that some form of consistent naming be used for the design objects in your implementation.

Names should consist of alphanumeric characters. Name should not include spaces, which are converted to underscores for JSR-170 queries. Underscores may be used instead, but this practice is discouraged. Recommended practice is to compress names. Use of other non-alphanumeric characters is not recommended. Object-specific names (not including various prefixes) should use Title Case. Names should be common words rather than acronyms to ensure that they are easily recognizable.

In many cases, a design object has a label as well as a name. Labels should be easily read and understood by business users of the system. Spaces and other characters are allowed in labels; spaces in particular are encouraged to make them easily readable. Labels need not be unique across the system. In fact, it is common for design objects with different names to have the same label.

Names must be unique throughout the system, but labels need not be unique.

### Project Prefix

Each project should have a three-letter prefix, which is applied to most design objects in the implementation. A project may consist of a single Site, or it may be comprised of a group of Sites that share Content Types and Templates and possibly Content Items as well. The project prefix should always be lower-case and is applied to design objects that require unique names across the system.

Note that the following strings are reserved and should not be used for as a project prefix:

- gbl
- nav
- psx
- rx
- rxs
- sys

For example, the prefix used in the FastForward implementation is "rff", for "Rhythmyx FastForward".

### Content Types

Content Types should use the project prefix and must be unique across the system. The names should be descriptive, for example:

- rffContact
- rffEvent

rffGeneric

Automated Index Content Types should include the string "Auto" in the name after the project prefix:

rffAutoIndex

Content Types also have a label. In most cases, the label can be the name without the project prefix:

Auto Index

Contact

Event

Generic

## Shared Field Groups

Shared Field files and Shared Field Sets must have unique names across the system and should use the project prefix. Shared Field Sets are not visible to business users and do not have a label.

## Fields

Fields should have descriptive names that make sense to business users, and do not use the project prefix. Field names should always be lower-case. Fields also have a label. The word "field" should not be included in the name or label of a field.

Examples:

city

location

postalCode

## Templates

Template names must be unique across the system and should use the project prefix. The following template is recommended:

<prefix><type><ContentType><Name>

where

prefix is the project prefix

type is one of the following values:

Abbreviation	Type
Bn	Binary
Ds	Dispatch
Db	Database Publishing
Gt	Global Template
Pg	Page
Sn	Snippet

ContentType specifies the Content Type with which the Template is associated if the Template is Type-specific.

Name is the Template name.

Templates also have a label.

## Slots

Slot names use the project prefix and must be unique across the system. Slots also have a label. The word "slot" must not appear in the name.

Slots should follow one of the following five conventions:

- "List" Slot - used for a list of Content Items of a specific Content Type  
<prefix><ContentType>List

For example, rffEventList

- "Used-on" Slot - Slot only appears on a specific page or group of pages  
<prefix><PageName>

For example, rffEventPage

- "Business Name" Slot - used for Slots that have a business name users will recognize  
<prefix><BusinessName>

For example, rffSidebar

- "Singleton Link" Slot - Used for image links and other special functions where a Slot will contain a single link to a specific Content Item  
<prefix><Name>Link

For example, rffImageAndTitleLink

- "Singleton Image" Slot - used for images where a single Content Item will be expected.  
<prefix><name>Image

For example, rffNavImage

## Communities

Communities do not have a label. The name is visible to business users, so it should be meaningful to business users and should not use the project prefix. The word "community" should not be used in the name.

Examples:

InternetContent

ExtranetAdmin

## Workflows

Workflows should have simple, descriptive names, which do not include the project prefix. The Word "workflow" should be included in the name.

Workflow States and Transitions should also have simple, descriptive names. Note that States and Transitions do not have labels. The names are directly visible to business users.

## Sites

Site should have simple, descriptive names without the project prefix. Site names must be unique. Sites do not have a label, so the name should be the name used by business users to refer to the Site. It is recommended that Site names not include spaces. The name should not include the word "site".

## Editions and Content Lists

Editions should have unique names that include the project prefix. The names should be organized by Site. Use of camelCase without spaces or underscores is recommended. Editions have a description, which is similar to the label of other design objects and should be a name friendly to business users.

Content Lists should follow the same conventions, but are more likely to require a specific name because there are typically more of them. If an Edition includes only one Content List, the Content List should usually have the same name as the Edition.

The recommended format of the name is:

```
<prefix><Site><Type><Name>
```

where

prefix is the project prefix

Site is the name of the Site to which the Edition publishes.

Type is one of the following:

Full

Incremental

Unpublish

Name is an optional additional name if necessary to distinguish multiple Editions of the same type for a specific Site.

## Context Variables

Context variables should have common sense names that do not use the project prefix. Use of TitleCase without spaces or underscores is recommended.

Context variables can be reused on multiple Sites even on multiple projects if they share Templates.

## **Publishers**

Publishers should have common sense names that do not include a project prefix. The names should use Title Case and should not include spaces or underscores.





## APPENDIX IV

# FastForward Implementation Plan

This appendix contains the complete FastForward Implementation Plan as installed with Rhythmyx. Note that in a production Rhythmyx CMS, the default implementation for some of these design objects may change. Design objects whose implementation is illustrated in the Rhythmyx Implementation Guide are implemented according to this plan unless otherwise specified in the text.

## Shared Field Sets

FastForward includes three shared Field Sets.

- **shared**  
This shared Field Set contains generic fields used by multiple Content Types.
- **sharedimage**  
This shared Field Set contains fields shared by Content Types used to managed image files.
- **sharedbinary**  
This shared Field Set contains field shared by Content Types used to managed binary files.

All of these shared Field Sets are defined in the `rxs_ct_shared` Shared Field Editor.

### shared Field Set

Name	Label	Hidden	Description	Control	Data Type/ Storage Size
displaytitle	Label		The title shown to users.	sys_EditBox	text/512
body	Body		The main body of content. Since the <code>sys_EditLive</code> control is used, the body is stored in rich text format and may include inline links and images.	sys_EditLive	text/max
filename	File name	yes	The file name of the item when it is published.	sys_EditBox	text/512
keywords	Keywords		Search terms that are not part of the item's content and are inserted into markup tags.	sys_TextArea	text/1024
callout	Callout		A synopsis of the body content.	sys_EditLive	text/max
description	Description		Search phrases that are not part of the item's content and are inserted into markup tags.	sys_TextArea	text/1024
webdavowner	WebDAV Owner	yes	Stores the user with a lock on the Content Item when content is uploaded through WebDAV.	sys_TextArea	text/255

## sharedimage Field Set

Name	Label	Hidden	Description	Control	Data Type/ StorageSize
img1	Image		Field that uploads the image.	sys_file	binary/max
img1_filename	Image Filename		File name of the uploaded image. sys_FileInfo extracts this data for system processing or user interface display.	sys_EditBox	text/10
img1_size	Image File Size	yes	File size of the uploaded image. sys_FileInfo extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img1_type	Image Mime Type		MIME type of the uploaded image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_EditBox	text/256
img1_ext	Image Extension	yes	Extension of the uploaded image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_EditBox	text/10
img1_height	Image Height		Height of the uploaded image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img1_width	Image Width		Width of the uploaded image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img_alt	Image Alt Text		Alternate text shown on screen if image does not render.	sys_EditBox	text/512
img2	Mini	yes	Field that uploads thumbnail of image.	sys_file	binary/max

img2_filename	Mini Filename	yes	File name of the uploaded thumbnail image. sys_FileInfo extracts this data for system processing or user interface display.	sys_Editbox	text/512
img2_size	Mini File Size	yes	File size of the uploaded thumbnail image. sys_FileInfo extracts this data for system processing or user interface display.	sys_Editbox	integer/none
img2_type	Mini Mime Type	yes	MIME type of the uploaded thumbnail image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_Editbox	text/256
img2_ext	Mini Extension	yes	Extension of the uploaded thumbnail image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_Editbox	text/10
img2_height	Mini Height	yes	Height of the uploaded thumbnail image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img2_width	Mini Width	yes	Width of the uploaded thumbnail image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none

## Shared Field Sets

### Shared

Name	Label	Rules	Description	Control	Data Type/ Storage Size
displaytitle	Title	Required	The title shown to users.	sys_EditBox	text/512

Name	Label	Rules	Description	Control	Data Type/ Storage Size
body	Body		The main body of content. Since the sys_EditLive control is used, the body is stored in rich text format and may include inline links and images.	sys_EditLive	text/max
filename	File name	Hidden	The file name of the item when it is published.	sys_EditBox	text/512
keywords	Keywords		Search terms that are not part of the item's content and are inserted into markup tags.	sys_TextArea	text/1024
callout	Callout		A synopsis of the body content.	sys_EditLive	text/max
description	Description		Search phrases that are not part of the item's content and are inserted into markup tags.	sys_TextArea	text/1024
webdavowner	WebDAV Owner	Hidden	Stores the user with a lock on the Content Item when content is uploaded through WebDAV.	sys_TextArea	text/255

## SharedImage

Name	Label	Rules	Description	Control	Data Type/ Storage Size
img1	Image	Required	Field that uploads the image.	sys_file	binary/max
img1_filename	Image Filename	Required	File name of the uploaded image. sys_FileInfo extracts this data for system processing or user interface display.	sys_EditBox	text/10
img1_size	Image File Size	Hidden	File size of the uploaded image. sys_FileInfo extracts this data for system processing or user interface display.	sys_EditBox	integer/none

Name	Label	Rules	Description	Control	Data Type/ Storage Size
img1_type	Image Mime Type	Required	MIME type of the uploaded image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_EditBox	text/256
img1_ext	Image Extension	Hidden, Required	Extension of the uploaded image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_EditBox	text/10
img1_height	Image Height		Height of the uploaded image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img1_width	Image Width		Width of the uploaded image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img_alt	Image Alt Text		Alternate text shown on screen if image does not render.	sys_EditBox	text/512
img2	Mini	Hidden	Field that uploads thumbnail of image.	sys_file	binary/max
img2_filename	Mini Filename	Hidden	File name of the uploaded thumbnail image. sys_FileInfo extracts this data for system processing or user interface display.	sys_Editbox	text/512
img2_size	Mini File Size	Hidden	File size of the uploaded thumbnail image. sys_FileInfo extracts this data for system processing or user interface display.	sys_Editbox	integer/none
img2_type	Mini Mime Type	Hidden	MIME type of the uploaded thumbnail image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_Editbox	text/256

Name	Label	Rules	Description	Control	Data Type/ Storage Size
img2_ext	Mini Extension	Hidden	Extension of the uploaded thumbnail image. sys_FileInfo extracts this data which is required to display the file in a browser.	sys_Editbox	text/10
img2_height	Mini Height	Hidden	Height of the uploaded thumbnail image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none
img2_width	Mini Width	Hidden	Width of the uploaded thumbnail image. The sys_imageInfoExtractor pre-exit extracts this data for system processing or user interface display.	sys_EditBox	integer/none

## sharedBinary Field Set Specification

Name	Label	Hidden	Description	Control	Data Type/ StorageSize
item_file_attachment	File:		Field that uploads the binary file.	sys_File	binary/max
item_file_attachment_filename	Binary Filename:		Field that stores the name of the binary file.	sys_EditBox	text/512
item_file_attachment_size	File Size:		Field that stores the size of the binary file.	sys_EditBox	integer
item_file_attachment_type	File Type:		Field that stores the MIMEtype of the binary field.	sys_EditBox	text/256
item_file_attachment_ext	File Extension:		Field that stores the extension of the binary file.	sys_EditBox	text/50

## Content Types

This section describes the Development Plan for the Content Types included in FastForward. The Development Plan for each Content Type includes the following information:

- A brief description of the Content Type
- The Content Type fields
- The Templates associated with the Content Type
- The Workflows associated with the Content Type
- The Communities to which the Content Type is visible

### rffAutoIndex Content Type Specification

Automated lists of Content Items.

#### Fieldsr

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
local	query	Query:	sys_DropDownSingle	required	text	2100
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
system	sys_hibernateVersion		sys_HiddenInput	required	text	255



## Templates

- rffSnAutoBulletList
- rffSnAutoList

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffBrief Content Type Specification

A text blurb used in conjunction with other Content Items. Not a stand-alone page.

## Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
local	placeholder	Placeholder	sys_HiddenInput	required	text	50
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffSnCallout

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

# rffCalendar Content Type Specification

An automated index of Event Content Items.

## Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description:	sys_TextArea	optional	text	1024
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
shared	shared/body	Body:	sys_TextArea	optional	text	max
local	calendar_start	Calendar Date:	sys_CalendarSimple	required	datetime	none
local	query	Query:	sys_DropDownSingle	required	text	255
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_EditBox	optional	text	512
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50

system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffPgCalendarMonth
- rffSnCallout
- rffSnTitleCalloutLink
- rffSnTitleLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

# rffContacts Content Type Specification

Personal contact information used in conjunction with the Press Release Content Type.

## Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
local	firstname	First Name:	sys_EditBox	required	text	100
local	middlename	Middle Name:	sys_EditBox	optional	text	100
local	lastname	Last Name:	sys_EditBox	required	text	100
local	dept	Dept:	sys_EditBox	optional	text	255
local	address1	Address 1:	sys_EditBox	optional	text	200

Source	Name	Label	Control Name	Occur	Data Type	Format
local	address2	Address 2:	sys_EditBox	optional	text	200
local	city	City:	sys_EditBox	optional	text	200
local	state	State	sys_EditBox	optional	text	100
local	zipcode	ZIP Code:	sys_EditBox	optional	text	10
local	country	Country:	sys_EditBox	optional	text	100
local	phone	Phone:	sys_EditBox	optional	text	50
local	cellphone	Cell Phone	sys_EditBox	optional	text	50
local	fax	Fax:	sys_EditBox	optional	text	50
local	email	Email:	sys_EditBox	optional	text	200
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
shared	shared/webdavowner	WebDAV Owner:	sys_HiddenInput	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffSnNameAndAddress
- rffSnNameAndEmail

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffEvent Content Type specification

A full-page Content Type that include description and event date and location information.

### Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Title:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	required	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description	sys_TextArea	optional	text	1024
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
shared	shared/body	Body:	sys_EditLive	optional	text	max
local	event_start	Event Start Date:	sys_CalendarSimple	required	datetime	none
local	event_end	Event End Date:	sys_CalendarSimple	required	datetime	none
local	event_location	Event Location	sys_EditBox	optional	text	1000
local	event_type	Event Type:	sys_DropDownSingle	optional	text	255
shared	shared/filename	File Name:	sys_EditBox	optional	text	512
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional		none
shared	shared/webdavowner	WebDAV Owner	sys_TextArea	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput			none

## Templates

- rffPgCIEvent
- rffPgEIEvent
- rffSnCallout
- rffSnDateRange
- rffSnTitleLink
- rffSnTitleWithDate

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffExternalLink Content Type Specification

A link to a non-managed page or external site.

### Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
local	url	URL:	sys_TextArea	required	text	2048
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
shared	shared/webdavowner	WebDAV Owner:	sys_textArea	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffSnLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffFile Content Type Specification

A single downloadable binary with appropriate metadata (such as MIME type, size, and so forth).

## Fields

Source	Name	Hidden	Label	Control Name	Occur	Data Type	Format
system	sys_title		System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle		Title:	sys_EditBox	required	text	512
system	sys_contentstartdate		Start Date:	sys_CalendarSimple	required	datetime	none
system	sys_contentexpirydate		Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate		Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	sharedbinary/item_file_attachment		File:	sys_file	required	binary	max
shared	sharedbinary/item_file_attachment_filename	Yes	Binary Filename:	sys_EditBox	required	text	512
shared	sharedbinary/item_file_attachment_type	Yes	File Type:	sys_EditBox	required	text	256
local	file_category		File Category:	sys_DropDownSingle	optional	text	50
local	filename		File Name:	sys_EditBox	required	text	512
shared	sharedbinary/item_file_attachment_ext	yes	File Extension:	sys_EditBox	required	text	50
system	sys_suffix	yes	Suffix:	sys_EditBox	required	text	50
shared	sharedbinary/item_file_attachment_size	yes	File Size:	sys_EditBox	required	integer	
system	sys_communityid		Community:	sys_DropDownSingle	required	integer	none

Source	Name	Hidden	Label	Control Name	Occur	Data Type	Format
system	sys_workflowid		Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang		Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview			sys_HiddenInput	optional		none
shared	shared/webdavowner	yes	WebDAV Owner:	sys_textArea	optional	text	255
system	sys_hibernateVersion			sys_HiddenInput			none

## Templates

- rffBnBinary
- rffSnTitleLink
- rffSnTitleAndIcon

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffGeneric Content Type

A full-page Content Type with built-in navigation and banners and a single Body field. This Content Type forms the basis for all Content that takes up a full page.

## Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none



Source	Name	Label	Control Name	Occur	Data Type	Format
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description:	sys_TextArea	optional	text	1024
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
shared	shared/body	Body:	sys_TextArea	optional	text	max
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_EditBox	optional	text	512
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
local	placeholder	Placeholder:	sys_HiddenInput	optional	text	1
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffPgCIGeneric
- rffPgEIGeneric
- rffSnCallout
- rffSnImageLink
- rffSnTitleCalloutAndMoreLink
- rffSnTitleCalloutLink
- rffSnTitleLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffGenericWord Content Type Specification

A full-page Content Type with built-in navigation and banners and a single Body field, edited using Microsoft Word.

### Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description:	sys_TextArea	optional	text	1024
local	PageAuthor	Page Author:	sys_HiddenInput	optional	text	100
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
local	bodysource	Body:	sys_FileWord	optional	binary	max
local	bodysource_filename	Filename	sys_EditBox	optional	text	50
shared	shared/body	Body:	sys_TextArea	optional	text	max
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_EditBox	optional	text	512
local	bodysource_encoding	Body Source Encoding:	sys_HiddenInput	optional	text	50
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffPgCIGeneric
- rffPgCIGenericWord
- rffPgEIGeneric
- rffSnCallout
- rffSnImageLink
- rffSnTitleCalloutAndMoreLink
- rffSnTitleCalloutLink
- rffSnTitleLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffHome

A Site home page

## Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description:	sys_TextArea	optional	text	1024

Source	Name	Label	Control Name	Occur	Data Type	Format
shared	shared/body	Body:	sys_TextArea	optional	text	max
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_EditBox	optional	text	512
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
local	placeholder	Placeholder	sys_HiddenInput	required	text	1
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffPgCIGeneric
- rffPgEIGeneric
- rffSnTitleLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffImage Content Type specification

The Image Content Type is used to upload image files, such as .gif and .jpg files. It should not be used for non-image binary files. The Image Content Type includes fields to upload and manage both full-size images thumbnails.

### Fields

Source	Name	Hidden	Label	Control Name	Occur	Data Type	Format
system	sys_title		System Title:	sys_EditBox	required	text	50
shared	shared/ displaytitle		Title:	sys_EditBox	required	text	512
system	sys_contentstartdate		Start Date:	sys_CalendarSimple	required	datetime	none
system	sys_contentexpirydate		Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate		Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/ description		Description:	sys_TextArea	optional	text	1024
shared	sharedimage/img1		Image:	sys_File	required	binary	max
shared	sharedimage/ img1_filename		Image File Name:	sys_EditBox	optional	text	512
shared	sharedimage/ img1_ext	yes	Image Extension:	sys_EditBox	required	text	10
shared	sharedimage/ img1_type		Image Mime Type:	sys_EditBox	optional	text	256
shared	sharedimage/ img1_height		Image Height:	sys_EditBox	optional	integer	none
shared	sharedimage/ img1_width		Image Width:	sys_EditBox	optional	integer	none
shared	sharedimage/ img1_alt		Image Alt Text:	sys_EditBox	optional	text	512
local	img_category		Image Category:	sys_DropDownSingle	optional	text	50
shared	sharedimage/ img1_size	yes	Image File Size:	sys_EditBox	optional	integer	none
shared	shared/filename	yes	File Name:	sys_EditBox	optional	text	512
system	sys_suffix		Suffix:	sys_EditBox	optional	text	50
shared	sharedimage/img2	yes	Mini:	sys_file	optional	binary	max
shared	sharedimage/ img2_filename	yes	Mini File Name:	sys_EditBox	optional	text	512
shared	sharedimage/ img2_ext	yes	Mini Extension:	sys_EditBox	optional	text	10
shared	sharedimage/ img2_type	yes	Mini Mime Type:	sys_EditBox	optional	text	256

Source	Name	Hidden	Label	Control Name	Occur	Data Type	Format
shared	sharedimage/ img2_height	yes	Mini Height	sys_EditBox	optional	integer	none
shared	sharedimage/ img2_width	yes	Mini Width	sys_EditBox	optional	integer	none
shared	sharedimage/ img2_size	yes	Mini File Size:	sys_EditBox	optional	integer	none
system	sys_communityid		Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid		Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang		Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview			sys_HiddenInput	optional		none
shared	shared/ webdavowner	yes	WebDAV Owner:	sys_textArea	optional	text	255
system	sys_hibernateVersion			sys_HiddenInput			none

## Templates

- rffBnImage
- rffSnFlash
- rffSnImage
- rffSnImageAndTitle

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin

## rffPressRelease

A page Content Type with additional data and metadata required for press release.

### Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	50
shared	shared/displaytitle	Label:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	optional	datetime	none
system	sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	optional	datetime	none
system	sys_reminderdate	Reminder Date:	sys_CalendarSimple	optional	datetime	none
shared	shared/keywords	Keywords:	sys_TextArea	optional	text	1024
shared	shared/description	Description:	sys_TextArea	optional	text	1024
shared	shared/callout	Callout:	sys_EditLive	optional	text	max
shared	shared/body	Body:	sys_TextArea	optional	text	max
local	pr_summary	Summary:	sys_EditLive	optional	text	max
local	includehome	Include on Home Page:	sys_SingleCheckBox	optional	text	50
local	pr_type	Type:	sys_DropDownSingle	required	text	255
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_EditBox	optional	text	512
system	sys_suffix	Suffix:	sys_EditBox	optional	text	50
system	sys_communityid	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	Locale:	sys_DropDownSingle	required	text	50
system	sys_currentview		sys_HiddenInput	optional	text	max
system	sys_hibernateVersion		sys_HiddenInput	required	text	255

## Templates

- rffPgCIPressRelease
- rffPgEIPressRelease
- rffSnCallout
- rffSnDateAndTitleLink
- rffSnHome
- rffSnTitleCalloutLink
- rffSnTitleLink

## Workflows:

- Standard (Default)
- Simple

## "Visible to" Communities:

- Corporate Investments
- Corporate Investments Admin
- Enterprise Investments
- Enterprise Investments Admin



---

## FastForward Workflows

The following sections document the implementation plan for the Simple and Standard Workflows. For each Workflow, the implementation plan lists:

- Each State in the Workflow
- The value of the Publishable field for each State  
This value determines where and when Content Items in the State will be published.
- The Sort Order of the State  
This value determines the order in which the States will be listed in the Workflow.
- A table specifying the Role Assignments for the State. The table consists of the following columns:
  - Roles assigned to the State.  
A Role must be assigned to the State to be able to see, access, and take action on Content Items in that State.
  - The Assignment Type for each Role  
The Assignment Type defines what a user can do with Content Items in a State.
  - Whether Ad Hoc assignment is enabled for each Role.
  - Whether Content Items in the State will be in the Inbox View of users in the Role.
  - Whether users in the Role will receive Notification e-mails sent to Role Recipients of Notification.
- A table defining the Transitions for the State, specifying the State to which the Content Item will be Transitioned; details about the Transition, such as whether the Transition is a Manual Transition or an Aging Transition, whether a specific number of approvals are required or approvals from specific Roles; and whether the Transition includes a Notifications (and if so, the details of the Notification).

## Implementation Plan for Simple Workflow

The Simple Workflow is the most basic recommended Workflow for Rhythmyx. It includes the minimum specific set of States, with associated Transitions and State-assigned Roles. This Workflow does not require any approvals before Content Items are published. The Simple Workflow (or minor variations based on the Simple Workflow) is designed for simple or support Content Types. In FastForward, this Workflow is available to all Communities.

### State: Draft

Publishable Value: Unpublish

Sort Order: 10

Assigned Roles

Roles	Assignment Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Author	Assignee	No	Yes	No
Editor	Reader	No	No	No

Transitions

Name	To State	Details	Notification
Approve	Pending	Manual Transition, 1 Approval	No
Direct to Public	Public	Manual Transition, 1 Approval, Admin only	No

### State: Pending

Publishable Value: Unpublish

Sort Order: 20

Assigned Roles

Roles	Assignment Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Editor	Reader	No	No	No
QA	Reader	No	No	No

## Transitions

Name	To State	Details	Notification
Age to Public	Public	Aging Transition, Triggered by Start Date of Content Item	No
Force to Public	Public	Manual Transition, 1 Approval, Admin only	No

**State: Public**

Publishable Value: Publish

Sort Order: 30

## Assigned Roles

Roles	Assignment Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

## Transitions

Name	To State	Details	Notification
Age to Archive	Archive	Aging Transition, triggered by Expiration Date of the Content Item	Content Archived, to State Role Recipients only, "A content item has transitioned into the archived state and will be removed from your web site."
Expired	Archive	Manual Transition, 1 Approval, Admin or Web Admin only	Content Archived, to State Role Recipients only, "A content item has transitioned into the archived state and will be removed from your web site."
Move to Quick Edit	Quick Edit	Manual Transition, 1 Approval, Admin or Web Admin only	No

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Reminder Transition	Public	Aging Transition, triggered by Reminder Date of Content Item. Reminds administrator that a Content Item is about to expire	Reminder Notification, to State Role Recipients only, "A Content Item has been waiting for your action."

**State: Quick Edit**

Publishable Value: Ignore

Sort Order: 40

Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Return to Public	Public	Manual Transition, 1 Approval, Admin only	No

**State: Archive**

Publishable Value: Archive

Sort Order: 50

Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Republish	Public	Manual Transition, 1 Approval, Admin or Web Admin only	No
Revive	Draft	Manual Transition, 1 Approval, Admin or Web Admin only	No

## Implementation Plan for Standard Workflow

The Standard Workflow is used for most Content Types. It requires one approval before a Content Item can be published.

### State: Draft

Publishable Value: Unpublish

Sort Order: 10

#### Assigned Roles

<b>Roles</b>	<b>Assginement Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Author	Assignee	No	Yes	No
Editor	Reader	No	No	No

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Submit	Review	Manual Transition, 1 Approval	Content Into Review, to State Role Recipients; "A content item has transitioned into the review state."
Direct to Public	Public	Manual Transition, 1 Approval, Admin only	No

**State: Review**

Publishable Value: Unpublish

Sort Order: 20

## Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Author	Reader	No	No	No
QA	Reader	No	No	No
Editor	Assignee	No	Yes	Yes

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Approve	Pending	Manual Transition, 1 Approval	No
Rework	Draft	Manual Transition, 1 Approval	No

**State: Pending**

Publishable Value: Unpublish

Sort Order: 30

## Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Reader	No	No	No

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Age to Public	Public	Aging Transition, Triggered by Publish Date of Content Item	No
Force to Public	Public	Manual Transition, 1 Approval, Admin only	No

**State: Public**

Publishable Value: Publish

Sort Order: 40

**Assigned Roles**

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

**Transitions**

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Age to Archive	Archive	Aging Transition, triggered by Expiration Date of the Content Item	Content Archived, to State Role Recipients only, "A content item has transitioned into the archived state and will be removed from your web site."
Expired	Archive	Manual Transition, 1 Approval, Admin or Web Admin only	Content Archived, to State Role Recipients only, "A content item has transitioned into the archived state and will be removed from your web site."
Move to Quick Edit	Quick Edit	Manual Transition, 1 Approval, Admin or Web Admin only	No
Reminder Transition	Public	Aging Transition, triggered by Reminder Date of Content Item. Reminds administrator that a Content Item is about to expire	Reminder Notification, to State Role Recipients only, "A Content Item has been waiting for your action."

**State: Quick Edit**

Publishable Value: Ignore

Sort Order: 50

## Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Return to Public	Public	Manual Transition, 1 Approval, Admin only	No

**State: Archive**

Publishable Value: Archive

Sort Order: 60

## Assigned Roles

<b>Roles</b>	<b>Assignment Type</b>	<b>Ad Hoc</b>	<b>Show in Inbox</b>	<b>Receive Notifications?</b>
Web Admin	Assignee	No	No	No
Admin	Assignee	No	No	No
RxPublisher	Reader	No	No	No
Author	Reader	No	No	No
Editor	Reader	No	No	No
QA	Assignee	No	No	No

## Transitions

<b>Name</b>	<b>To State</b>	<b>Details</b>	<b>Notification</b>
Republish	Public	Manual Transition, 1 Approval, Admin or Web Admin only	No



Revive

Draft

Manual Transition, 1 Approval,  
Admin or Web Admin only

No

# FastForward Publishing Configurations

This section details the publishing configurations of FastForward.

## FastForward Sites

FastForward includes two Sites: Corporate Investments and Enterprise Investments

SiteName	Corporate Investments	Enterprise Investments
Description	Represents the Corporate Investments web site	Represents the Enterprise Investments web site
Site Address URL	http://127.0.0.1:9662/CI_Home	http://127.0.0.1:9662/EI_Home
Site Home Page URL	http://localhost:9662/Rhythmyx/rxs_Home_cas/page.html?sys_revision=5&sys_siteid=303&sys_authtype=0&sys_contentid=551&sys_variantid=399&sys_folderid=523&sys_context=0	http://localhost:9662/Rhythmyx/rxs_Home_cas/page.html?sys_revision=2&sys_siteid=301&sys_authtype=0&sys_contentid=466&sys_variantid=342&sys_folderid=301&sys_context=0
Publishing Root Location	../CI_Home.war	../EI_Home.war
Publisher	Localhost Publisher Default Port	Localhost Publisher Default Port
Folder Root	//Sites/CorporateInvestments	//Sites/EnterpriseInvestments
Global Template	rffGtCorporateInvestmentsCommon	rffGtEnterpriseInvestmentsCommon

## FastForward Content Lists

FastForward is delivered with the following Content Lists

### rffCiFullBinary

Site Root Full for Binary Content Types - Corporate Investments

URL:

```
/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_assembly_context=301&sys_contentlist=rffCiFullBinary
```

Edition Type: Automatic

Generator: sys\_SearchGenerator

```
Query: select rx:sys_contentid, rx:sys_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where jcr:path like '//Sites/CorporateInvestments%'
```

Template Expander: sys\_SiteTemplateExpander

Site ID: 303

Item Filter: Public

Publication Filter: NA

### **rffCiFullNonBinary**

Site Root Full for Non-Binary Content Types - Corporate Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffCiFullNonBinary

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from  
rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffcategor,rx:rffcontacts,rx:rffevent,rx:rffexternallink,  
rx:rffgenericword,rx:rffgeneric,rx:rffhome,rx:rffpressrelease where jcr:path like  
'//Sites/CorporateInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 303

Item Filter: Public

Publication Filter: NA

### **rffCiIncremental**

Site Root Incremental - Corporate Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffCiIncremental

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from nt:base where jcr:path like  
'//Sites/CorporateInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 303

Item Filter: Public

Publication Filter: Incremental

### **rffCiUnpublish**

Delete any published pages of items in an archive state - Corporate Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffCiUnpublish

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where jcr:path like '//Sites/CorporateInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 303

Item Filter: Unpublish

Publication Filter: NA

### **rffEiFullBinary**

Site Root Full for Binary Content Types - Enterprise Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffEiFullBinary

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where jcr:path like '//Sites/EnterpriseInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 301

Item Filter: Public

Publication Filter: NA

### **rffEiFullNonBinary**

Site Root Full for Binary Content Types - Enterprise Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffEiFullNonBinary

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffcategory,rx:rffcontacts,rx:rffevent,rx:rffexternallink,rx:rffgenericword,rx:rffgeneric,rx:rffhome,rx:rffpressrelease where jcr:path like '//Sites/EnterpriseInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 301

Item Filter: Public

Publication Filter: NA

**rffEiIncremental**

Site Root Incremental - EnterpriseInvestments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffEiIncremental

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from nt:base where jcr:path like '//Sites/EnterpriseInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 303

Item Filter: Public

Publication Filter: Incremental

**rffEiPublishNow**

Site Publish Now - Enterprise Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffEiPublishNow

Edition Type: Automatic

Generator: sys\_SelectedItemsGenerator

Template Expander: sys\_SiteTemplateExpander

Site ID: 301

default\_template: Dispatch

Item Filter: sitefolder

Publication Filter: NA

**rffEiUnpublish**

Delete any published pages of items in an archive state - Enterprise Investments

URL:

/Rhythmyx/contentlist?sys\_deliverytype=filesystem&sys\_assembly\_context=301&sys\_contentlist=rffEiUnpublish

Edition Type: Automatic

Generator: sys\_SearchGenerator

Query: select rx:sys\_contentid, rx:sys\_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where jcr:path like '//Sites/EnterpriseInvestments%'

Template Expander: sys\_SiteTemplateExpander

Site ID: 301

Item Filter: Unpublish

Publication Filter: NA

## FastForward Item Filters

The following Item Filters are shipped with FastForward.

### incremental

Incremental publishing filter

Rules

sys\_incrementalPublish

### preview

Preview Item Filter

Rules

None

Associated Authorization Type

All Content

### public

Public Item Filter

Rules

sys\_filterByPublishableFlag

Parameter	Value
sys_FlagValues	y,i

Associated Authorization Type

All Public Content

### sitefolder

Allows public content that is present on the target site (either the site being published, or the site referenced in the related content information). The content type involved must have a publishable template on the target site.

Inherit From

Public

Rules

sys\_filterBySiteFolder

Associated Authorization Type

Site Folder Content

## **unpublish**

Rules

sys\_filterByPublishableFlag

<b>Parameter</b>	<b>Value</b>
sys_FlagValues	u





## APPENDIX V

# Content Editor System Definition

The following table describes the fields defined in the Content Editor System Definition that are eligible to be included in Content Editors. By default, all of these fields are defined with the following property values:

Treat data as binary: No

Show in Preview: Yes

Allow this field to be searched: Yes

Name	Label	Mandatory	Comments
sys_communityid	Community Id	Yes	Defined when the Content Item is created and never modified afterwards. By default, value is derived from the currently logged Community of the user that creates the Content Item. Hidden by default. If visible, options include all Communities defined in the system.
sys_contentexpirydate	Content expiration date	No	
sys_contentstartdate	Content start date	Yes	Date format is yyyy-MM-dd
sys_currentview	(None)	Yes	Hidden Input.
sys_hibernateVersion	(None)	Yes	Hidden Input. Field only used internally, but must be included on all Content Editors.
sys_lang	Locale ID	Yes	Defined when the Content Item is created and never modified afterwards. By default, value is derived from the currently logged Locale of the user that creates the Content Item. Hidden by default. If visible, options include all Locales defined in the system.
sys_pathname	Path name	No	
sys_pubdate	Publication date	No	
sys_reminderdate	Reminder date	No	

Name	Label	Mandatory	Comments
sys_suffix	Suffix	No	Defaults to ".html". Hidden by default.
sys_title	System title	Yes	This field cannot be empty and must be unique within the folder.
sys_workflowid	Workflow	Yes	Hidden by default.

The next table describes fields defined in the Content Editor System Definition that are not eligible to be included in Content Editors. These fields are used mostly for processing of Content Items or to provide human-readable information for ID fields defined in the system definition. The value of some of these fields is computed at runtime. Those fields are not eligible to be searched, but, like all fields in the system definition, can be included in Display Formats.

Name	Label	Searchable	Comments
sys_assignees	Assignees	No	Computed.
sys_assignmenttype	Assignment type	No	Computed. Valid values include: <ul style="list-style-type: none"> <li>▪ None</li> <li>▪ Reader</li> <li>▪ Assignee</li> <li>▪ Admin</li> </ul>
sys_assignmenttypeid	Assignment type ID	No	Computed.
sys_checkoutstatus	Checkout status	No	Computed.
sys_communityname	Community Name	Yes	
sys_contentcreatedby	Created by	Yes	Defined when the Content Item is created and never modified afterwards
sys_contentcreateddate	Created on	Yes	Defined when the Content Item is created and never modified afterwards.
sys_contentcheckoutusername	Checked out user name	Yes	
sys_contentid	Content id	Yes	Defined when the Content Item is created and never modified afterwards.
sys_contentlastmodifieddate	Last modified date	Yes	
sys_contentlastmodifier	Last modified by	Yes	
sys_contentstateid	Workflow State ID	Yes	
sys_contenttypeid	Content Type	Yes	Defined when the Content Item is created and never modified afterwards.

<b>Name</b>	<b>Label</b>	<b>Searchable</b>	<b>Comments</b>
sys_contenttypename	Content Type Name	Yes	
sys_folderid	Folder Path	Yes	
sys_localename	Locale Name	Yes	
sys_objecttype	Object type	No	Defined when the Content Item is created and never modified afterwards.
sys_publishabletype	Publishable status	No	Computed
sys_relevancy	Rank	Yes	This field is used to provide the relevancy ranking returned by the external search engine. The field value is overwritten by the search engine at the time the search results are processed. If no rank is available, or if the search was performed against the internal engine, the value is left at -1.
sys_siteid	Site	Yes	
sys_statename	Workflow State Name	Yes	
sys_thumbnail	Thumbnail	Yes	
sys_variantid	Variant	Yes	
sys_variantname	Variant Name	Yes	
sys_workflowname	Workflow Name	Yes	



# Index

## #

#children macro • 160  
 #displayfield • 126  
 #field • 126  
 #inner macro • 168  
 #slot macros • 152, 160

## \$

\$rx Functions • 415, 418, 419, 420, 421, 422, 424, 425, 427, 428

## A

Absolute Aging • 41  
 Access Control Lists • 61, 70  
 Accessibility • 403  
 ACL • 70, See Access Control Lists  
 Active Assembly Tutorial • 403  
 Ad Hoc Assignment • 48  
 Administrator • 32  
 Aging • 41, 55, 58  
 Allowed Content • 114, 115  
 Ancestor • 263  
 Ancestor-sibling • 263  
 Approvals (on Transitions) • 56  
 Assembly Binding Variables • 433  
 Automated Index • See Automated Slots  
 Automated Slots • 187, 190, 192, 193  
 Axis • 263

## B

Binary  
   fields • 93  
   files • 140  
   Templates • 120, 140  
   treat data as • 81  
 Bindings • 136, 137, 140, 143, 147, 192, 290  
   Bindings/variables • 290, 409, 410, 414, 431, 432, 433

## C

Cascading Stylsheets • 286, 288  
 Catalogging • 23

Child Field Set • 240, 242  
 Child Table • 160  
 Comments • 54  
 Communities • 29, 33  
 Community Visibility • 33  
 Content Editor • 485, See Content Type  
 Content Finder • 114, 115  
 Content List • 297, 301, 303  
   Database Publishing • 385  
 Content Type • 209, 211, 212, 214, 215, 219, 225, 226, 229, 230, 231, 233, 236, 238, 240, 241, 242, 246, 249, 250, 251, 273  
   child field set • 240, 242  
   Content Type/properties • 233  
   definition • 209  
   including shared and system fields • 219  
   overriding a shared field • 241  
 Context (Database Publishing) • 383  
 Context Variables • 174, 175, 176  
 Contexts • 308, 309, 315  
 Controls • 81, 90, 93  
 CSS • See Cascading Stylsheets

## D

Data bindings • See Bindings  
 Data Type • 81, 93  
 Database Publishing • 359, 361, 364, 389  
   Binding Variables • 431  
   Content List • 385  
   Context • 383  
   Database Publishing/components • 360  
   Database Publishing/Oracle • 371  
   Database Publishing/plugin parameter • 363  
   Database Publishing/procedure • 364  
   Database Publishing/reviewing publications • 390  
   Database Publishing/troubleshooting • 394  
   datasource • 365  
   Edition • 387  
   Site • 373  
   Templates • 120, 368, 375  
   unpublishing • 393  
 Datasource • 365  
 Default Security Provider • 17, 19  
 Default Value • 81, 93  
 Descendant • 263  
 Development Infrastructure • 15  
 Display Formats • 403  
 DOCTYPE • 121

**E**

Editions • 328, 330, 334, 337, 339  
    Database Publishing • 387  
Escalating Notices • 58

**F**

FastForward Implementation Plan • 445, 446,  
    452, 469, 478  
Fields • 75, 81, 93, 99, 100, 103, 219, 231, 241  
    binary • 93  
    Fields/Field and Field Set Editor • 79  
    transformations • 99  
    validation • 99, 103  
    visibility • 100  
Folders • 70  
FROM Clause • 190

**G**

Global Template • 163, 166, 168, 171, 174, 175,  
    176  
    adding Managed Navigation • 171  
    for Site • 65  
    Global Template/ • 174, 175, 176

**H**

Home Page URL • 65  
HTML  
    preparing for use in Templates • 121  
    search properties • 81  
Hypertext Links • 143

**I**

Image Links • 147  
Implementation Guide • 7  
Implementation Roadmap • 8, 10  
Initial State • 39  
Inline scripting • 121  
Internationalization • 403

**J**

JSR-170 • 187, 190

**K**

Keywords • 273

**L**

Label, and internationalization • 81  
Landing Page • 267  
Lifecycle Analysis • 403

LIKE Operator • 190  
Links • 143, 147  
List Control • 90  
Local Content • 168  
Local Fields • 231  
Local Template • 120, See Templates  
Localization • 403  
Location Schemes • 308, 309, 315, 319

**M**

Macros • See Velocity  
Managed Navigation • 68, 114, 120, 261, 263,  
    267, 268, 269, 272, 273, 277, 278, 279, 286,  
    288, 289, 290  
    adding to a Global Template • 171  
    adding to a Site • 68  
    bindings • 290  
    Cascading Stylesheets • 286, 288, 289  
    Managed Navigation/Customizing Look and  
        Feel • 278, 279, 286, 288, 289, 290  
    maintaining • 267, 268, 269, 272, 273  
    Slots • 277  
    Templates • 278, 279, 290

Max Results • 190

Members • 23  
Microsoft Word • 403  
Migration • 254  
Mnemonic • 81, 93

**N**

Naming Conventions • 437, 438, 439, 440, 441,  
    442, 443  
Navigation Sections  
    merging • 272  
    Merging Navigation Sections • 272  
    splitting • 269  
    Splitting Navigation Sections • 269  
NavImage • 261, 263, 268  
Navon • 68, 261, 267, 269, 272  
    landing page • 267  
    merging • 272  
    Navon Properties/Binding Variables • 290  
    splitting • 269  
NavTree • 61, 68, 263, 267  
Notification • 42, 48  
    enabling for a Role • 48

**O**

Object Properties • 435  
Oracle • 371

ORDER BY Clause • 190

## P

Page Template • 120, 149, 152, 160  
 Post-processing • 250, 251  
 Pre-processing • 250, 251  
 Publishing • 294, 296, 339, 340, 343, 344  
   Database Publishing • See Database Publishing  
   FTP • 65, 351, 353, 355, 356, 357  
   Publishing/Local Web Server • 346, 348, 349  
   Publishing/setting up Site • 61, 63, 65  
 Publishing Root Location • 65

## R

Repeating Transitions • 55  
 Rich Text Control • 81  
 Roadmap • 8, 10  
 Roles • 20, 23  
   Ad Hoc assignment • 48  
   adding users to • 23  
   assigning to a State • 39  
   Community • 20, 29  
   Notification • 48  
   Workflow • 20, 35  
 Root • 263  
 runTd.bat/runTd.sh • 368

## S

Scripting • 121  
 Search (field properties) • 81, 93  
 Searches • 403  
 Security Provider  
   Default Security Provider • 17, 19  
 SELECT Clause • 190  
 Self • 263  
 Server Administrator • 14  
 Shared Fields • 75, 77, 78, 79, 81, 93  
   creating Shared Field Sets • 81, 93  
   including in a Content Editor • 219  
   including shared/system • 219  
   overriding • 241  
 Shared Templates • 120  
 Show in Preview • 81, 93  
 Sibling • 263  
 Site Address • 65  
 Sites • 61, 63, 65, 67

adding Managed Navigation • 68  
 adding Subfolders • 67  
 creating Site Root • 63  
 Database Publishing • 373  
   registering • 65  
 Slots • 107, 114, 119, 187, 190, 192, 193, 277  
   Automated • 107, 187, 190, 192, 193  
   controlling Contents • 119  
   Managed Navigation • 107, 277  
   queries • 190, 193  
   Standard • 114, 115, 119  
 Slots/Creating • 114, 115, 119  
 Snippets  
   Complex • 143, 147  
   Snippet Drawer • 126  
   Snippet Template • 122, 123, 126  
 Standard Slots • 114, 115, 119  
 States • 36, 37, 39, 51  
   assigning Roles • 37  
   creating • 36  
   Initial State • 39  
 Static Files • 174, 175, 176  
 Storage size • 81, 90  
 System Fields • 219

## T

Table Definition Builder • 368, 371  
 Templates • 107, 120, 121, 122, 123, 126, 134, 140, 143, 147, 149, 152, 160, 163, 166, 168, 171, 174, 175, 176, 181, 278, 279  
   binary • 140  
   Database Publishing • 368, 375  
   dispatch • 181, 184  
   Global Templates • 163, 166, 168, 171, 174, 175, 176  
   Local • 120  
   Managed Navigation • 278, 279, 290  
   page • 149, 152, 160  
   Templates/debugging • 134  
   Templates/HTML, preparing for use in  
     Templates • 121  
   Templates/Page without Global • 178, 180  
   Templates/Snippet • 122, 123, 126, 143, 147  
   Templates/Troubleshooting • 194, 195, 196, 197, 198, 199, 200, 201, 203, 205, 206, 207  
 Text Extraction • 254  
 Text, treating as binary • 81  
 Transformation • 99, 250  
 Transition • 39, 41, 51, 54, See Aging

- approvals • 56
- comments • 54
- creating • 39, 41
- Treat data as binary • 81
- Troubleshooting • 194
- Tutorial • 403
- Type-specific Template • 120

## U

- Unpublishing • 393
- USERLOGIN • 17, 19
- Users
  - adding to a Role • 23
  - adding to default Security Provider • 17, 19

## V

- Validation • 99, 103, 250
- Variable Parameters • 192, 193
- Variable Selectors • 288, 289
- Variables • 409
- Velocity • 120, 126, 152, 432
  - Velocity/Binding Variables • 432
- Views • 403
- Visibility • 99, 100

## W

- Web Services • 403
- WebDAV • 254, 403
- WebImageFX • 238
- WHERE Clause • 190
- Word • See Microsoft Word
- Workbench • 12
- Workflow • 31, 32, 33, 35, 36, 37, 39, 41, 42, 47, 48, 51, 54, 55, 56, 58, 469, 470, 473, See
  - Transition, See States
  - associating with a Community • 33
  - creating • 32
  - defined • 31
  - Roles • 35
  - Workflow/copying • 47