Rhythmyx

Rhythmyx Implementation Guide

Version 7.0.3

Printed on 5 October, 2011

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 grated. 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.

Licenses and Source Code

Rhythmyx uses Mozilla's JavaScript C API. See ***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.

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.

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

Schema documentation was created using XMLSpy™.

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

Contents

About the Rhythmyx Implementation Guide 11

Rhythmyx Implementation Roadmap 12
Implementation in the Rhythmyx Implementation Roadmap 14

Accessing Rhythmyx Client Interfaces 15

Starting the Rhythmyx Workbench 16
Starting the Rhythmyx Server Administrator 18

Setting Up the Development Infrastructure 19

Creating User Logins 21
Adding Users to the USERLOGIN Table Manually 21
Adding Users to the USERLOGIN Table Using a Script 23
Roles 24
Creating a Role 24
Adding Users to a Role 28
Communities 33
Creating a New Community 33
Workflows 35
About the Workflow Administrator 35
Implementing the Simple Workflow 36
Implementing the Standard Workflow 53

Setting up the Publishing Site and Basic Navigation 67

Creating the Site Root Folder 69
Registering the Publishing Site with Rhythmyx 71
Creating Site Subfolders 73
Managed Navigation for the Site 74
Adding a NavTree to the Site Hierarchy 74
Defining Access to Folders using Access Control Lists (ACLs) 76

Creating Shared Fields 81

shared Field Set 83
sharedimage Field Set 84
The Rhythmyx Workbench Field and Field Set Editor 85
Creating Shared Field Sets and Configuring Fields 87
Implementing the "shared" Field Set 87
Implementing a List Control 95
Implementing the sharedimage Field Set 98
Field Visibility, Validation, and Transform Rules 104
Adding a Field Visibility Rule 105
Adding a Field Validation Rule 108

Creating Slots and Templates 113

Creating Slots....120
Creating a Standard Slot....121
Creating Templates....125
Preparing HTML for Use in Templates....126
Implementing Snippet Templates....127
Bindings....141
Implementing a Binary Template....144
Complex Snippets....147
Implementing Page Templates....153
Implementing Global Templates....169
Implementing a Page Template Without a Global Template....183
Dispatch Templates....187
Creating an Automated Slot....192
Creating a Simple Automated Slot....192
Automated Slots with Variable Parameters....198
Troubleshooting Templates....200
Property Not Found Error....200
Macro Rendered as Plain Text....201
Invalid Argument....202
Problem Assembling Output: Value is Badly Formed....203
Parameter Not Defined....204
Lexical Error....205
Velocity Code in Output....206
Illegal Argument Exception: Target Template May Not be Null....207
Problems Assembling Binary Outputs....208
Could Not Find Method <Name> for Object [null]....209
Java.lang.RuntimeException: Could not find method <name> for object <bindingfunction>....211
Problem Parsing Expression....212
Java.lang.NullPointerException....213

Creating Content Types 215

Summary of Content Types....217
Generic Content Type....217
Image Content Type....219
Event Content Type....220
Basic Content Type Creation....222
Creating the Generic Content Type Object....223
Including Shared and System Fields....227
Specifying an Icon for the Generic Content Type....229
Making the Generic Content Type Visible to Another Community....231
The Generic Content Editor....235
Viewing Generic Content Items....236
Image Content Type Creation....239
Creating the Image Content Type Object....240
Including a Local Field....241
Entering Content Editor Properties....243
The Image Content Editor....246
Viewing Image Content Items....248
WebImageFX....249

Creating a Content Type with a Child Field Set 250
Overriding a Shared Field....................251
Adding a Child Field Set 252
Populating a Field from an External Lookup....................256
The Event Content Editor 267
Viewing Event Content Items....................270
Item Transformation, Validation, and Pre- and Post-Processing 271
Adding Pre-processing Extensions 272
Implementing Text Extraction 275
Creating a Text Extraction Content Type 275

Managed Navigation 279

How Managed Navigation Works 281
Maintaining Managed Navigation Content Items 285
Navigation Communities....................285
Assigning a Landing Page to a Navon....................285
Creating a NavImage 286
Splitting Navigation Sections 287
Merging Navigation Items....................290
Reordering a Submenu 291
Creating and Using Keywords....................291
Managed Navigation Slot 295
Customizing Navigation Look and Feel 296
Creating Managed Navigation Templates 296
Customizing Navigation CSS....................303
Navon Properties 307

Configuring Publishing 309

Publishing Specifications....................311
Content Explorer's Publishing DesignTab 313
Defining Content Lists....................314
Defining the Full Binary Content List....................314
Defining the Full Non-Binary Content List....................318
Defining the Incremental Content List 319
Defining Contexts and Location Schemes 321
Creating the Publish Context and its Generic Location Scheme 321
Common Errors in JEXL Expressions....................327
Creating Additional Contexts 328
Creating Additional Location Schemes 328
Creating Editions 330
Full Publish Edition....................330
Incremental Edition Edition....................334
Testing your Content Lists 337
Testing Publishing of your Editions 338
Publishing the Full Publish Edition 338
Publishing the Incremental Edition 340
Implementing Demand Publishing (Publish Now) 342
Setting Up the Corporate Investments Site 344
Copying a Site in the Content Tab....................344
Copying a Site Registration....................349
Copying Editions and Content Lists 351

Setting Up Publishing to a Local Web Server 353
Implementing FTP Delivery 356
Setting up Standard FTP Delivery 356
Setting Up Secure FTP (SFTP) Delivery 359

Database Publishing in Rhythmyx 363

Database Publishing Implementation Process 364
Database Publishing Specifications 365
Creating a JNDI Datasource Configuration and a Database Connection 368
Creating a Database Publishing Template 373
Defining Bindings to Publish Content Item Data 378
Defining Bindings to Publish Local Child Data 382
Defining Bindings to Publish Child Data from an External Repository 385
Testing and Debugging a Database Publishing Template 389
Defining the Publishing Configurations for Database Publishing 392

Specialized Implementations 395

Next Steps 397

Appendices 399

Setting Up SSL 401

Enabling SSL on the Rhythmyx Server on Windows 402
Enabling SSL on the Rhythmyx Server on Solaris and Linux 403
Implementing a Self-signed Certificate 404

Binding Variables 407

System Variables 408
System Functions 412
$rx.asmhelper 413
$rx.codec 417
$rx.cond 418
$rx.db 419
$rx.doc 419
$rx.ext 420
$rx.guid 421
$rx.i18n 421
$rx.keyword 422
$rx.link 424
$rx.location 425
$rx.nav 427
$rx.pagination 428
$rx.session 430
$rx.string 430

Navon Properties 432
Database Publishing Variables 433
Velocity Tool Extensions..434
Assembly Items and Assembly Nodes..435

Accessing Object Properties 437

Naming Conventions 439

File and Application Naming Conventions..440
Design Object Naming Conventions 441
Project Prefix..441
Content Types..441
Templates 442
Slots..443
Communities 443
Workflows..444
Sites 444
Editions and Content Lists..444
Context Variables 444
Publishers 445

FastForward Implementation Plan 447

Shared Field Sets 448
shared Field Set 448
sharedimage Field Set..449
sharedBinary Field Set Specification..450
Content Types..452
rffAutoIndex Content Type Specification 452
rffBrief Content Type Specification 453
rffCalendar Content Type Specification 454
rffContacts Content Type Specification 456
rffEvent Content Type specification..457
rffExternalLink Content Type Specification 459
rffFile Content Type Specification 460
rffGeneric Content Type..462
rffGenericWord Content Type Specification..463
rffHome 465
rffImage Content Type specification 467
rffPressRelease 469
FastForward Workflows 471
Implementation Plan for Simple Workflow..472
Implementation Plan for Standard Workflow..475
FastForward Publishing Configurations 479
FastForward Sites 479
FastForward Item Filters 479
FastForward Content Lists..480

Content Editor System Definition 483

Index 487

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 19) 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 67) 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 (see page 81) 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 113) shows the implementation of both Standard and Automated Slots, Global Templates, and a variety of Local Templates.

Creating Content Types (see page 215) 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*** (see page 81)).

Managed Navigation (on page 279) describes how to implement the infrastructure required to publish Managed Navigation successfully, with a special emphasis on Managed Navigation Templates.

Configuring Publishing (see page 309) 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 24), ***Communities*** (see page 33), ***Workflows*** (on page 35), and ***users*** (see "Creating User Logins" on page 21) sketched out during modeling 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 67). The Site Folder structure may not be established during modeling 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 page 81).
5. ***Create your Slots*** (see "Creating Slots" on page 120).
6. ***Create your Global template*** (see "Implementing Global Templates" on page 169).
7. ***Create your Content Types*** (see page 215), either by modifying existing FastForward Content Types or by creating new ones.
8. ***Create your local and shared Templates*** (see page 125).
9. ***Completing the set up of your Site Folders and navigation hierarchy*** (see "Managed Navigation" on page 279).
10. Modify the configuration of your ***Roles*** (see page 24), ***Communities*** (see page 33), ***Workflows*** (on page 35), and ***users*** (see "Creating User Logins" on page 21) 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.

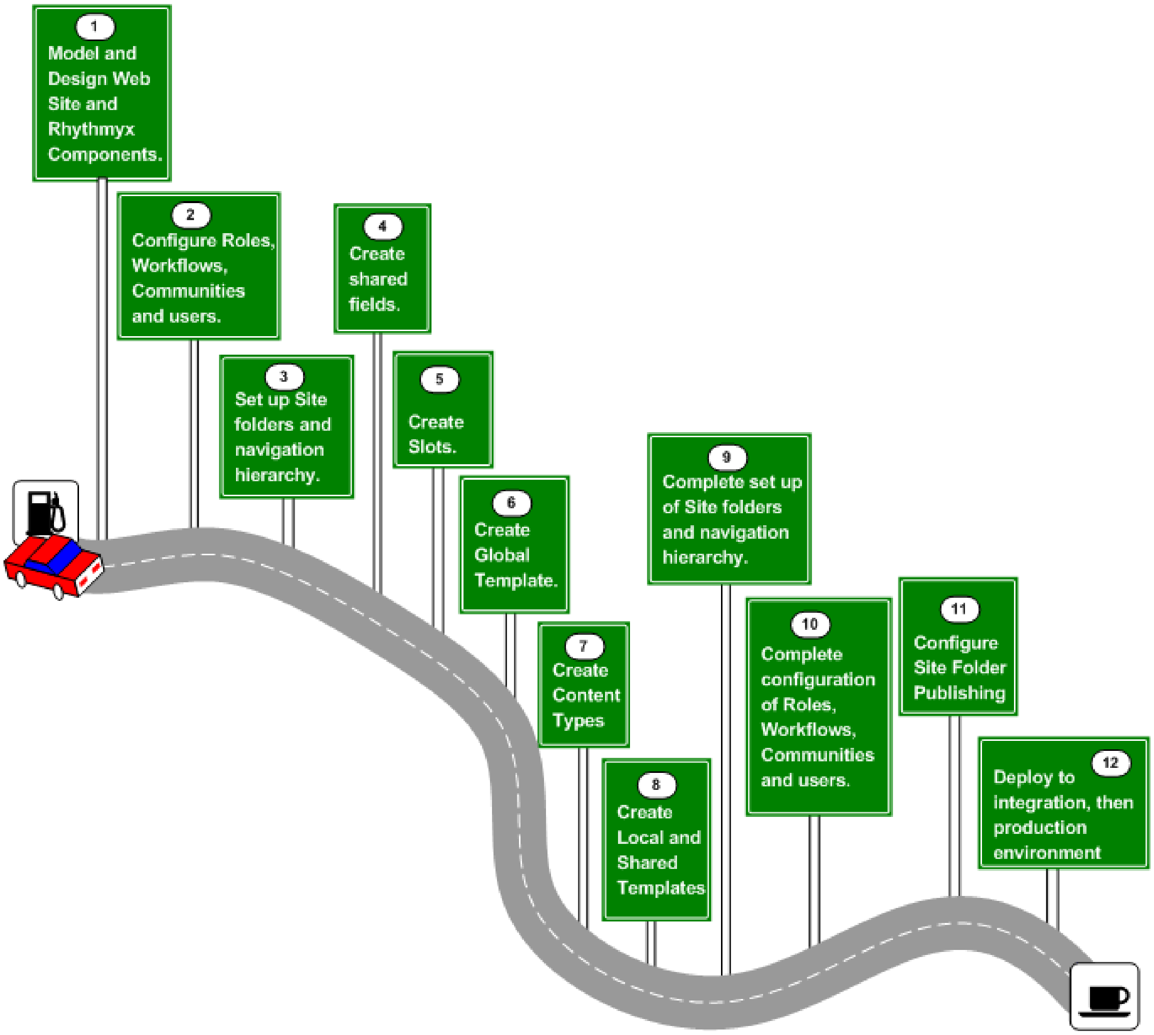

1
Model and Design Web Site and Rhythmyx Components.
2
Configure Roles, Workflows, Communities and users.
3
Set up Site folders and navigation hierarchy.
4
Create shared fields.
5
Create Slots.
6
Create Global Template.
7
Create Content Types
8
Create Local and Shared Templates
9
Complete set up of Site folders and navigation hierarchy.
10
Complete configuration of Roles, Workflows, Communities and users.
11
Configure Site Folder Publishing
12
Deploy to integration, then production environment

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

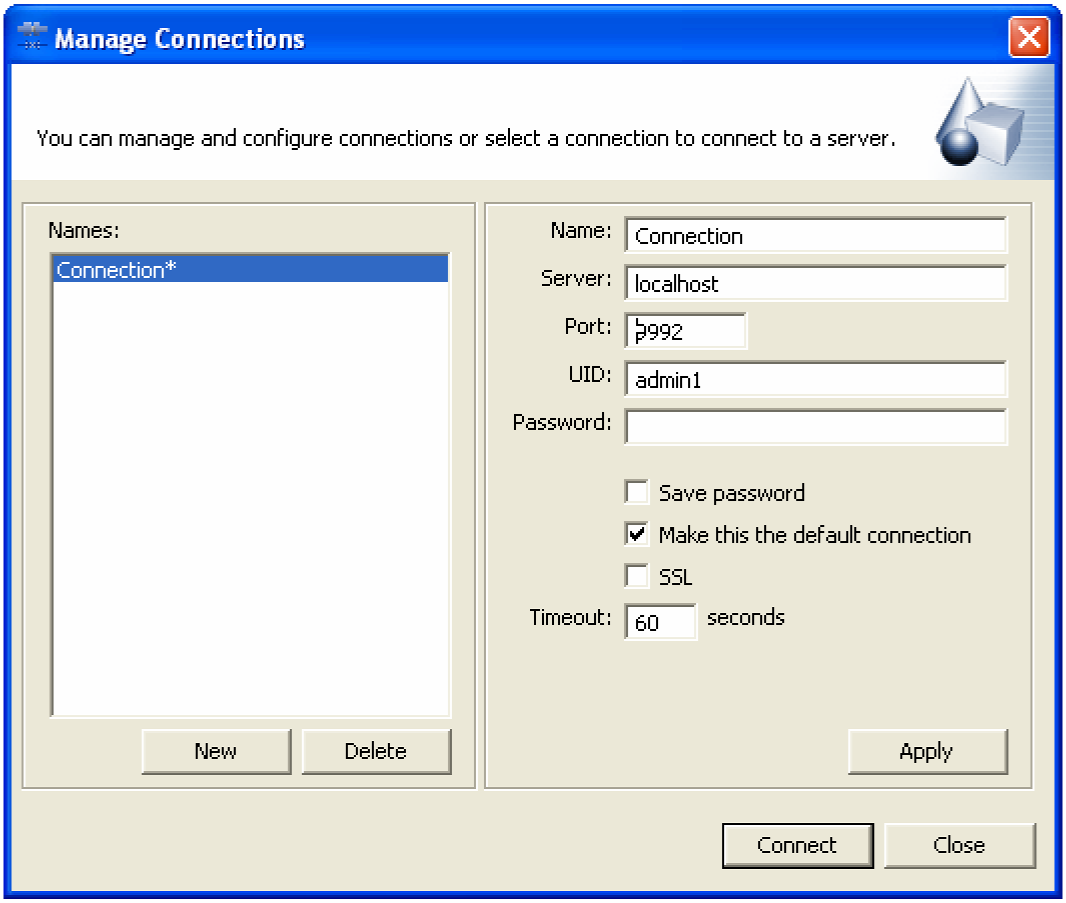
These clients are part of the Rhythmyx Developer Tools. Rhythmyx Developer Tools are certified only on Microsoft Windows operating systems. Use of these clients on other operating systems is not supported.

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:



You can manage and configure connections or select a connection to connect to a server.

Connection*

Name: Connection

Server: localhost

Port: 9992

UID: admin1

Password:

☐ Save password

☑ Make this the default connection

☐ SSL

Timeout: 60 seconds

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 ***Setting Up SSL*** (on page 401).

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.

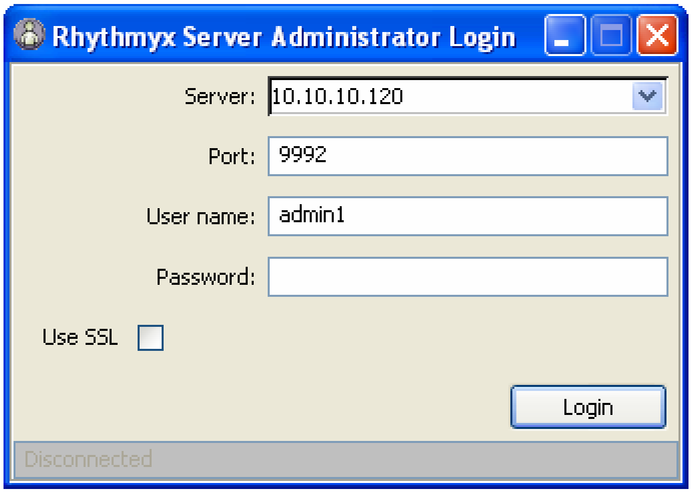
For details about specific Rhythmyx Workbench functionality, see the Rhythmyx Workbench online Help.

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 RhythmyxServerAdministrator.exe.

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



Server: 10.10.10.120

Port: 9992

User name: admin1

Password:

Use SSL □

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 12) 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 21).
2. ***Create the Roles you need to get started with your implementation*** (see "Creating a Role" on page 24).
3. ***Add users to Roles*** (see "Adding Users to a Role" on page 28).
4. ***Create a Community for your Site*** (see "Creating a New Community" on page 33).
5. ***Create a Workflow, defining the process for approval and publication of content in your Site*** (see "Workflows" on page 35).

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:

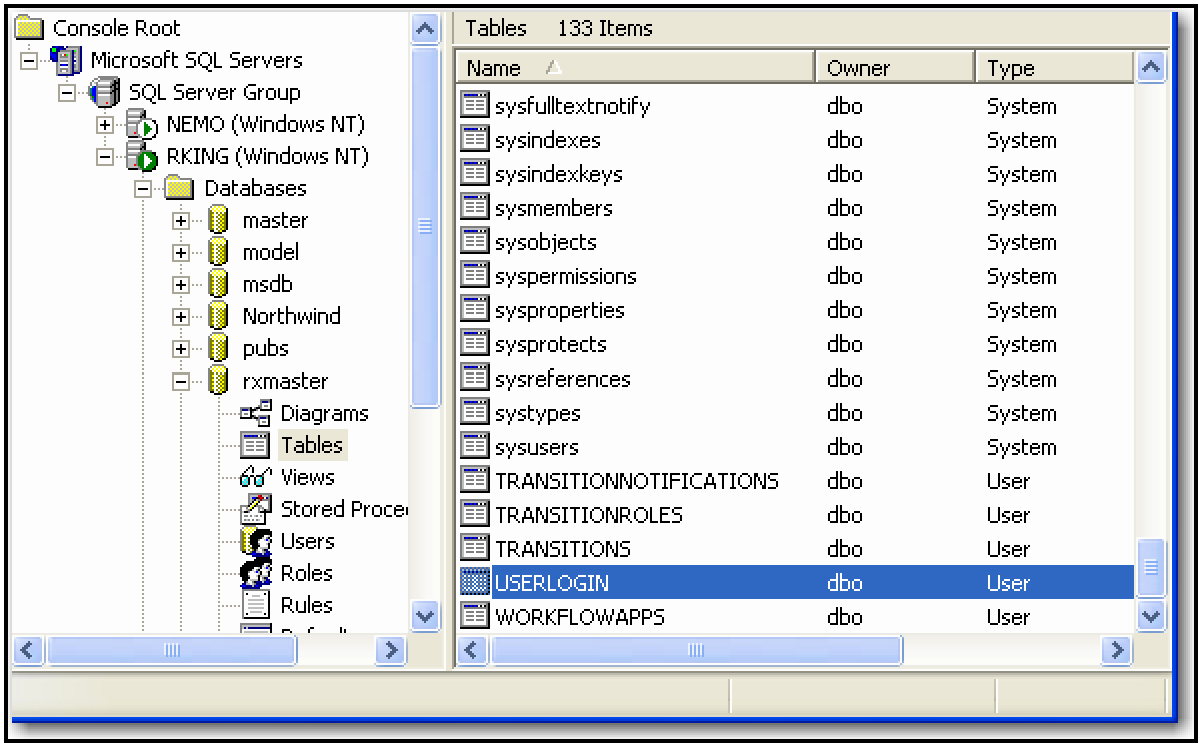
USERNAME	PASSWORD
Lisa Kerr	lisakerr
Ed Wong	edwong
Rita Perez	ritaperez

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.



Name	Owner	Type
sysfulltextnotify	dbo	System
sysindexes	dbo	System
sysindexkeys	dbo	System
sysmembers	dbo	System
sysobjects	dbo	System
syspermissions	dbo	System
sysproperties	dbo	System
sysprotects	dbo	System
sysreferences	dbo	System
systypes	dbo	System
sysusers	dbo	System
TRANSITIONNOTIFICATIONS	dbo	User
TRANSITIONROLES	dbo	User
TRANSITIONS	dbo	User
USERLOGIN	dbo	User
WORKFLOWAPPS	dbo	User

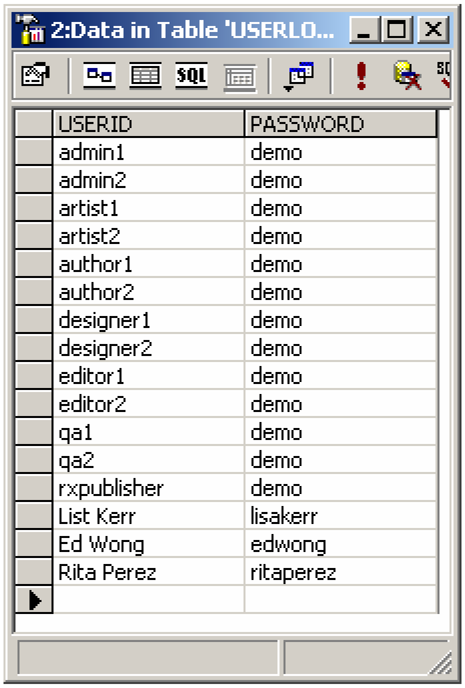
Figure 4: USERLOGIN Table

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.



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 (USERID, PASSWORD) values 'Lisa Kerr', 'lisakerr'
insert into USERLOGIN (USERID, PASSWORD) values 'Ed Wong', 'edwong'
insert into USERLOGIN (USERID, 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 38) 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

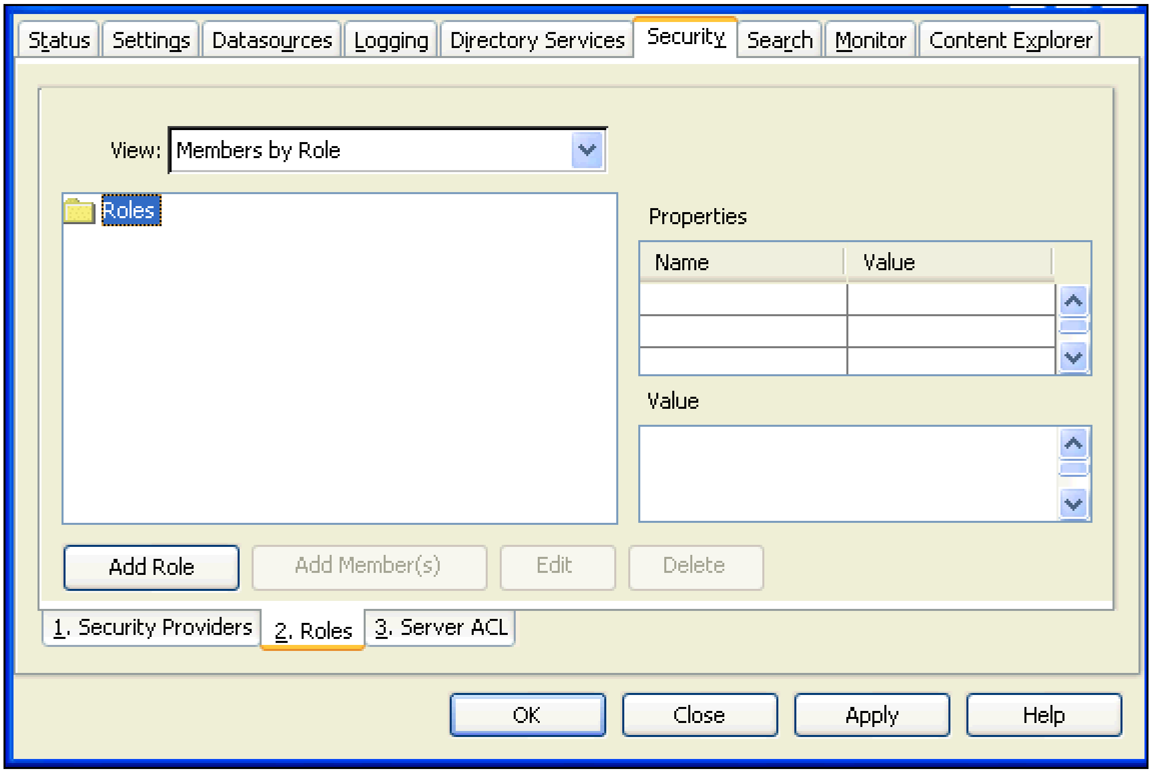
To illustrate the process of creating a Role, we will create the EI_Members Role. Later, we will assign this Role to the Enterprise Investments Community to grant users access to that Community.

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 is 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 18).

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.



View: Members by Role

Figure 6: Server Admin Roles Tab

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

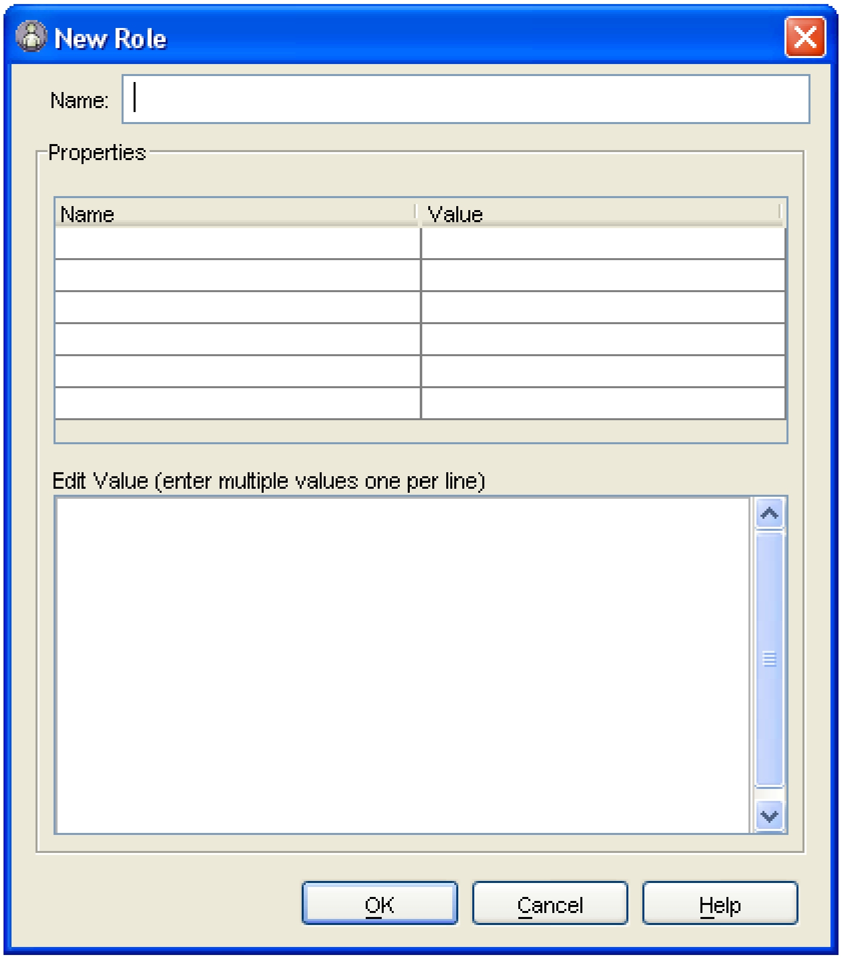
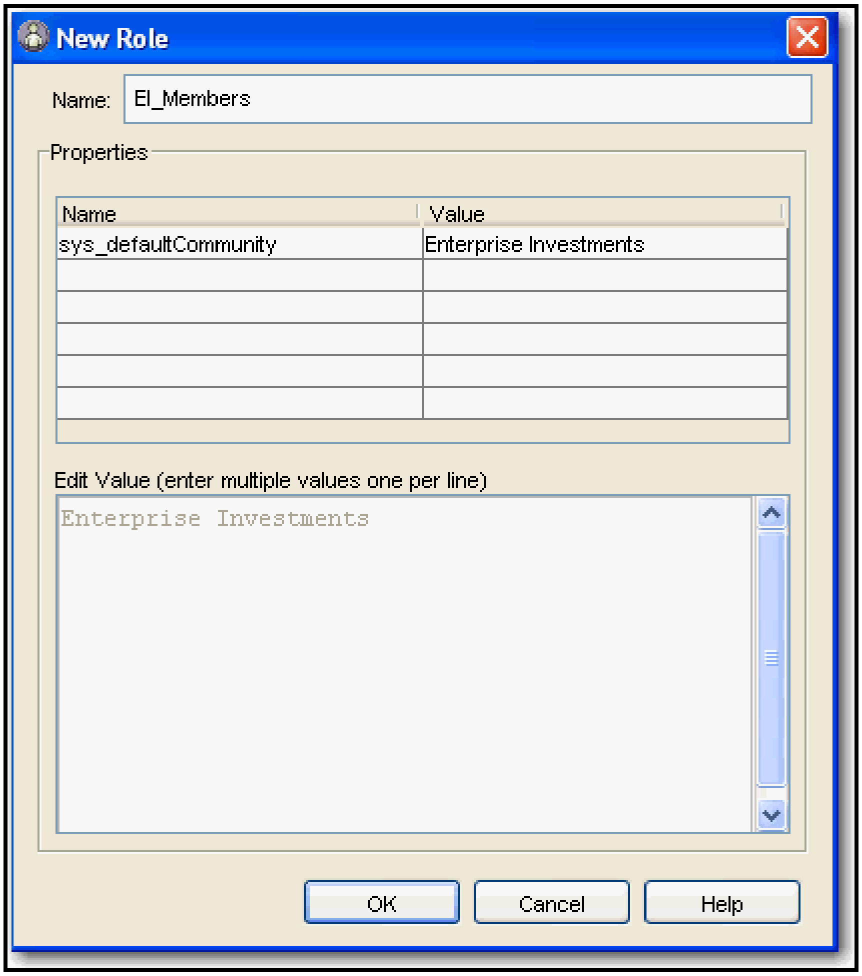


Figure 7: New Role Dialog

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*.



Name: EI_Members

Properties

Name	Value
sys_defaultCommunity	Enterprise Investments

Edit Value (enter multiple values one per line)

```
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.

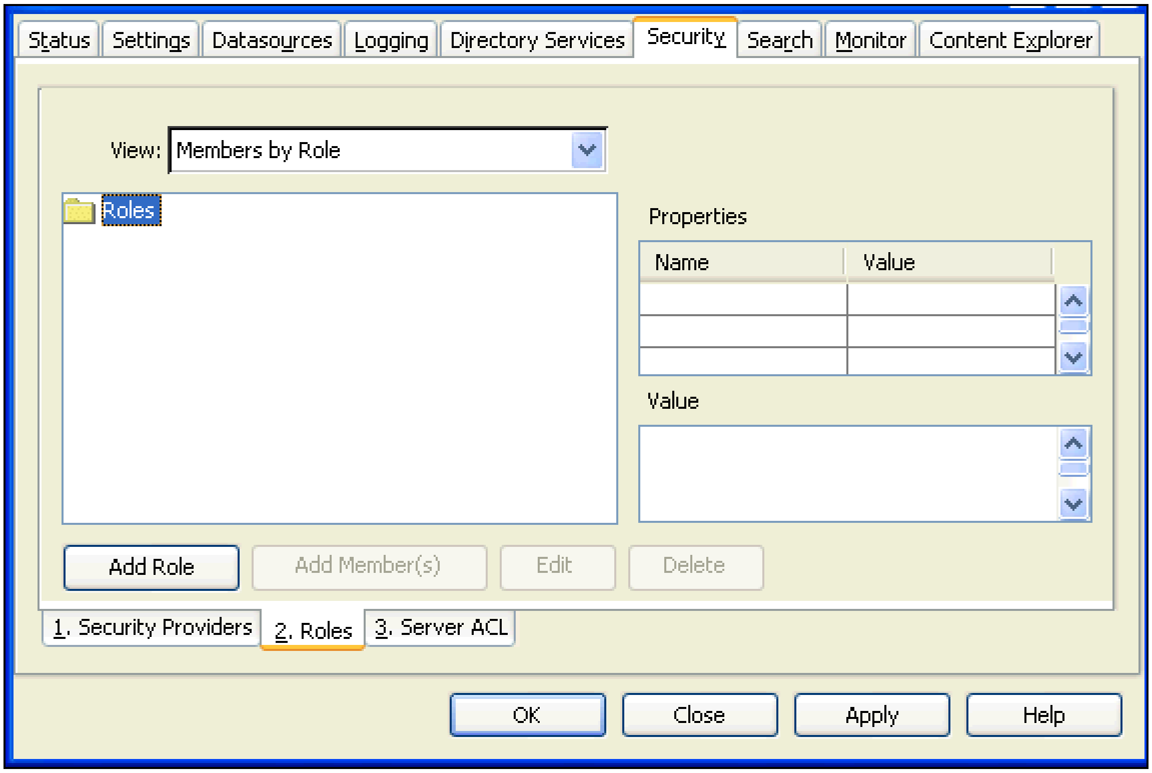
Adding Users to a Role

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 21) (Lisa Kerr, Ed Wong, and Rita Perez) to the EI_Members Role we created in ***Creating a Role*** (see page 24).

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:

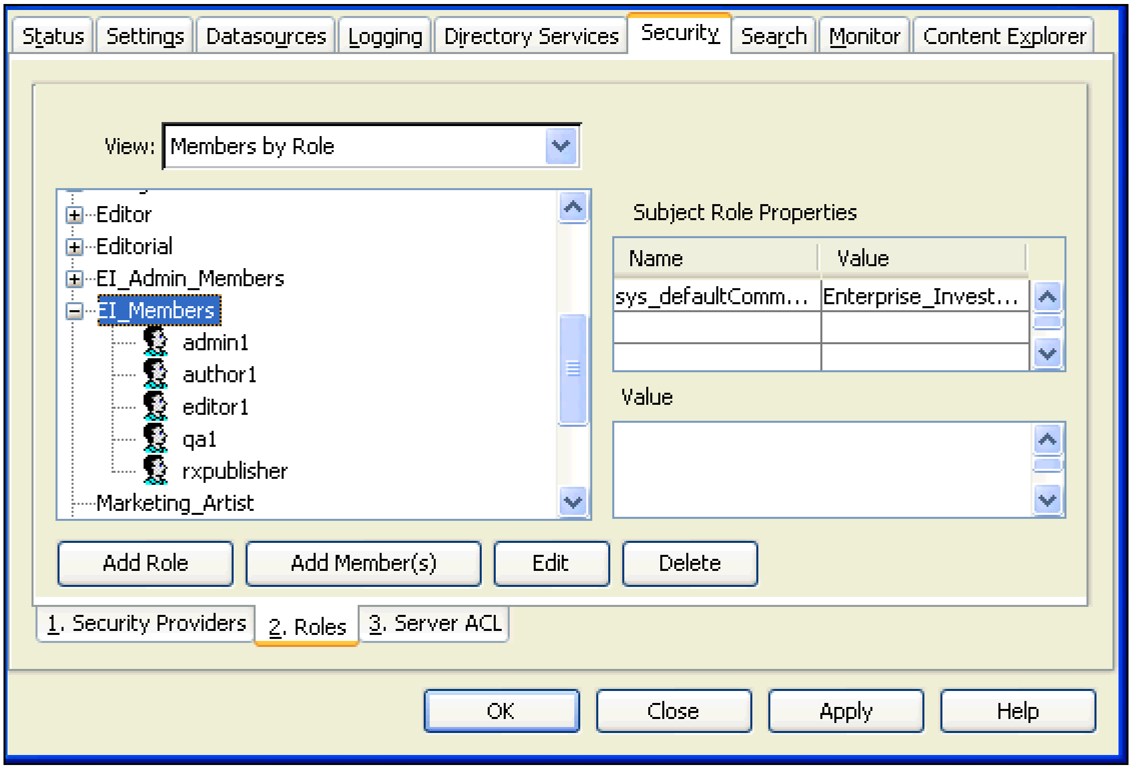
1 Open the Rhythmyx Server Administrator, click the Security tab along the top, and then click the Roles tab along the bottom. The Server Administrator displays the Roles tab dialog.



View: Members by Role

Figure 9: Server Admin 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.



View: Members by Role

- Editor
- Editorial
- EI_Admin_Members
- EI_Members
  - admin1
  - author1
  - editor1
  - qa1
  - rxpublisher
- Marketing_Artist

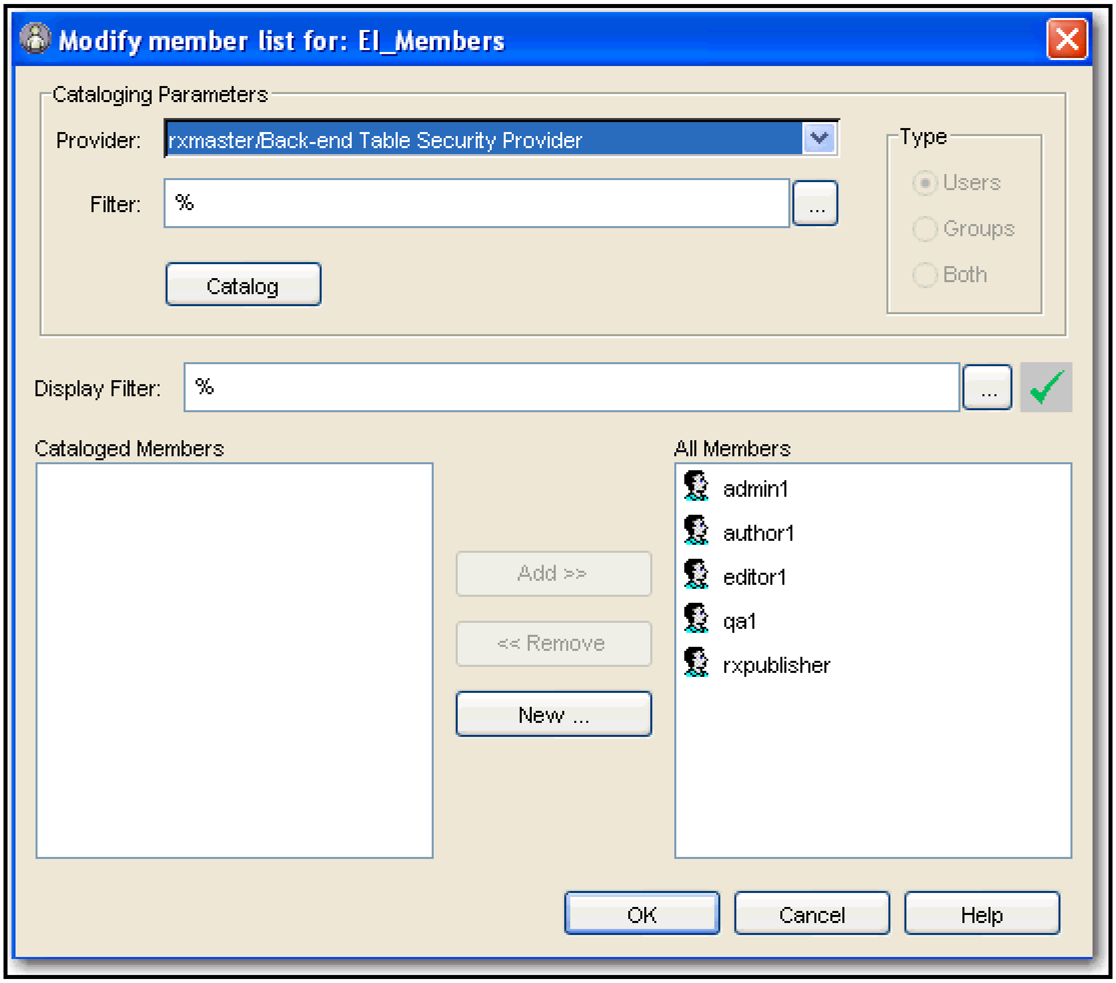
Subject Role Properties

Name	Value
sys_defaultComm…	Enterprise_Invest…

Value

Figure 10: Roles tab showing EI_Members by Role

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



Modify member list for: EI_Members

Cataloging Parameters

Provider: rxmaster/Back-end Table Security Provider

Filter: %

Type: ● Users ○ Groups ○ Both

Display Filter: %

Cataloged Members

All Members:
- admin1
- author1
- editor1
- qa1
- rxpublisher

Figure 11: Modify Member List for: EI_Members Dialog

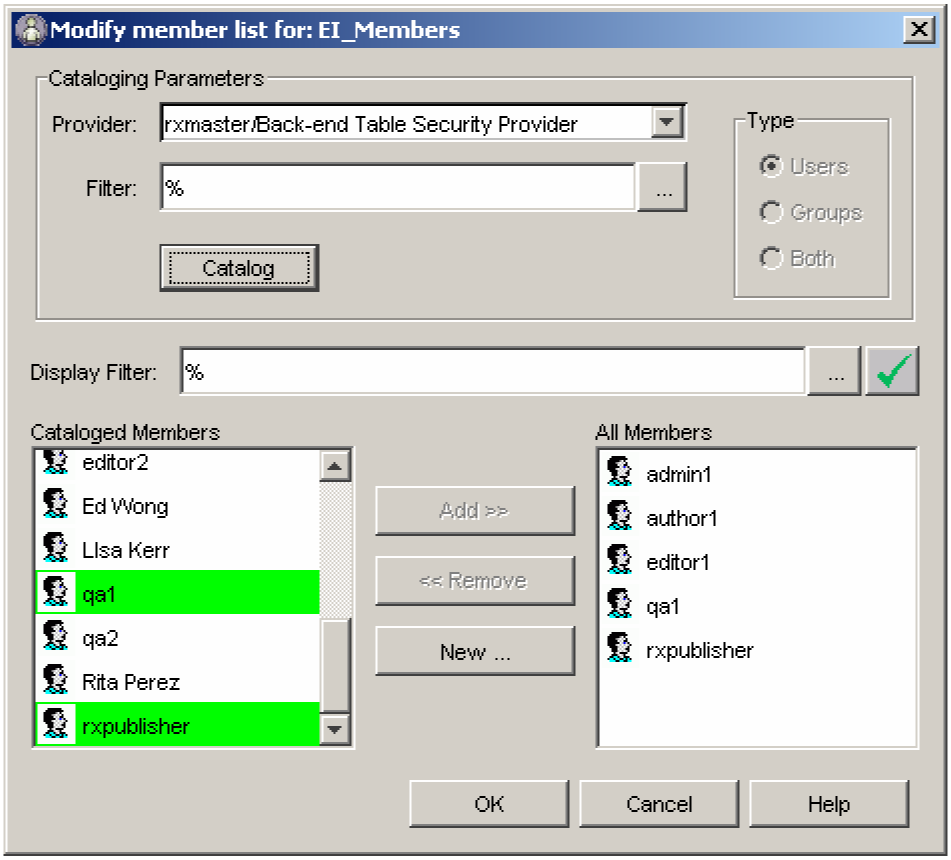
You can retrieve a list of members available from each Security Provider. This function, known as cataloging, makes it easier to add new Members to the Role. Note that you can also add new Members manually, but cataloging is generally more efficient and less error prone, and ensures that the Members 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 21).

b) Click the [**Catalog**] button.

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



Modify member list for: EI_Members

Cataloging Parameters

Provider: rxmaster/Back-end Table Security Provider

Filter: %

Type: ● Users ○ Groups ○ Both

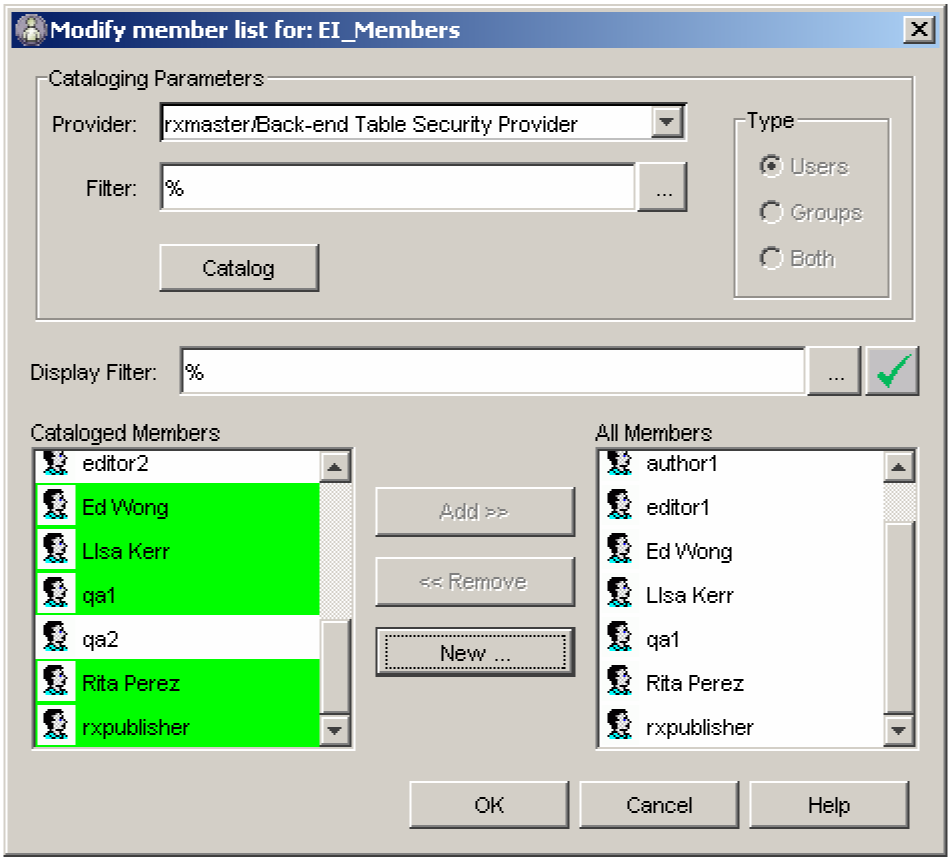
Display Filter: %

Cataloged Members:
- editor2
- Ed Wong
- LIsa Kerr
- qa1
- qa2
- Rita Perez
- rxpublisher

All Members:
- admin1
- author1
- editor1
- qa1
- rxpublisher

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

5 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.



Modify member list for: EI_Members

Cataloging Parameters

Provider: rxmaster/Back-end Table Security Provider

Filter: %

Type: ● Users ○ Groups ○ Both

Display Filter: %

Cataloged Members: editor2, Ed Wong, LIsa Kerr, qa1, qa2, Rita Perez, rxpublisher

All Members: author1, editor1, Ed Wong, LIsa Kerr, qa1, Rita Perez, rxpublisher

Figure 13: Modify Members dialog showing new Members added to the EI_Members Role

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

NOTE: Rhythmyx automatically adds the user rxserver to the Admin Role. Since the rxserver user is required to perform Aging Transitions (see "Implementing an Aging Transition" *on page 44), you must either use the Admin Role as the Workflow administrator Role or reassign the rxserver user to the Role that you want to use as the Workflow administrator Role. For details, see About the Workflow Administrator (on page 35).*

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

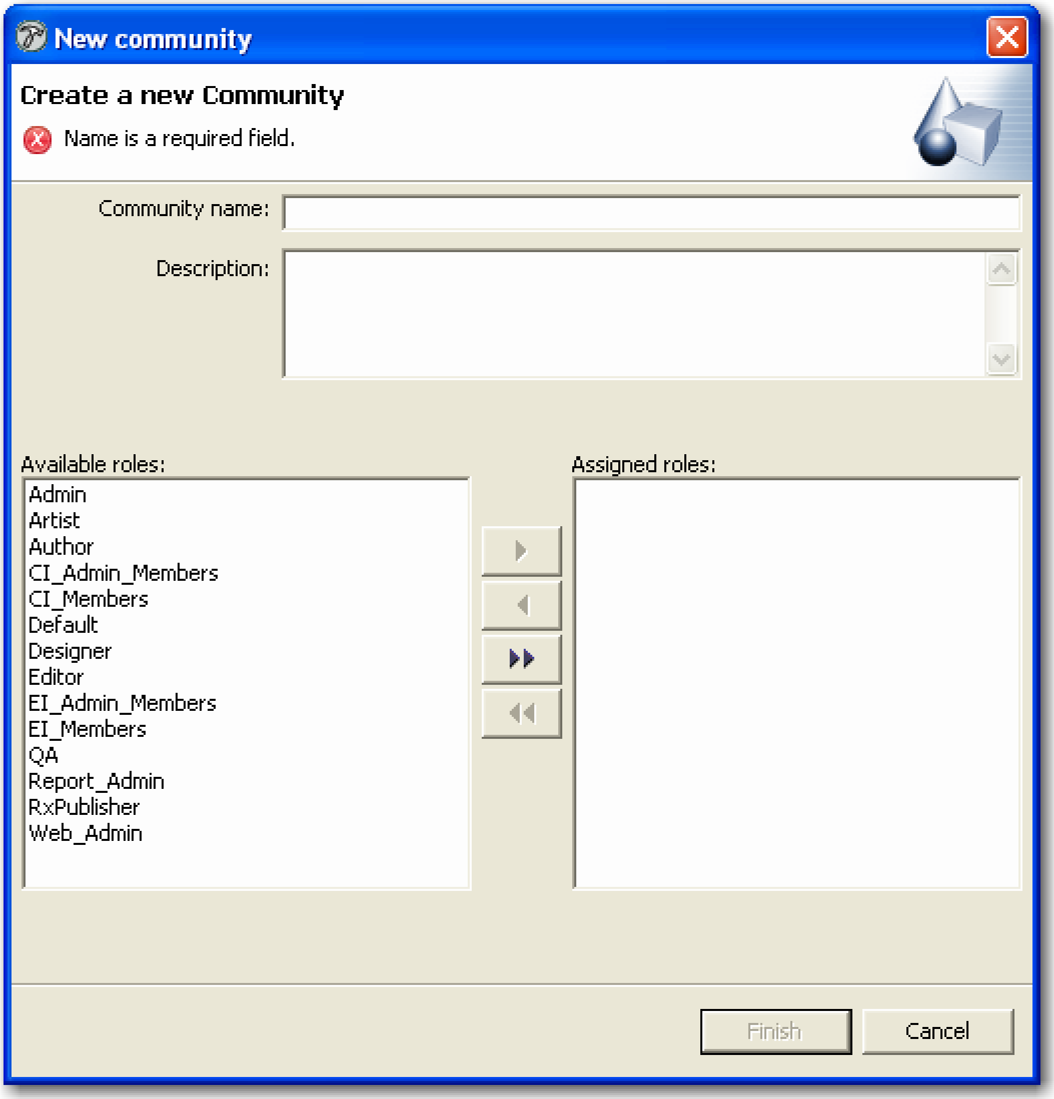
To illustrate the process of creating a Community, we will create the Enterprise Investments Community, with the description "*Community for users that create and manage Content Items for the Enterprise Investments Site*". We will assign ***the EI_Members Role that we created earlier*** (see "Creating a Role" on page 24) 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:

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.

About the Workflow Administrator

You must have a role that represents the Workflow administrator that you give ***assignee access*** (see "Assigning a Role to a State" on page 40) to all States. The purpose of this role is to be able to edit and transition content in any state and 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. The Workflow administrator role assigns designated end users the power to check in content items.

In addition, you must always have a user named *rxserver* that has at least ***reader access*** (see "Assigning a Role to a State" on page 40) to all States. However, in general, it is recommended that the *rxserver* user be assigned to the Workflow administrator role because it enables aging transitions to occur. Therefore, the *rxserver* user usually has ***assignee access*** (see "Assigning a Role to a State" on page 40) to every State.

By default, Rhythmyx assigns *rxserver* to the Admin role. If you want to use a different role as a Workflow administrator, you must assign *rxserver* to that role.

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. For details, see ***About the Workflow Administrator*** (on page 35). 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.

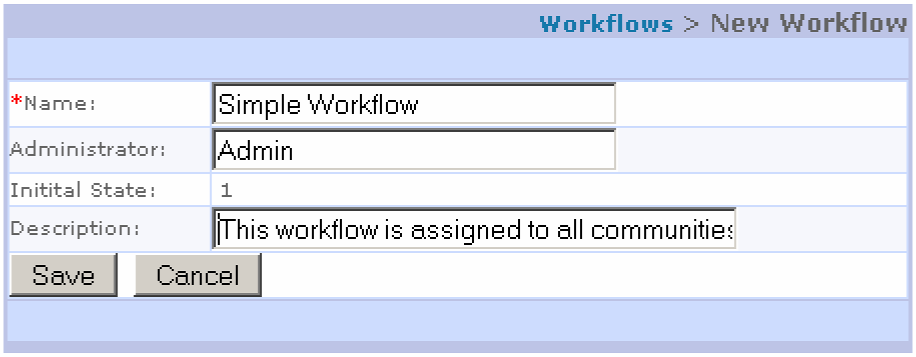
Content Explorer displays the New Workflow page.

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.



*Name:	Simple Workflow
Administrator:	Admin
Initital State:	1
Description:	This workflow is assigned to all communitie

Figure 15: Creating the Simple Workflow Object

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

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 the Simple Workflow with the Enterprise Investments Community:

1 Start the Rhythmyx Workbench. For instructions, see ***Starting the Rhythmyx Workbench*** (on page 16).

2 Display the Community Visibility View. To display the Community Visibility View:

a) In the Menu bar of the Rhythmyx Workbench, choose *Window > Show View > Other*.

The Rhythmyx Workbench displays the Show View dialog

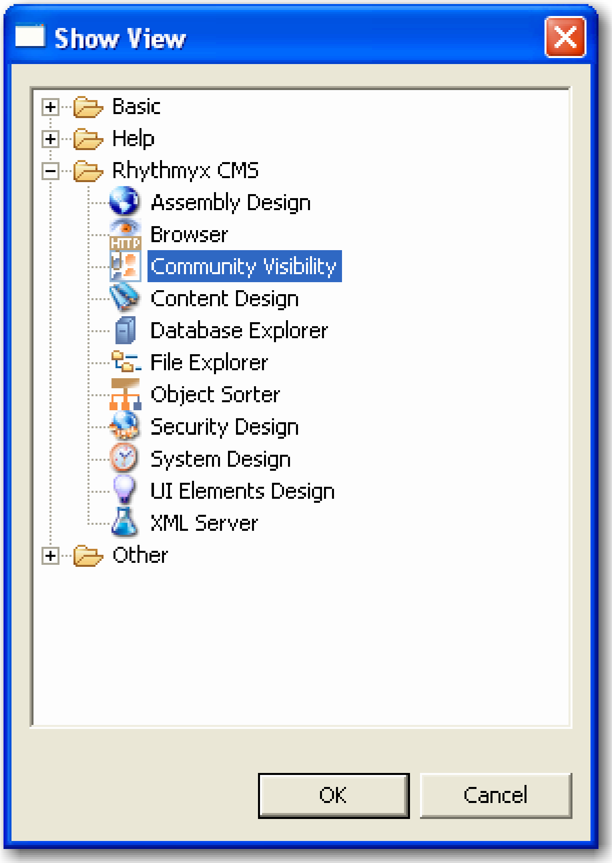


Figure 16: Show View dialog

b) Select Community Visibility and click the [**OK**] button.

c) Drag the Community Visibility View to the working area of the Rhythmyx Workbench.

3 In Community Visibility View, expand the Enterprise Investments Community, then expand the Visible Workflows node.

4 Display the System Design View.

5 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.

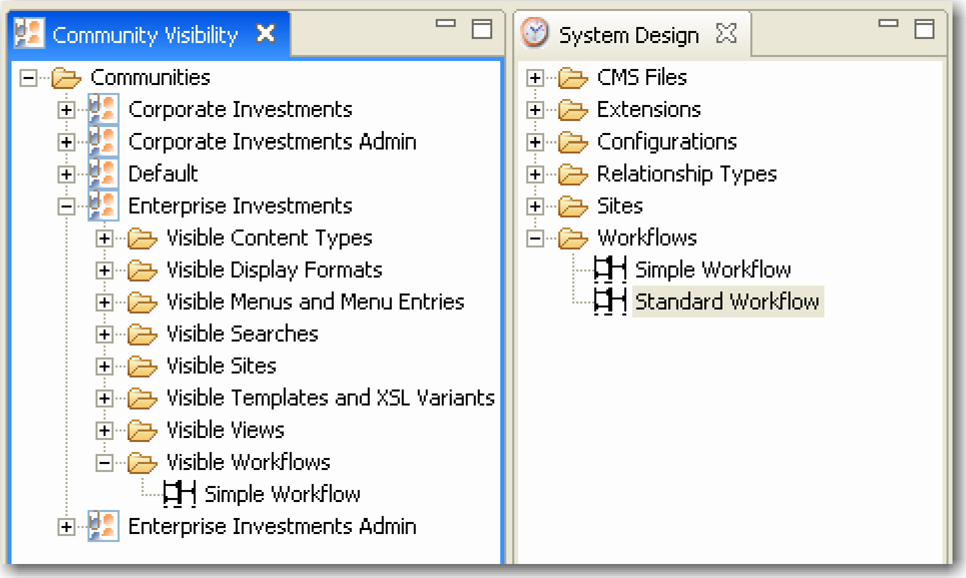


Figure 17: Standard Workflow added to the Enterprise Investments Community

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 24)". 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 472), 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

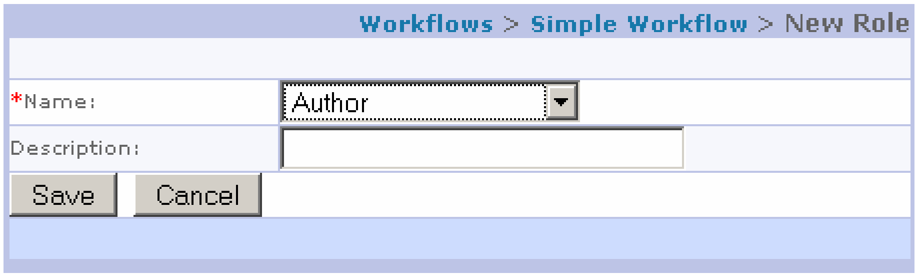
NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

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

1. Log in to Content Explorer and click the Workflow tab.
2. Click the Simple Workflow link in the Name (ID) column.

   Content Explorer displays the Simple Workflow page.
3. Click the New Roles link.

   Content Explore displays the New Role page.
4. In the Name drop list, choose *Author*. (The drop list contains all Roles in the system not already added to the Workflow.)



Workflows > Simple Workflow > New Role

*Name: Author

Description:

Figure 18: Adding a Role to a Workflow

5. Click Save. You have added the Role to the Workflow and can now assign the Role 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 472), 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

Publishable Value	Description	Example States in the Simple Workflow
Ignore	Publish the Last Public Revision of the Content Item.	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

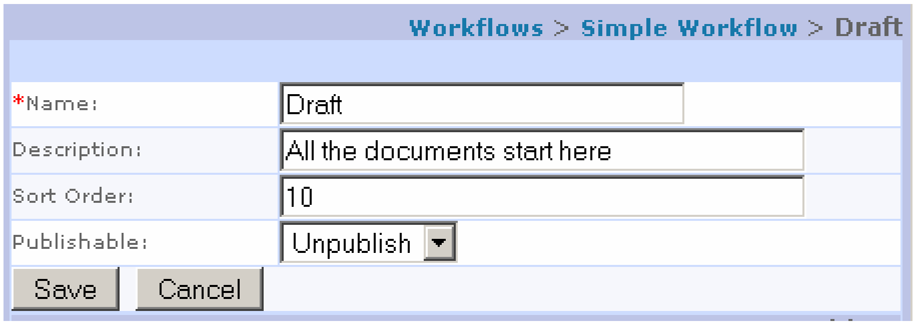
Note that it is possible to add other values to extend the Publishable functionality. For details see "Extending Publishable States" in the *Rhythmyx Technical Reference Manual*.

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*.



Workflows > Simple Workflow > Draft

*Name:	Draft
Description:	All the documents start here
Sort Order:	10
Publishable:	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 38) 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. You must give the Role you define as the Workflow administrator assignee access to all States. For details, see ***About the Workflow Administrator*** (on page 35).
- 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	**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

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.

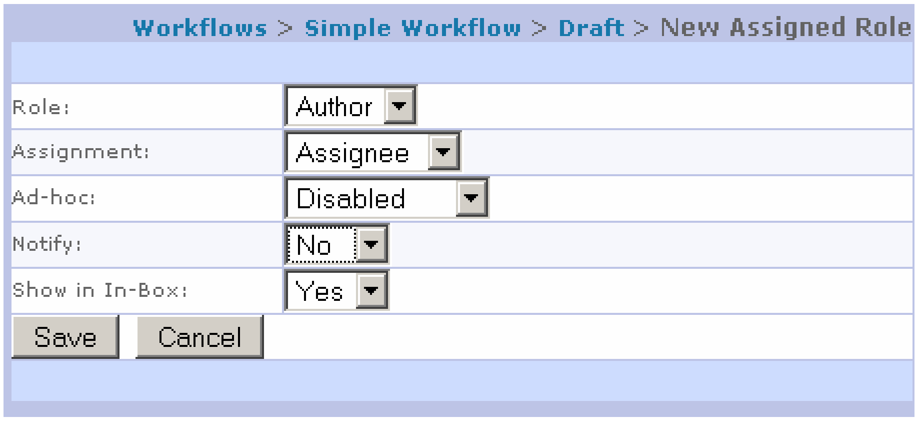
   Content Explorer displays the New Assigned Role page.

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.



Workflows > Simple Workflow > Draft > New Assigned Role

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

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.

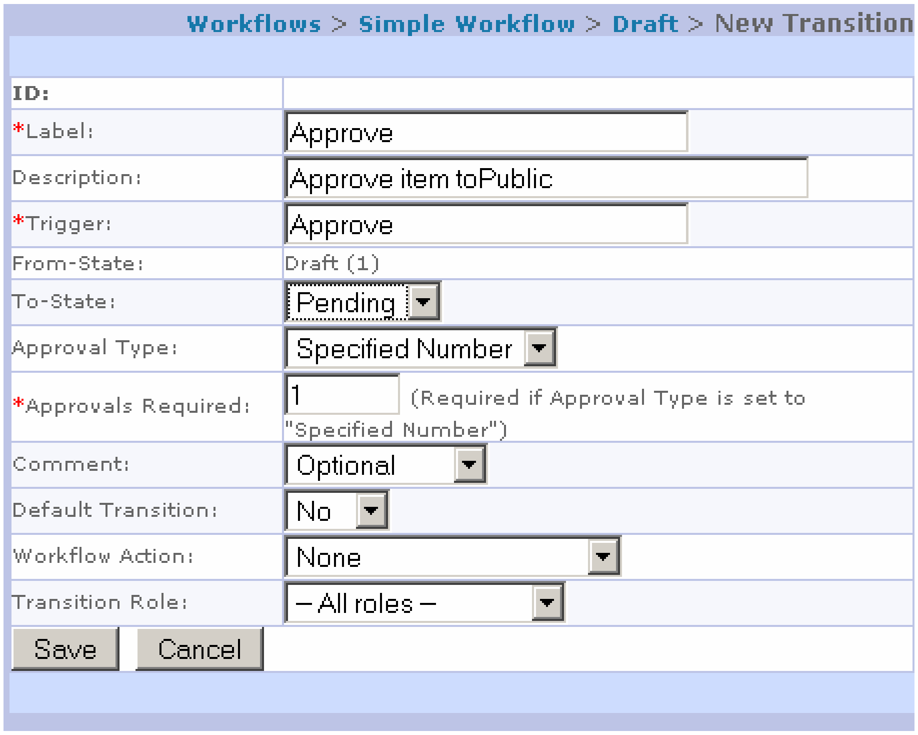
6 In the **Description** field, enter *Approve Item to Pending*.

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 --

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.)

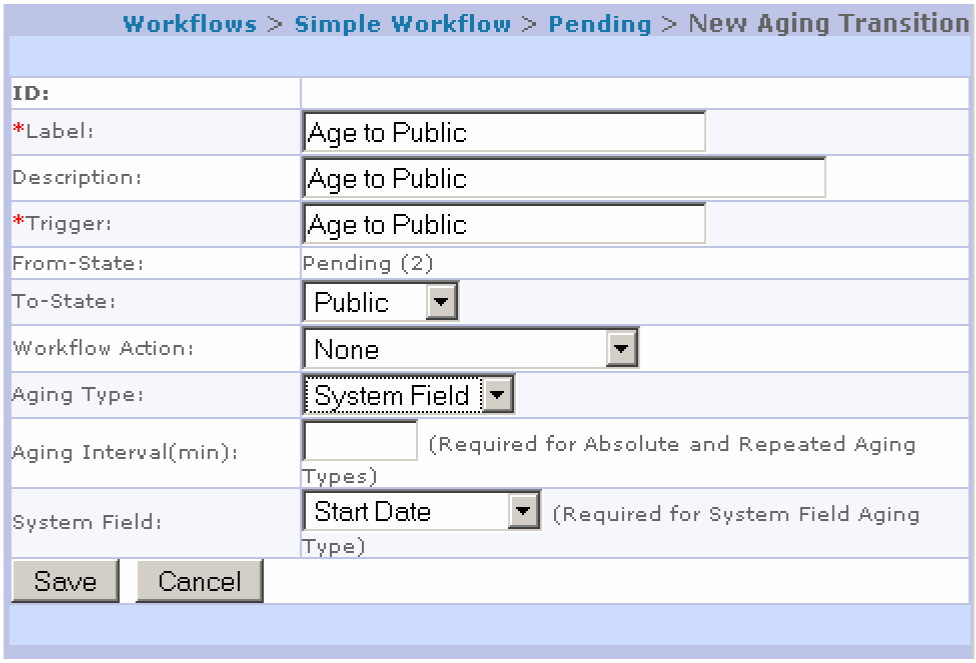
Content Explorer displays the New Aging Transition page.

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)

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.

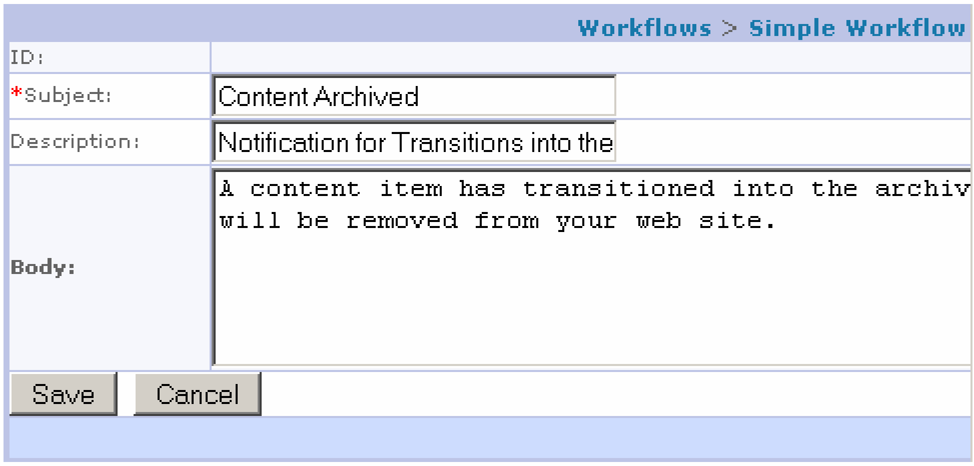
3 In the Notifications section, click the New Notifications link.

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.*



ID:

*Subject: Content Archived

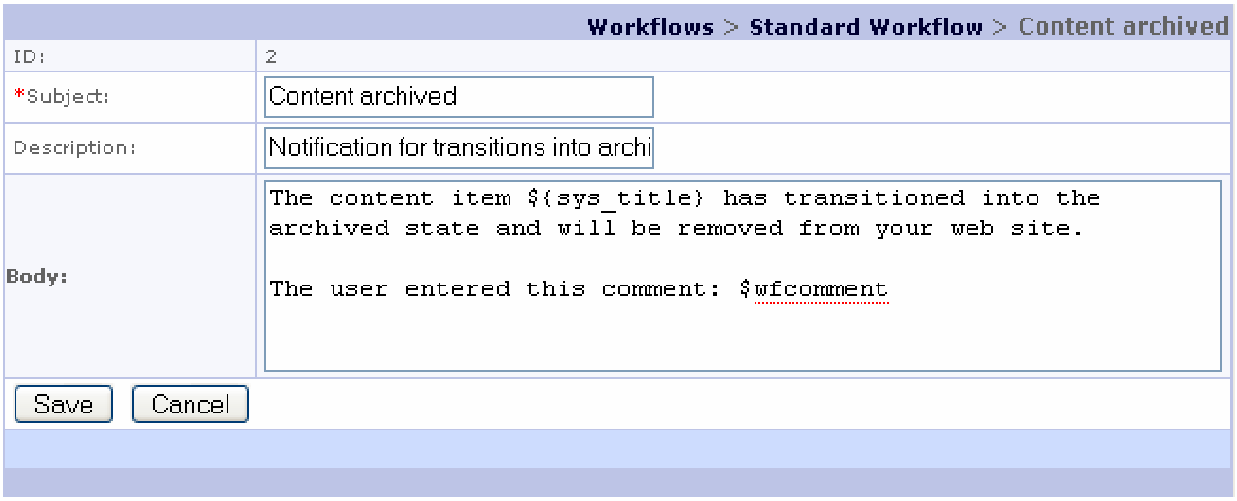
Description: Notification for Transitions into the

Body: A content item has transitioned into the archiv
will be removed from your web site.

Figure 23: Defining the Content Archived Notification

If you want to include any comments the user entered when making the transition, include the macro $wfcomment.

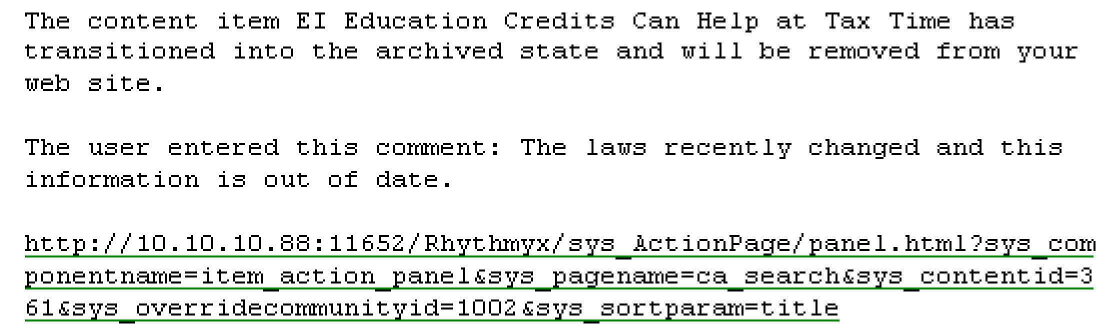
You can also include data from any field in the Rhythmyx Content Item by adding ${fieldname}, where fieldname is the name of the Rhythmyx field whose data you want to include. For example:



ID:	2
*Subject:	Content archived
Description:	Notification for transitions into archi
Body:	The content item ${sys_title} has transitioned into the archived state and will be removed from your web site. The user entered this comment: $wfcomment

Figure 24: Notification with $wfcomment macro and the field sys_title added

This Notification would result in the following e-mail message:



```
The content item EI Education Credits Can Help at Tax Time has
transitioned into the archived state and will be removed from your
web site.

The user entered this comment: The laws recently changed and this
information is out of date.

http://10.10.10.88:11652/Rhythmyx/sys_ActionPage/panel.html?sys_com
ponentname=item_action_panel&sys_pagename=ca_search&sys_contentid=3
61&sys_overridecommunityid=1002&sys_sortparam=title
```

Figure 25: Notification e-mail resulting from the example 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.

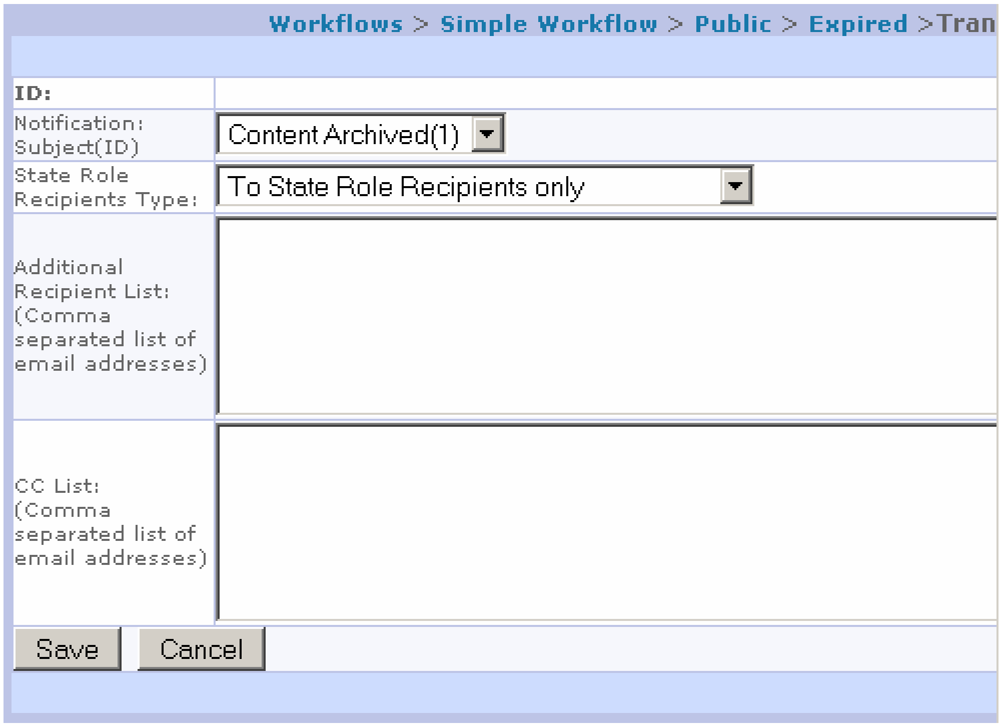
5 In the Transition Notifications section, click the New Transition Notification link.

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.

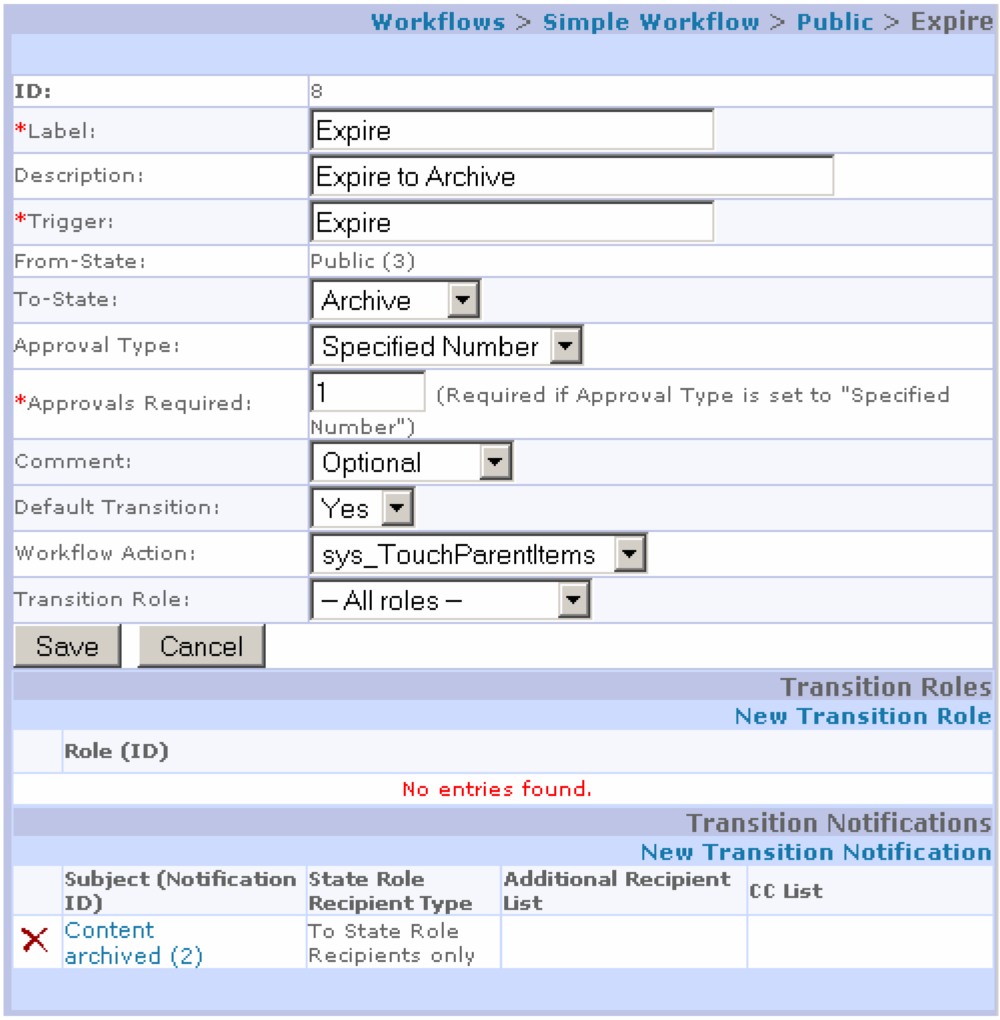


Workflows > Simple Workflow > Public > Expired >Tran

ID:	
Notification: Subject(ID)	Content Archived(1)
State Role Recipients Type:	To State Role Recipients only
Additional Recipient List: (Comma separated list of email addresses)	
CC List: (Comma separated list of email addresses)	

Figure 26: 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 --

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 27: 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_SSLLINK_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 .***Setting Up SSL*** (on page 401).

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\rxworkflow.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 `rxworkflow.properties`.

Implementing Quick Edit

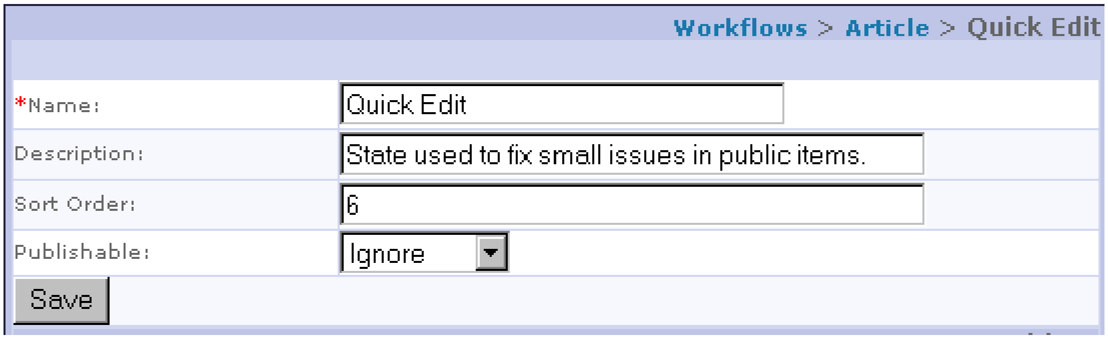
Rhythmyx does not allow users to check out Content Items in a Public State. Therefore, a user must Transition an item from the Public State to an editable State, then check it out, in order to edit the item. This process can be cumbersome if the user only needs to correct a minor misspelling.

To make the process easier for the user, for any Content Item in a Public State, Rhythmyx includes the menu option *Edit > Quick Edit*. This menu option provides a single action that Transitions the Content Item from Public and checks it out to the user. Each Workflow must implement some special feature to make the option available to users.

To implement Quick Edit in a Workflow:

1 The Workflow must include a Quick Edit State.

To make a State a Quick Edit State, in the Publishable drop list, choose *Ignore*. Any State with this value in the Publishable parameter is a Quick Edit State. A Workflow can include any number of Quick Edit States.

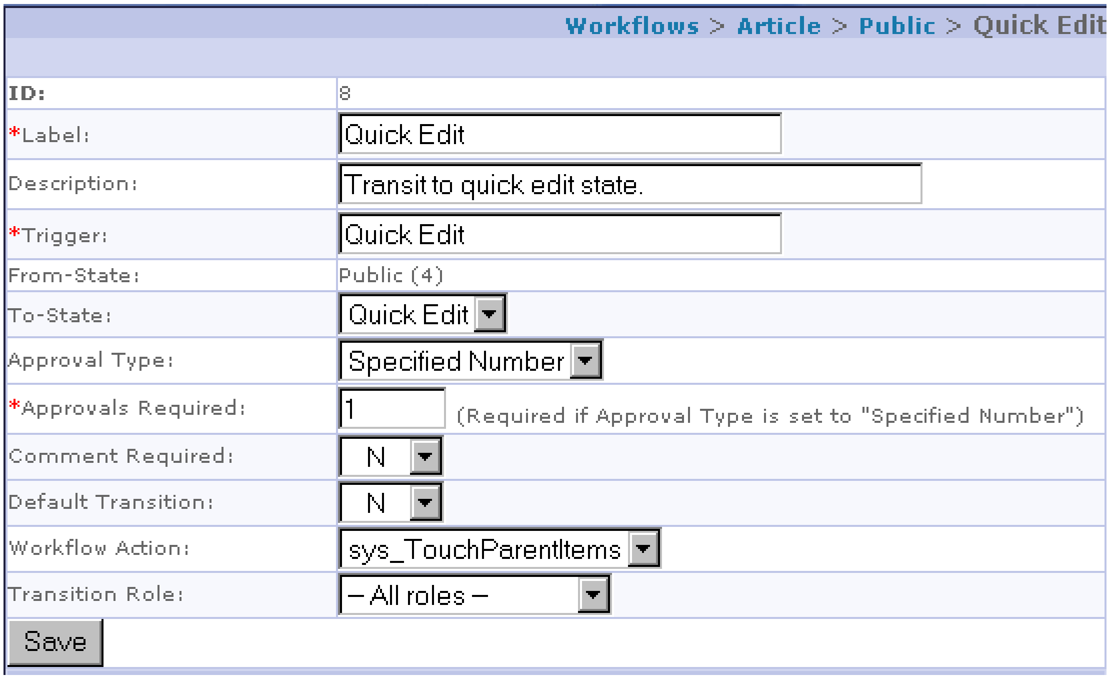


*Name:	Quick Edit
Description:	State used to fix small issues in public items.
Sort Order:	6
Publishable:	Ignore

Figure 28: Quick Edit State Definition in Default Article Workflow

2 Define a Quick Edit Transition from the Public State to the Quick Edit State.

To make a Transition a Quick Edit Transition, the value in the Trigger field must be *Quick Edit*. The server requires this value to activate the Quick Edit processing.



ID:	8
*Label:	Quick Edit
Description:	Transit to quick edit state.
*Trigger:	Quick Edit
From-State:	Public (4)
To-State:	Quick Edit
Approval Type:	Specified Number
*Approvals Required:	1 (Required if Approval Type is set to "Specified Number")
Comment Required:	N
Default Transition:	N
Workflow Action:	sys_TouchParentItems
Transition Role:	-- All roles --

Figure 29: Quick Edit Transition from the Default Article Workflow

Note that each Public State can have only one Quick Edit Transition because the value of the Trigger field for each Transition in the State must be unique. If multiple Transitions share the same Trigger, Rhythmyx cannot determine which to execute.

A Workflow can include multiple Public States, however, and each State may have its own Quick Edit Transition, because Transitions in different States can have the same Trigger.

3 Define a Transition from the Quick Edit State back to the Public State. In the Workflow Actions drop list, choose *sys_TouchParentItems* to ensure that all related content items are updated with the changes made during Quick Edit.

The following graphic shows an portion of a State diagram of a Workflow in which Quick Edit has been implemented. Note the Quick Edit State, the Quick Edit Transition to the State, and the Publish Transition back to Public.

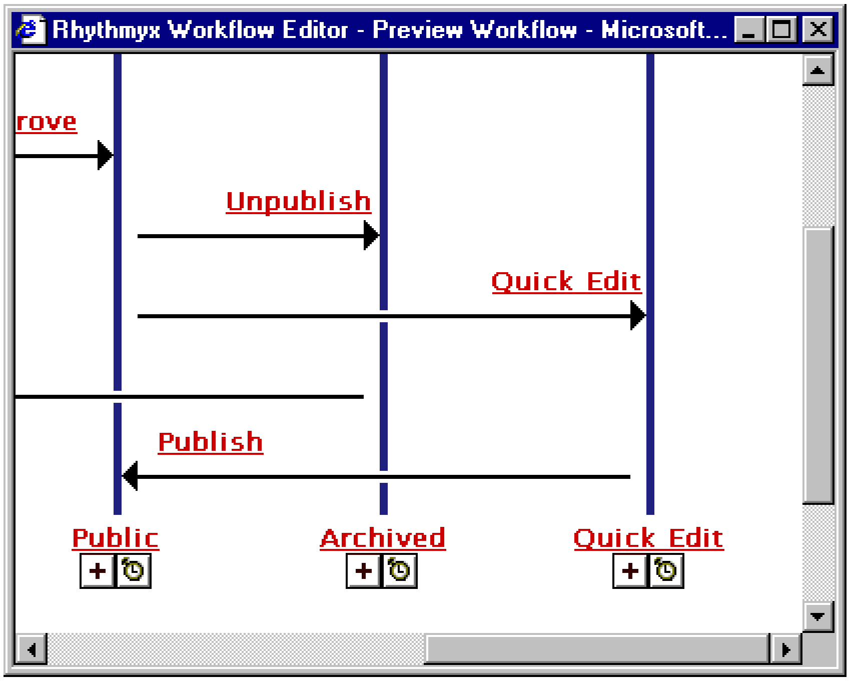


Figure 30: Workflow Diagram Showing Quick Edit

NOTE: If a Content Item is in a Quick Edit State, the Publishing engine publishes the Content Item's last public Revision. If a Content Item is linked to a Related Content Item that is currently in a Quick Edit State, the Publishing engine publishes a link to the last public Revision of the Related Content Item.

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 Public State;

- change the Approve Transition from Review to Pending to require approvals from both the Editor and Web Admin 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:

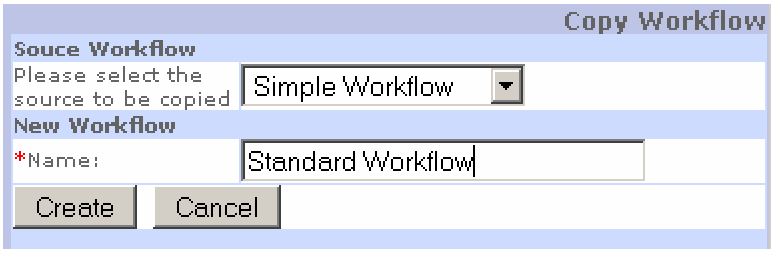
1 Log in to Content Explorer and click the Workflow tab.

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*.



Copy Workflow

Souce Workflow

Please select the source to be copied: Simple Workflow

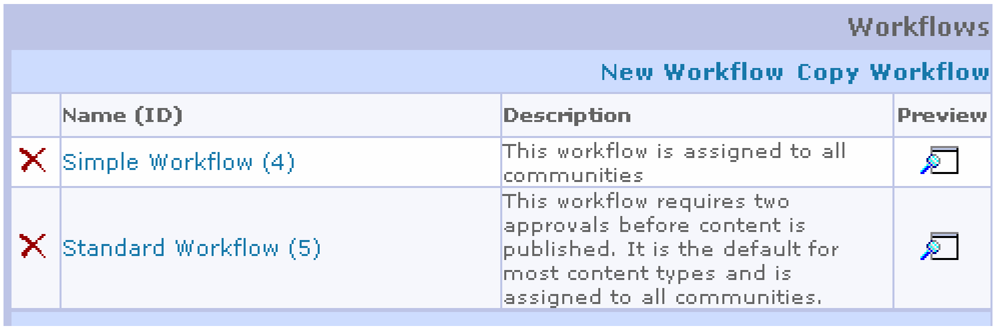
New Workflow

*Name: Standard Workflow

Figure 31: 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

	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 32: 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.

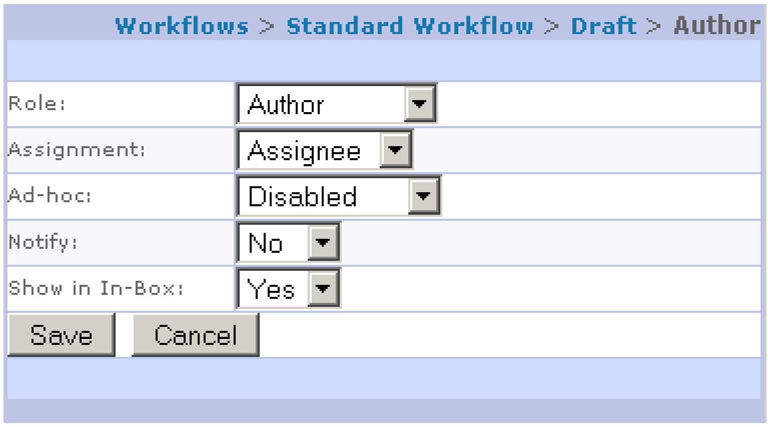
2 Click the Standard Workflow name in the Name (ID) column.

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.



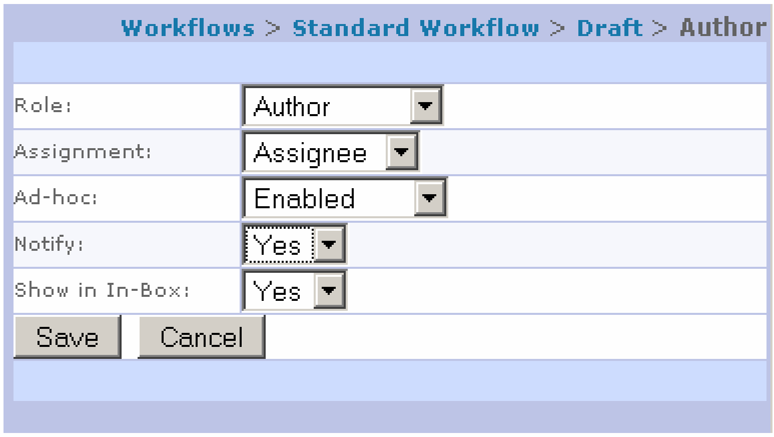
Workflows > Standard Workflow > Draft > Author

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

Figure 33: 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

Figure 34: 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:

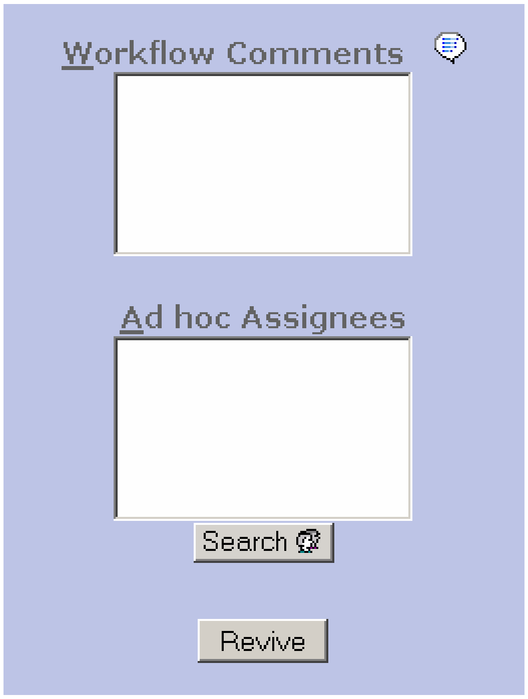
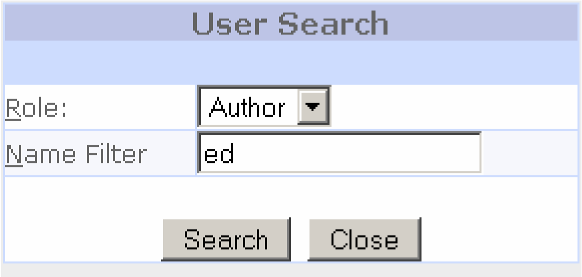


Figure 35: Content Explorer Transition dialog for Workflow Comments and Ad Hoc assignee

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:



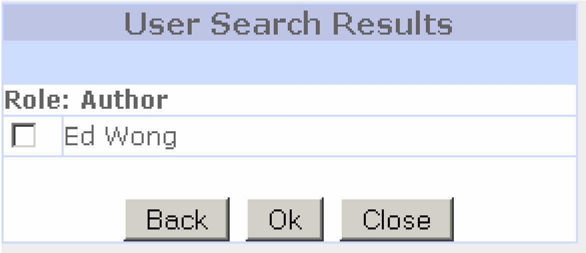
User Search

Role: Author

Name Filter: ed

Figure 36: Searching for users with names beginning "ed"

which returns the following results:



User Search Results

Role: Author

□ Ed Wong

Figure 37: 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 39). 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 use 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 43).

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

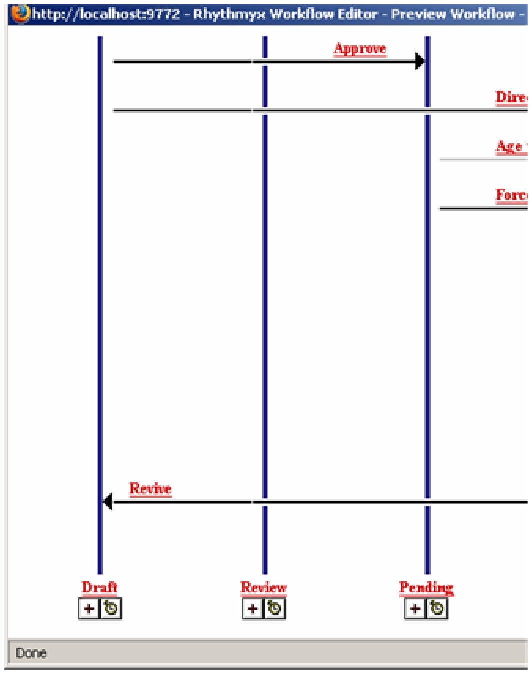

http://localhost:9772 - Rhythmyx Workflow Editor - Preview Workflow -
Approve
Dire
Age
Forc
Revive
Draft
Review
Pending
Done

Figure 38: 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:

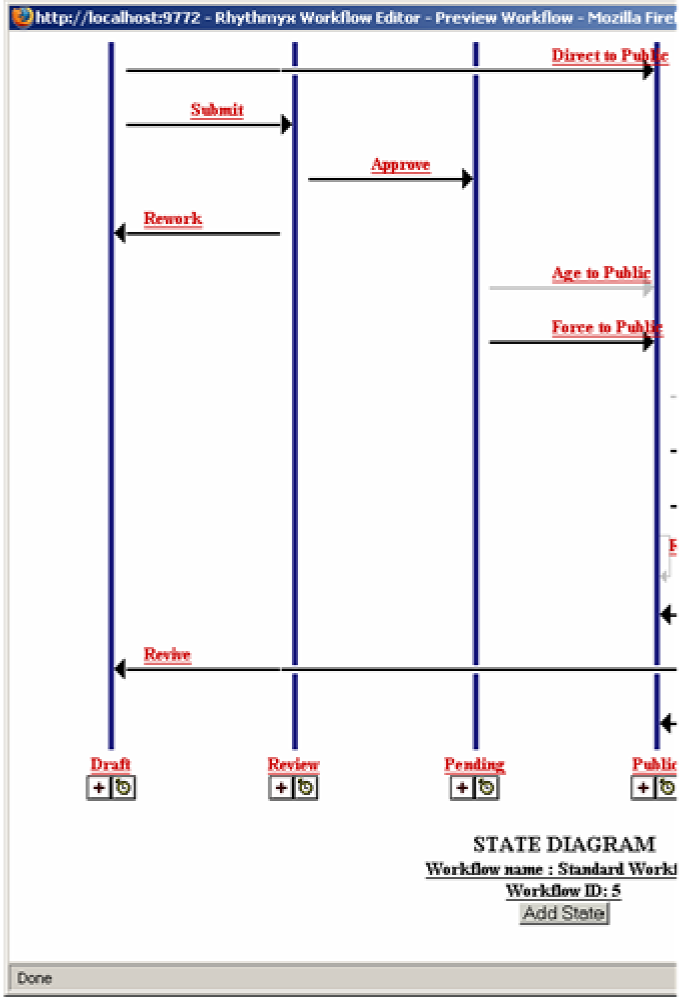


Figure 39: Preview of Standard Workflow after updating Transitions

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.

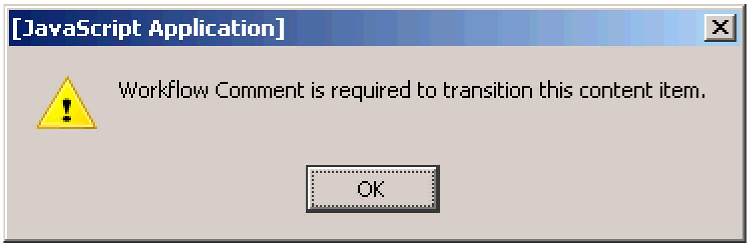
   Content Explorer displays the Standard Workflow page.
3. In the States section of the page, click Review link.

   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:

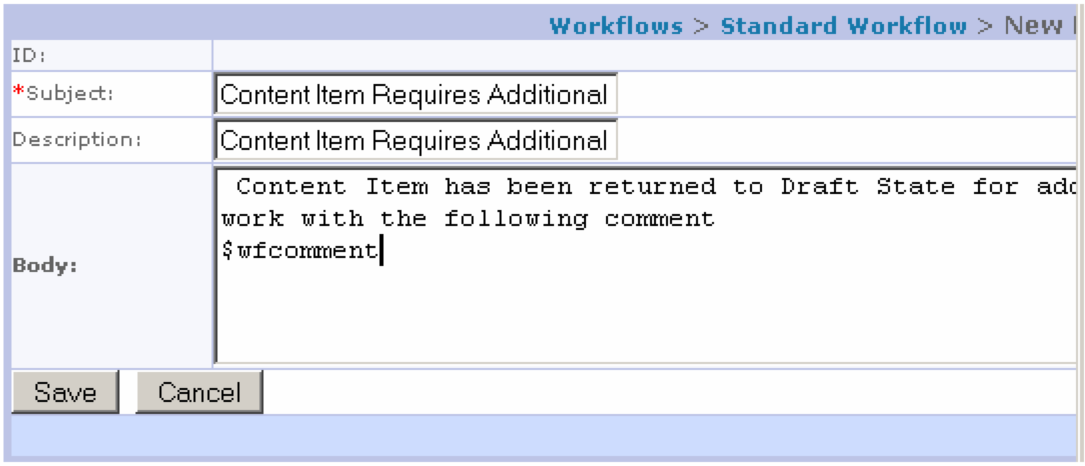


Workflow Comment is required to transition this content item.

Figure 40: 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 46), but we would add the macro `$wfcomment` on a new line:



Subject: Content Item Requires Additional

Description: Content Item Requires Additional

Body:

```
 Content Item has been returned to Draft State for adc
work with the following comment
$wfcomment
```

Figure 41: Notification with $wfcomment macro

Now, if the user assigned the Content Item to Ed Wong and included a message, Ed Wong would receive an e-mail message.

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.

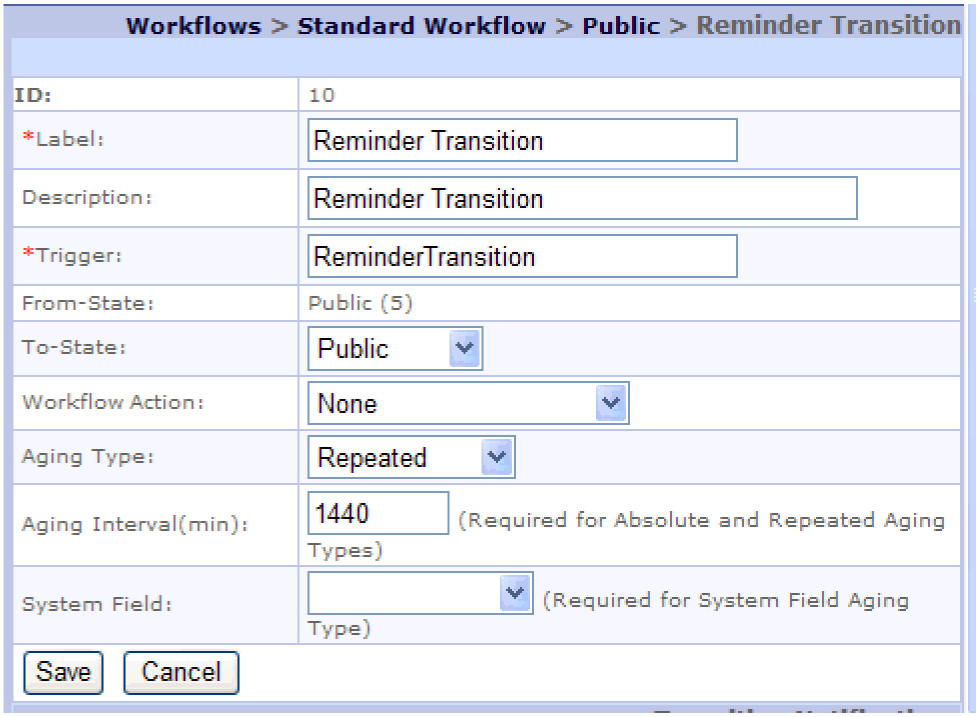
To implement the Reminder Transition:

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 Public 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 *Public*. 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 > Public > Reminder Transition

ID:	10
*Label:	Reminder Transition
Description:	Reminder Transition
*Trigger:	ReminderTransition
From-State:	Public (5)
To-State:	Public
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)

Figure 42: Implementing a Repeating Transition

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

Content Explorer returns to the Public State page. Use the procedure described in "***Implementing Notifications*** (see page 46)" 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, during modeling and design, we decided that the Approve Transition from the Review State to the Pending State will require approvals from both the Editor Role and the Web Admin Role.

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.

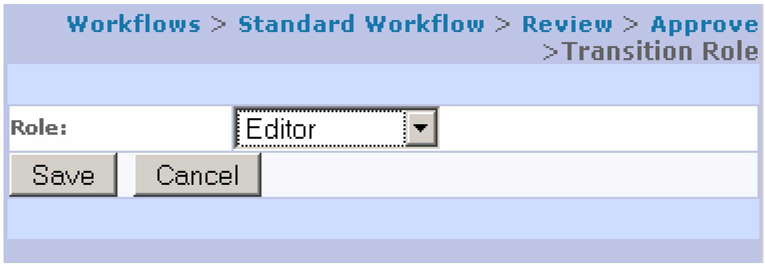
Content Explorer displays the Edit Transition page for the Approve Transition.

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.



Workflows > Standard Workflow > Review > Approve >Transition Role

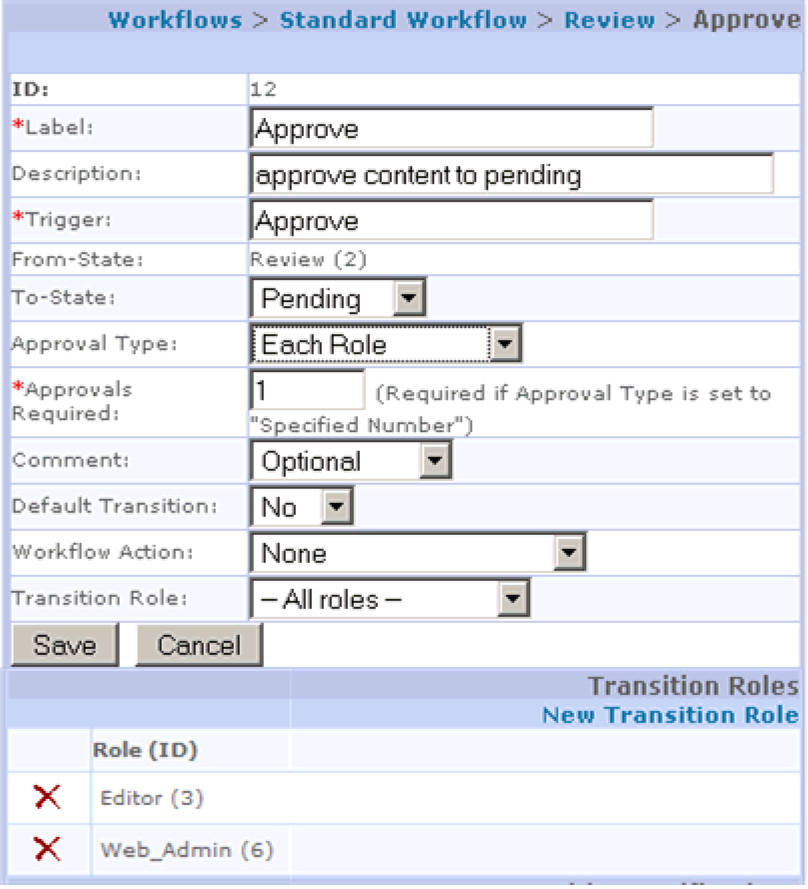
Role:	Editor

Figure 43: 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 Web Admin Role.

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



Workflows > Standard Workflow > Review > Approve

ID:	12
*Label:	Approve
Description:	approve content to pending
*Trigger:	Approve
From-State:	Review (2)
To-State:	Pending
Approval Type:	Each Role
*Approvals Required:	1 (Required if Approval Type is set to "Specified Number")
Comment:	Optional
Default Transition:	No
Workflow Action:	None
Transition Role:	– All roles –

Transition Roles

New Transition Role

	Role (ID)
✗	Editor (3)
✗	Web_Admin (6)

Figure 44: 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.

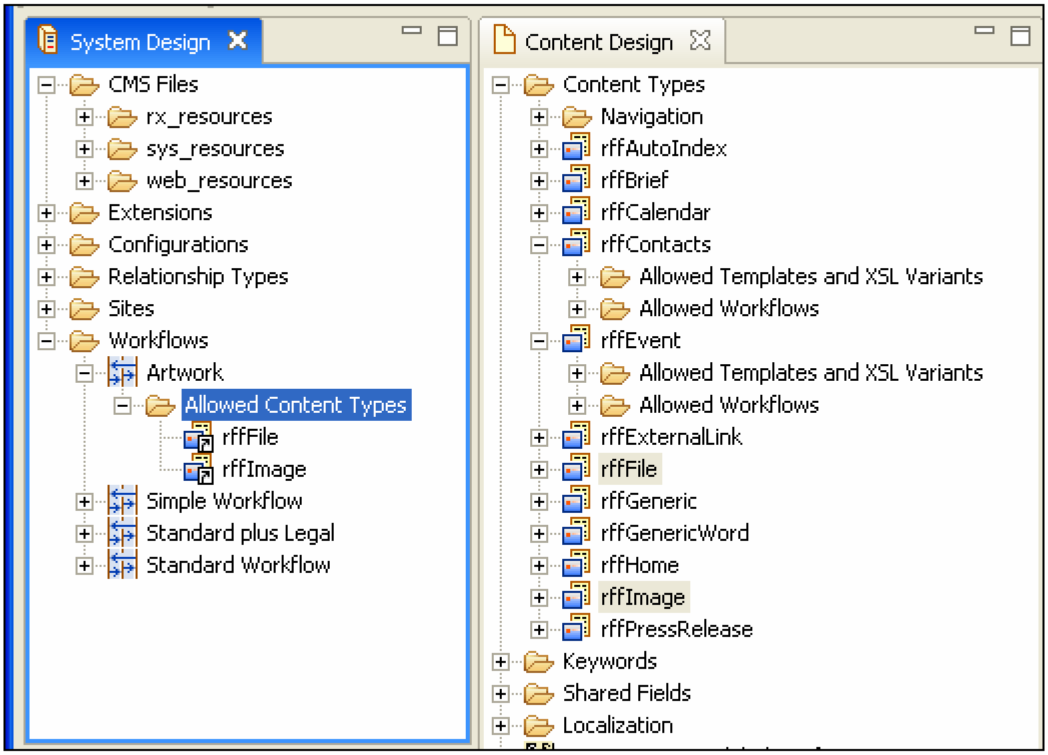
If your system already includes Content Types when you copy a Workflow, and you want to associate the copied Workflow with existing Content Types, you must associate them manually.

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.



System Design

- CMS Files
  - rx_resources
  - sys_resources
  - web_resources
- Extensions
- Configurations
- Relationship Types
- Sites
- Workflows
  - Artwork
    - Allowed Content Types
      - rffFile
      - rffImage
  - Simple Workflow
  - Standard plus Legal
  - Standard Workflow

Content Design

- Content Types
  - Navigation
  - rffAutoIndex
  - rffBrief
  - rffCalendar
  - rffContacts
    - Allowed Templates and XSL Variants
    - Allowed Workflows
  - rffEvent
    - Allowed Templates and XSL Variants
    - Allowed Workflows
  - rffExternalLink
  - rffFile
  - rffGeneric
  - rffGenericWord
  - rffHome
  - rffImage
  - rffPressRelease
- Keywords
- Shared Fields
- Localization

Figure 45: 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

  Define 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 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 in Rhythmyx.
- 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. (But the Site Root Folder need not be an immediate child of the //Sites node. Both //Sites/EnterpriseInvestments and //Sites/Investments/EnterpriseInvestments are valid Site Root Folders.)

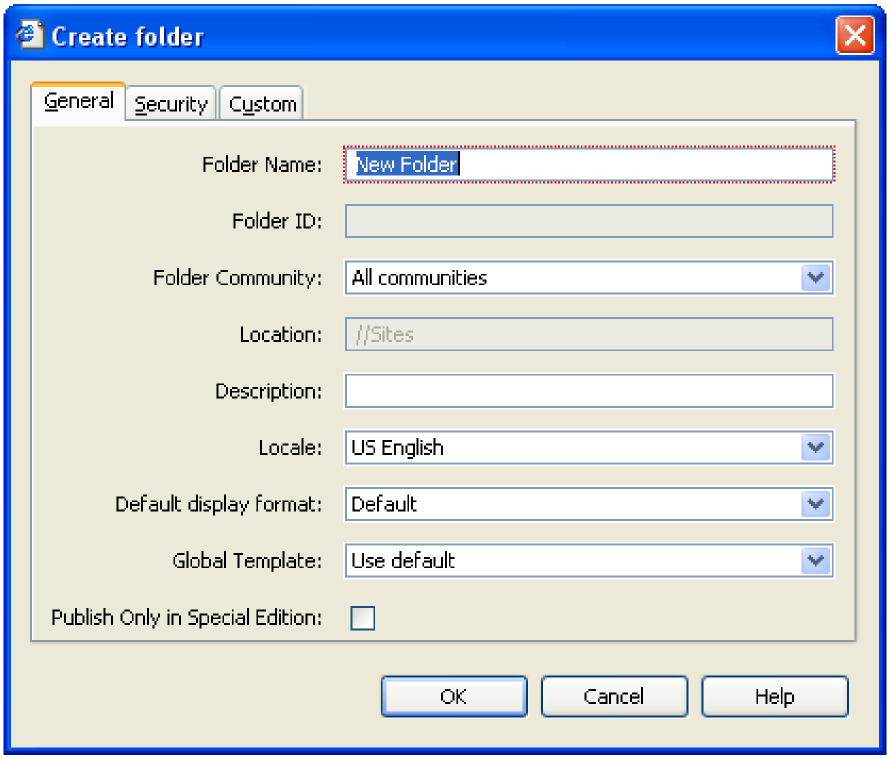
Creating the Site Root Folder is a simple procedure but the processes that follow require careful attention to be sure the publishing environment remains consistent.

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.



Folder Name: New Folder

Folder ID:

Folder Community: All communities

Location: //Sites

Description:

Locale: US English

Default display format: Default

Global Template: Use default

Publish Only in Special Edition: □

Figure 46: 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.

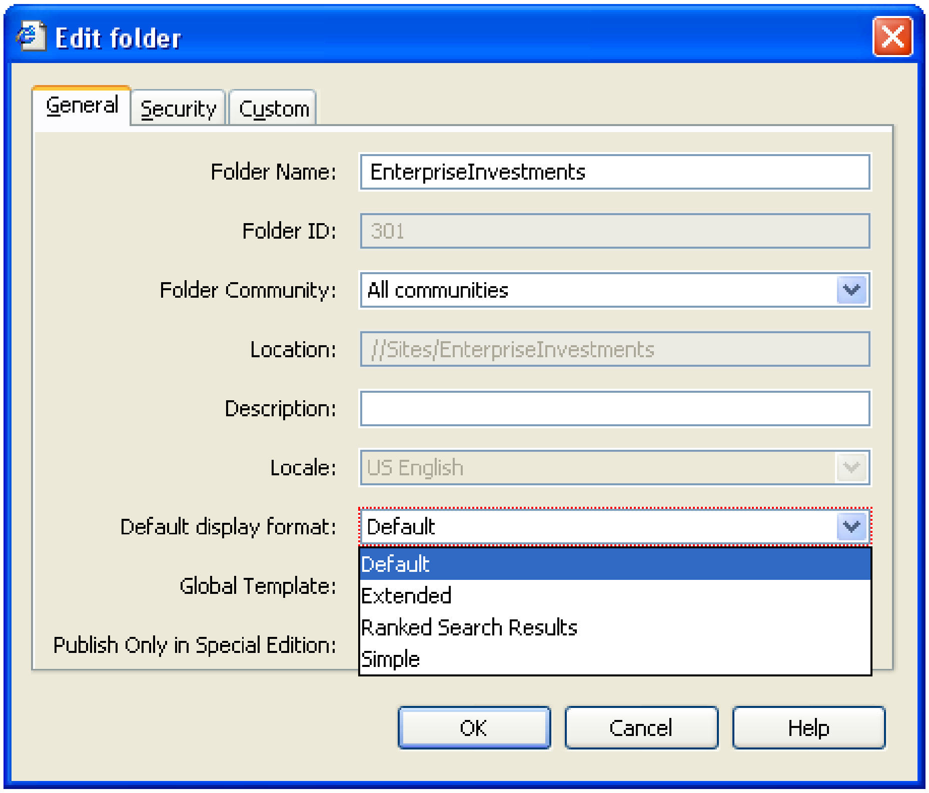
4 The **Folder ID** field is assigned a value by Rhythmyx when you have finished entering all of the Site Root folder attributes and save.

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 drop down 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 drop down list of available Display Formats for the Site. For this exercise select Default.



Folder Name: EnterpriseInvestments

Folder ID: 301

Folder Community: All communities

Location: //Sites/EnterpriseInvestments

Description:

Locale: US English

Default display format: Default

Default
Extended
Ranked Search Results
Simple

Global Template:

Publish Only in Special Edition:

Figure 47: 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. This is an outdated feature formerly used for special processing.

NOTE: You can also copy an existing Site root folder, its sub-folders, and optionally, its contents. For instructions, see the topic "Copying a Site Folder Tree" in the Content Explorer online help.

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. For the purpose of this exercise, we will assume that the EI Global Template has already been created. (See ***Implementing Global Templates*** (on page 169) for details about creating a Global Template.)

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

To register the Site with Rhythmyx:

1 Log in to Rhythmyx Content Explorer and click the Publishing Design tab.

2 Click on the *Sites* node.

The Edit and View pane displays the Site List.

3 In the Menu bar, choose *Action* > *Create Site*.

The Edit and View pane displays the Site Editor.

4 The Site Name defaults to *Site_0*. Change this value to EnterpriseInvestments.

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 Rhythmyx Path field specifies the Rhythmyx Site Folder that will be treated as the root Folder when the Site is published. Enter *//Sites/EnterpriseInvestments*. Note that the Folder name (or path) you specify must exist. If you did not create the Site Folder in Content Explorer before starting this procedure, Content Explorer will return an error when you attempt to save the Site registration. You can use the ... button to browse the //Sites node in the Navigation pane of Content Explorer to find the root location you want to use for the Site. The Site root Folder need not be a direct child of the Sites Folder. In other words, you can specify either `//Sites/EnterpriseInvestments` and `//Sites/Investments/EnterpriseInvestments`. Both are valid root paths.

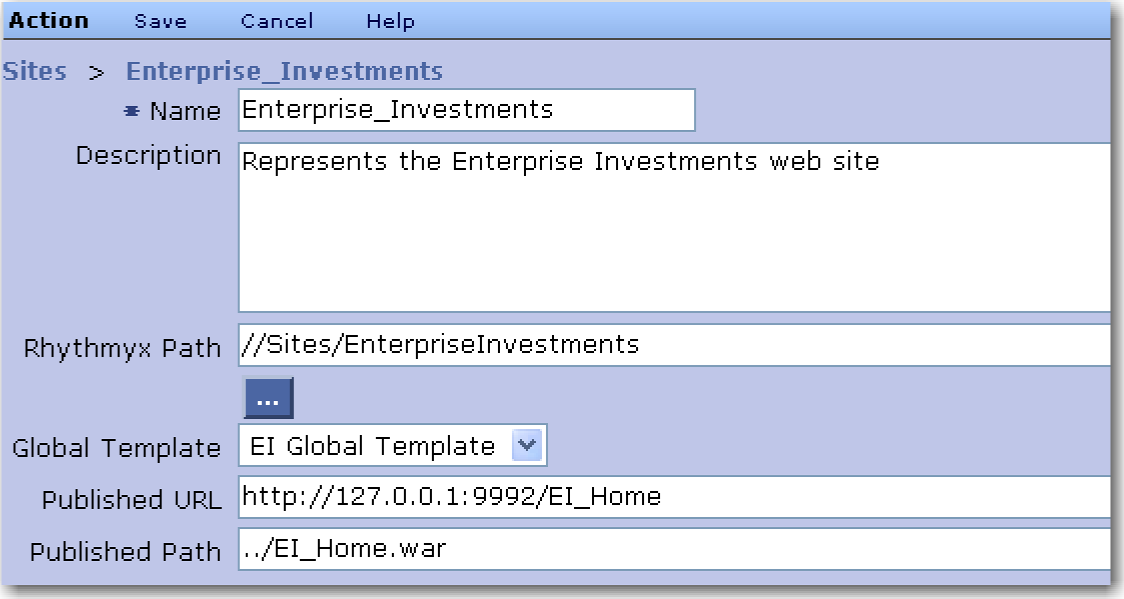
7 In the Global Template drop list, choose *EI GLobal Template*.

8 The Published URL field is optional It defines the URL entered in a browser to access the Site when it is published. Since we plan to use the File System Delivery Type for this Site, enter http://127.0.0.1:9992/EI_Home. (If you want to use one of the FTP Delivery Types for this Site, you would enter the virtual directory where the Site output would be published.)

9 The Published Path field is optional. It specifies the directory location where the files will be saved. For this example enter the following: ../EI_Home.war. If you specify a relative path, the location is relative to the application server root (<Rhythmyx root>\AppServer\server\rx\deploy\rxapp.ear); so the path ../EI_Home actually goes to the directory <Rhythmyx root>\AppServer\server\rx\deploy\EI_Home.

10 In the Menu bar, click *Save*.
The Enterprise Investments site is now registered in Rhythmyx.

The Enterprise Investments Site registration resembles the following graphic:



* Name: Enterprise_Investments
Description: Represents the Enterprise Investments web site
Rhythmyx Path: //Sites/EnterpriseInvestments
Global Template: EI Global Template
Published URL: http://127.0.0.1:9992/EI_Home
Published Path: ../EI_Home.war

Figure 48: Enterprise Investments Site Registration

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.

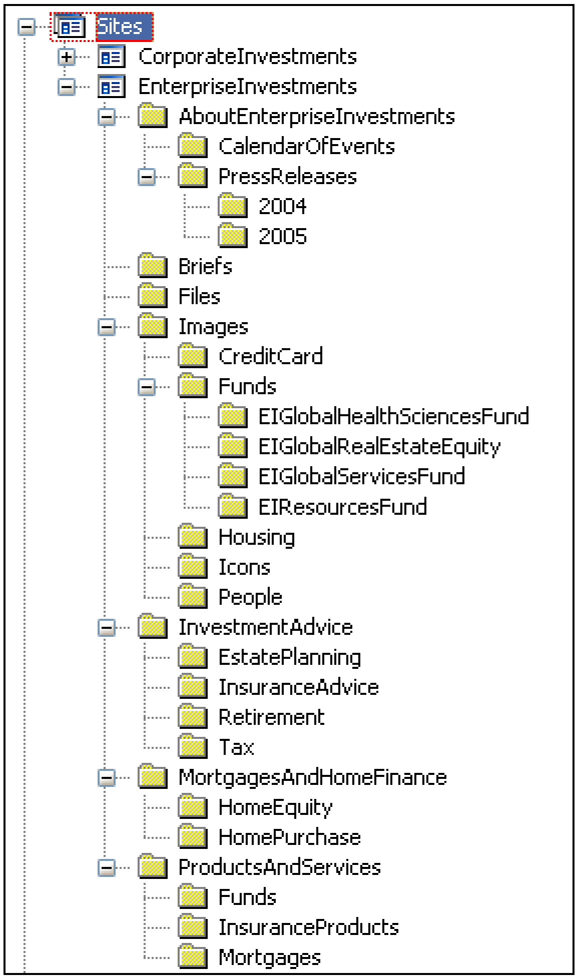


Figure 49: Site hierarchy in Content Explorer

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:

- 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 279) for more information on implementing and configuring Managed Navigation.

Adding a NavTree to the Site Hierarchy

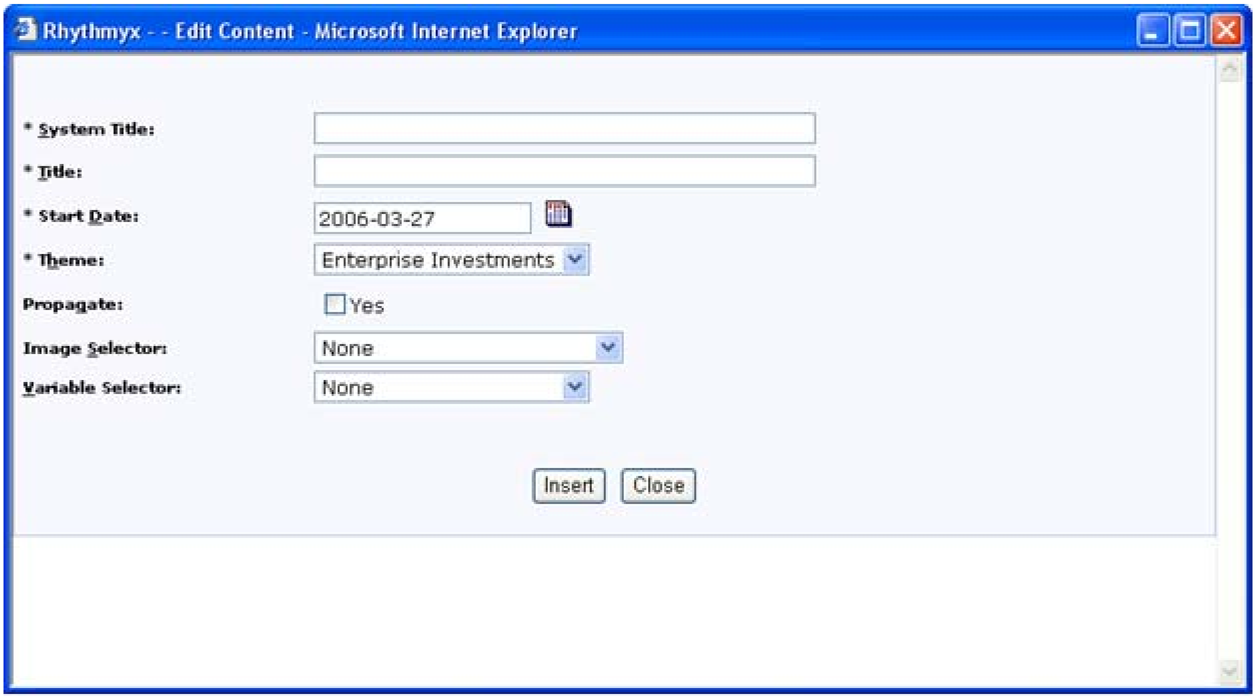
Adding a NavTree to the root of the Site hierarchy is the first step in setting up Managed Navigation.

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.



* System Title:

* Title:

* Start Date: 2006-03-27

* Theme: Enterprise Investments

Propagate: □ Yes

Image Selector: None

Variable Selector: None

Figure 50: NavTree 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 drop down 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.

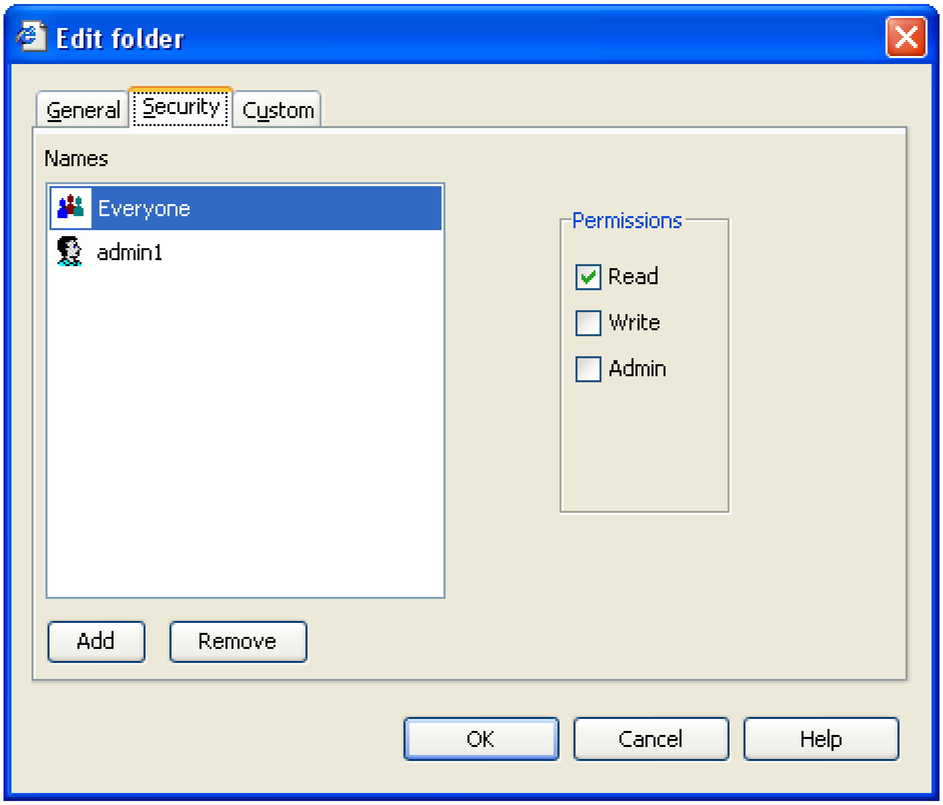
11 Click the [**Insert**] button.
This will add the NavTree to the Repository.

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.



Names

- Everyone
- admin1

Permissions

☑ Read

□ Write

□ Admin

Figure 51: 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.

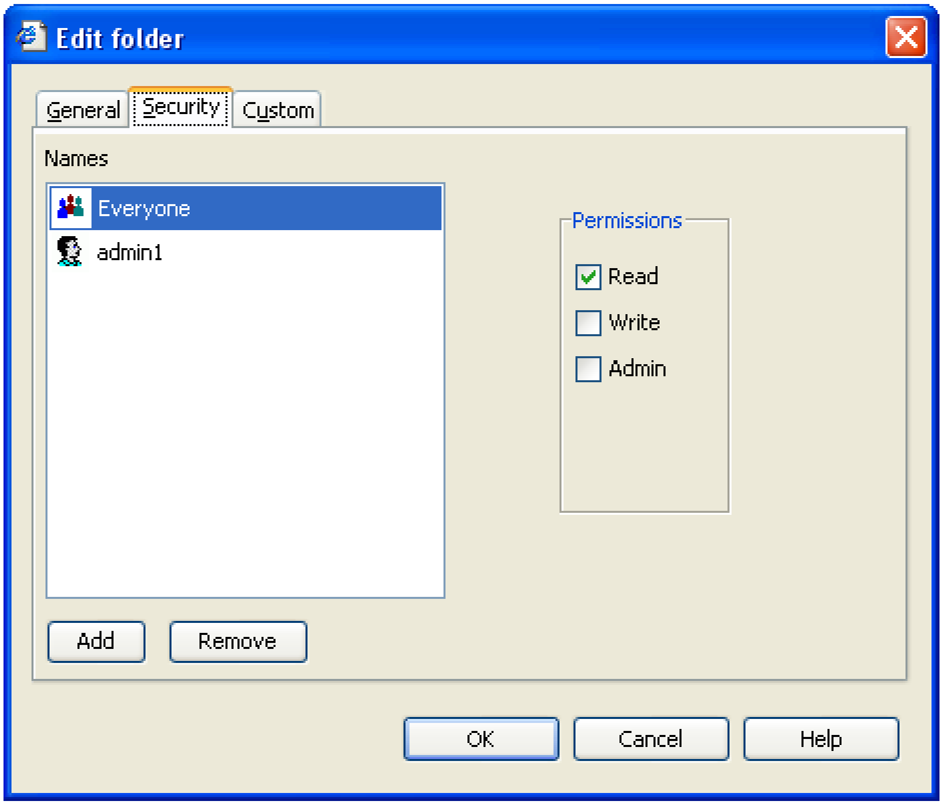


Figure 52: Edit folder dialog

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.

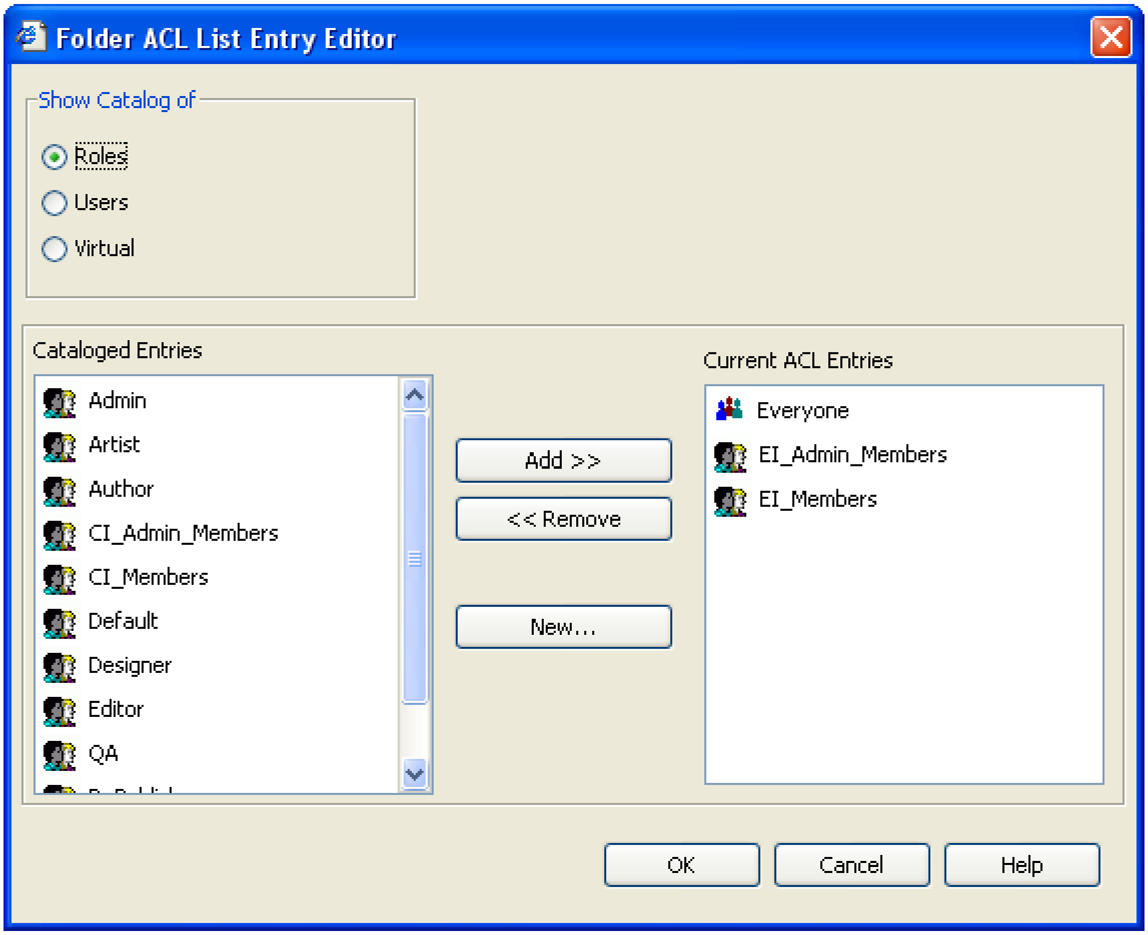


Figure 53: Folder ACL

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

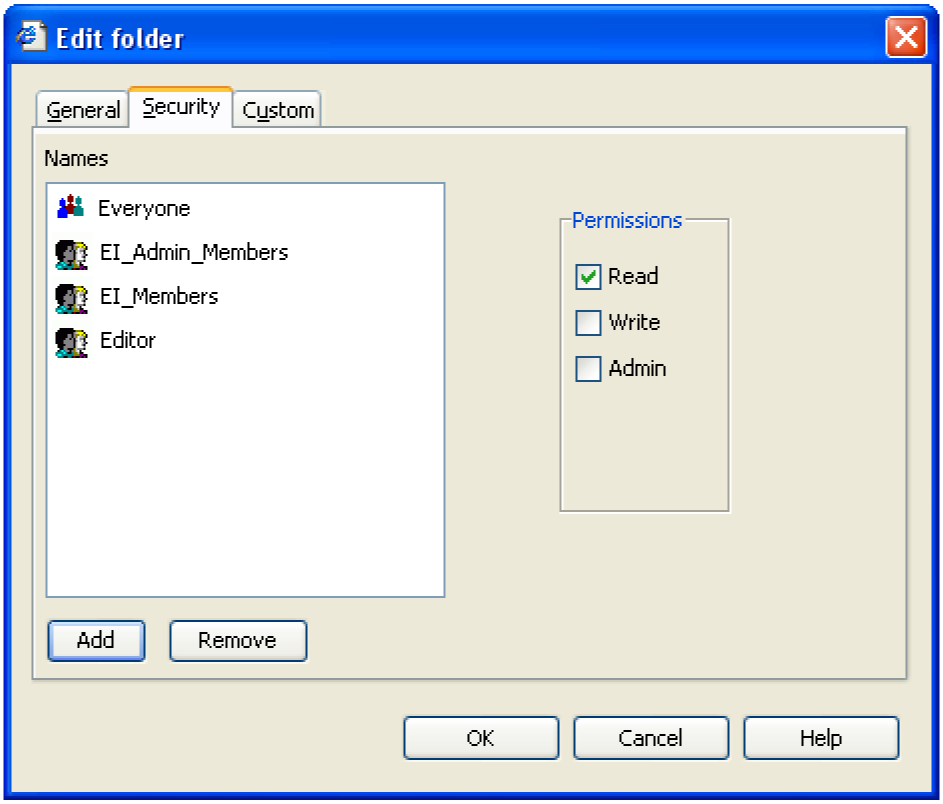
Users - lists all system-defined users in the Cataloged Entries section of the dialog

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.


Edit folder

General | Security | Custom

Names

- Everyone
- EI_Admin_Members
- EI_Members
- Editor

Permissions

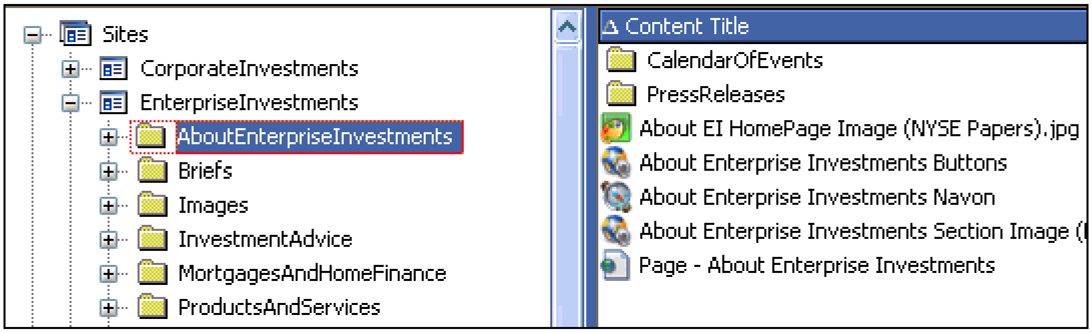
☑ Read
□ Write
□ Admin

Figure 54: 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.



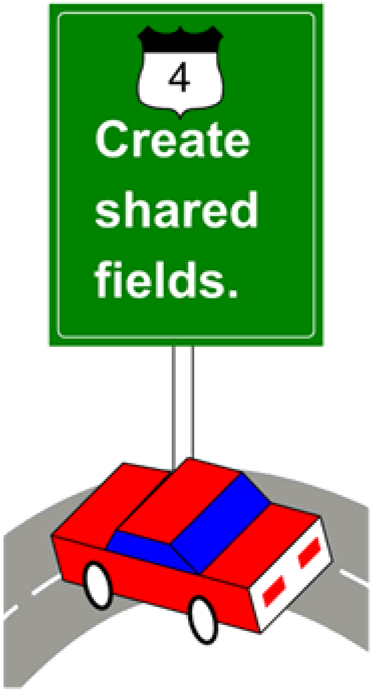
Sites
CorporateInvestments
EnterpriseInvestments
AboutEnterpriseInvestments
Briefs
Images
InvestmentAdvice
MortgagesAndHomeFinance
ProductsAndServices

Content Title
CalendarOfEvents
PressReleases
About EI HomePage Image (NYSE Papers).jpg
About Enterprise Investments Buttons
About Enterprise Investments Navon
About Enterprise Investments Section Image (
Page - About Enterprise Investments

Figure 55: Folder visibility

CHAPTER 5

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 215). 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 83) and *sharedimage (see page 84) 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 448) 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**
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.

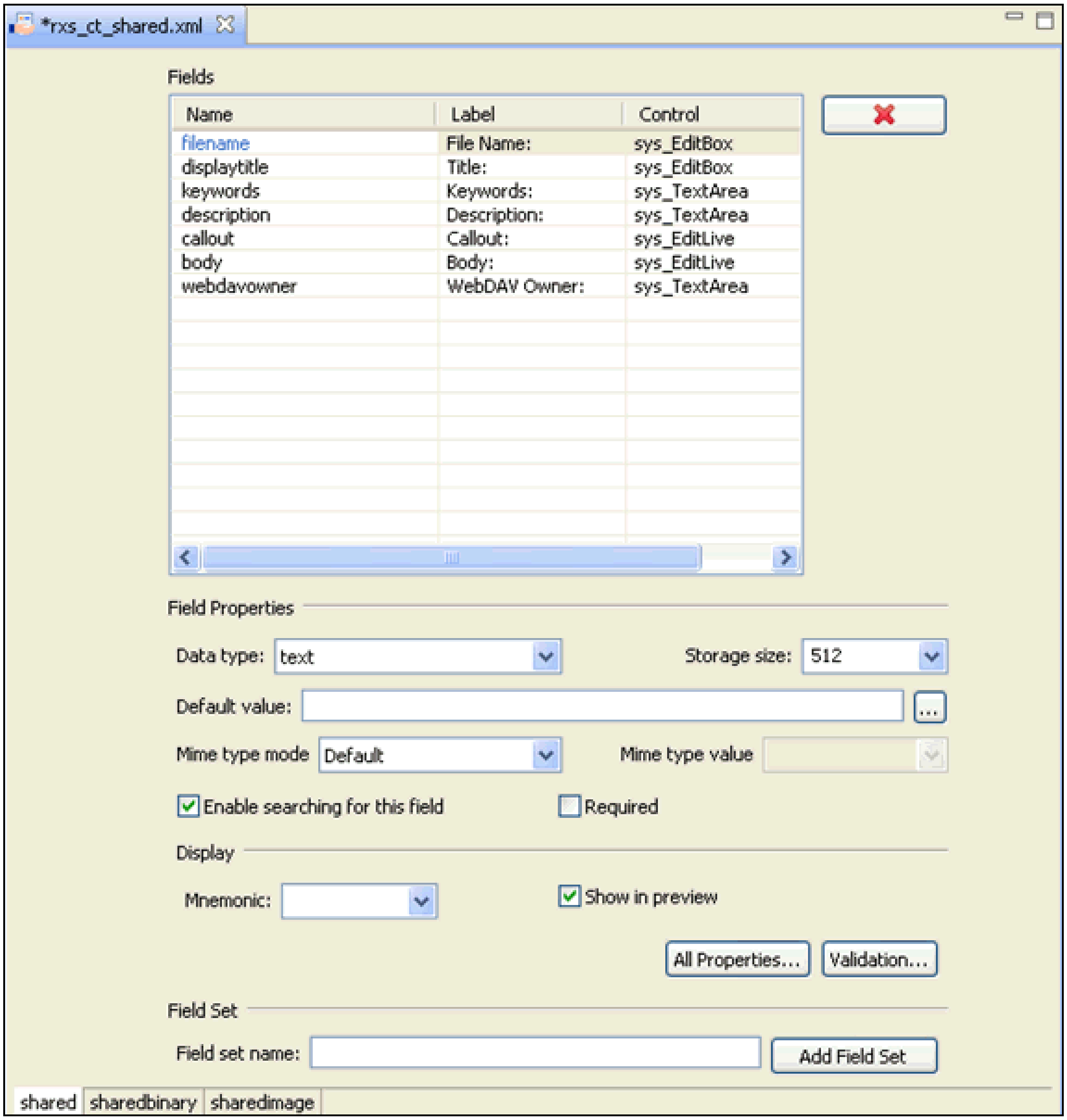
The *img1_filename*, *img1_size*, *img1_type*, *img1_ext* and *img2_filename*, *img2_size*, *img2_type*, and *img2_ext* fields in the sharedimage field set store metadata that is automatically extracted when the sys_file control uploads the *img1* or *img2* file. Other metadata is also extracted, and 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_imageInfoExtractor extension which is included as a Java extension (a Java plugin) when the sys_file control is used. This extension requires certain names for fields in which it stores data. See the topic *sys_imageInfoExtractor* in the *Rhythmyx Technical Reference* for more information about this extension. See the topic ***The Image Content Editor*** (see page 246) in the chapter *Creating Content Types* for more information about using sys_imageInfoExtractor with a Content Type.

The ***sharedimage Field Set specification*** (see page 449) 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.



Name	Label	Control
filename	File Name:	sys_EditBox
displaytitle	Title:	sys_EditBox
keywords	Keywords:	sys_TextArea
description	Description:	sys_TextArea
callout	Callout:	sys_EditLive
body	Body:	sys_EditLive
webdavowner	WebDAV Owner:	sys_TextArea

Data type: text

Storage size: 512

Default value:

Mime type mode: Default

Mime type value

☑ Enable searching for this field

□ Required

Mnemonic:

☑ Show in preview

Field set name:

Figure 56: Field and Field Set Editor

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

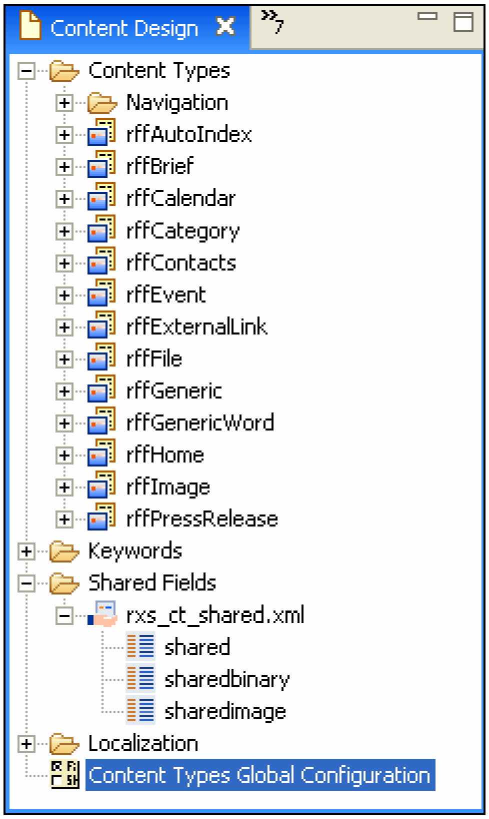


Figure 57: Content Design view

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 215).

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 87) will cover common choices for frequently used fields, while the topic ***Implementing the "sharedimage" Field Set*** (see page 98) 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 95) 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 448). You are also introduced to the 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 than 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*.

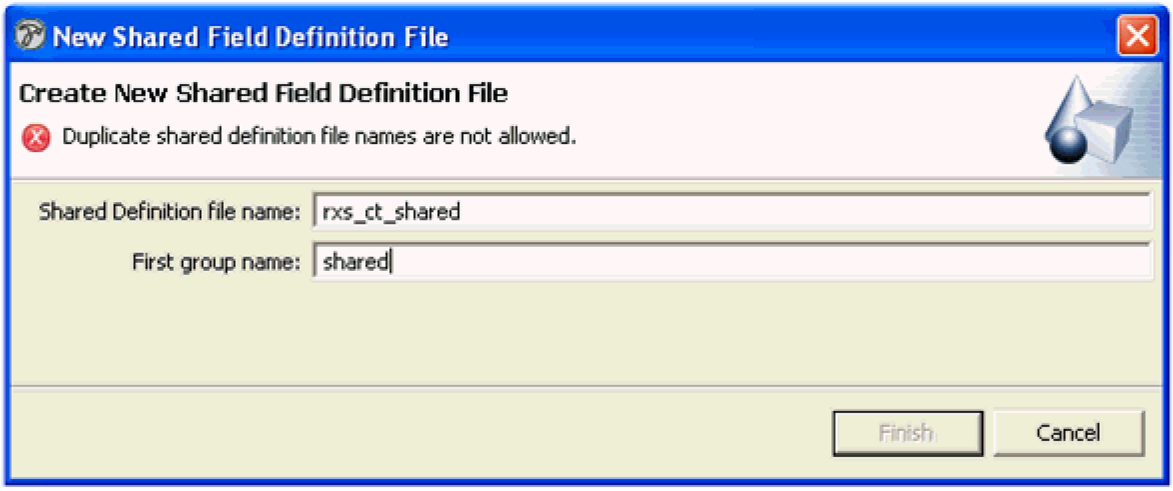
The Select a Wizard dialog opens.

2 Choose *Shared Field File* and click [**Next**].

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*.



New Shared Field Definition File

Create New Shared Field Definition File

Duplicate shared definition file names are not allowed.

Shared Definition file name: rxs_ct_shared

First group name: shared

Figure 58: New Shared Field Definition File wizard

5 Click [**Finish**].

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.

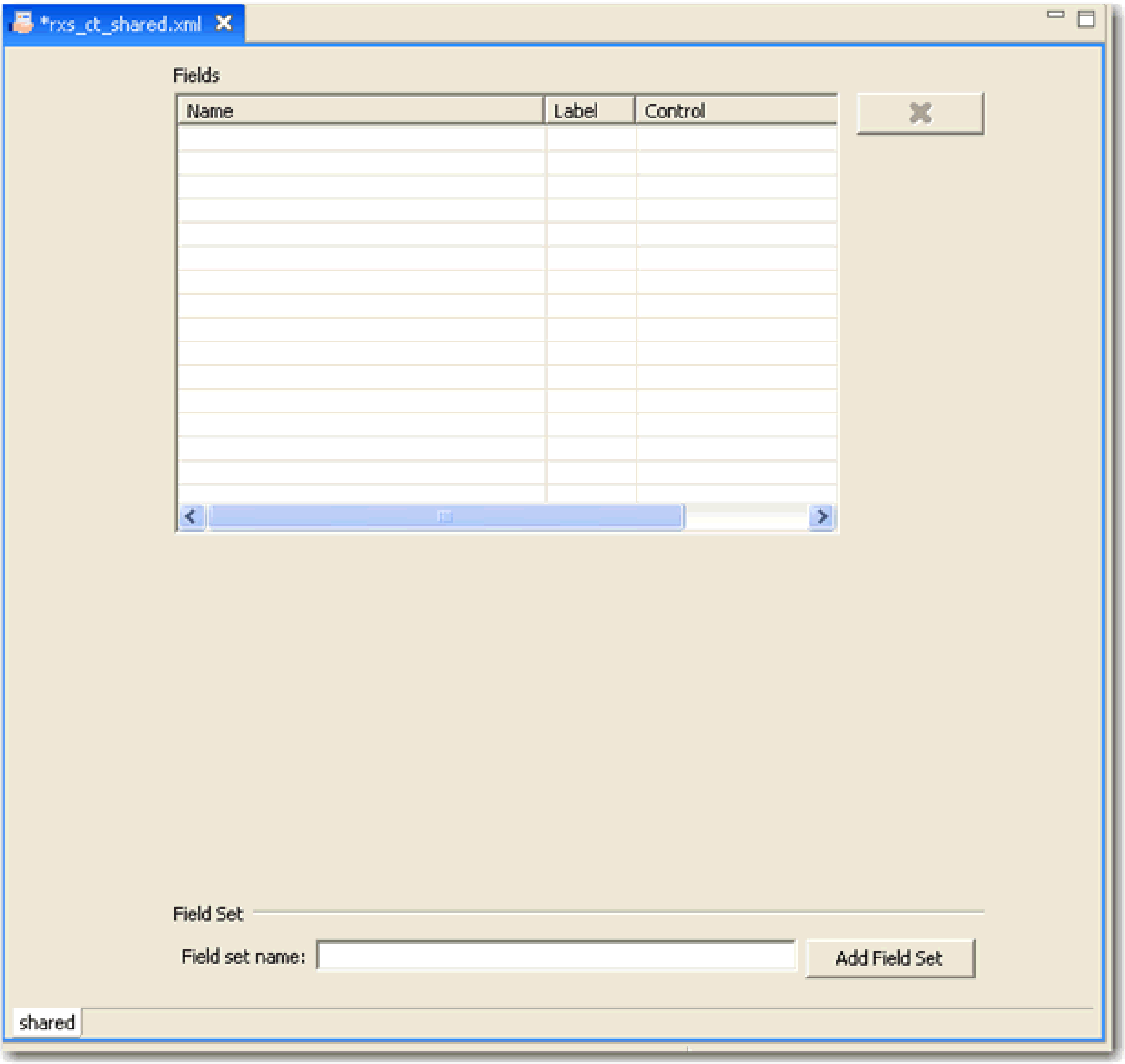


Figure 59: Shared Field Definition File editor

6 Begin by entering the *displaytitle* field.

d) 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:*.

e) 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

f) Click on the *displaytitle* row to display **Field Properties** in the lower part of the editor.

g) In **Data Type**, leave the default value of *text.*

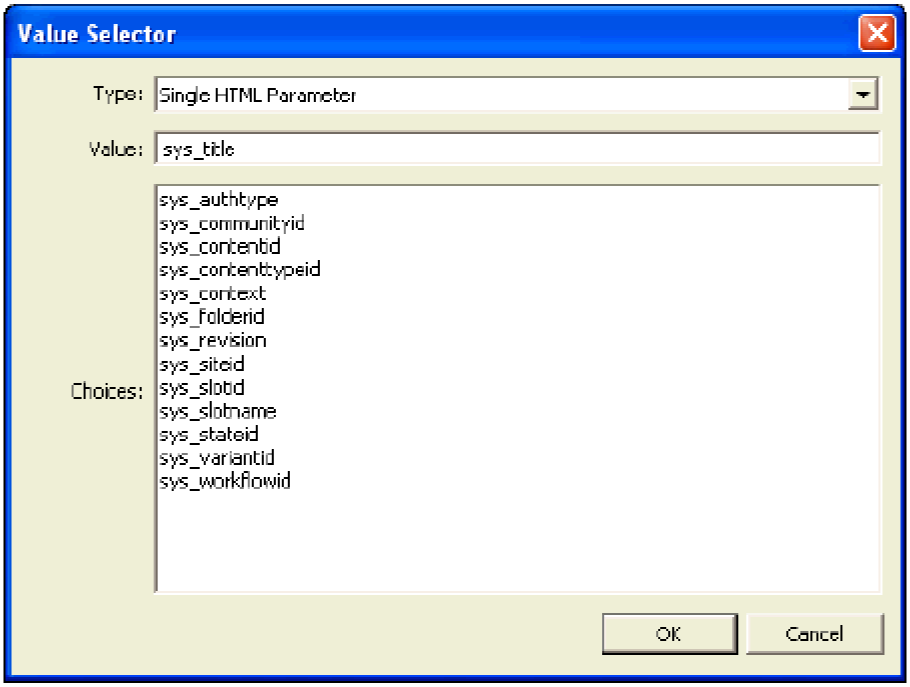
h) In **Storage size**, change the default value of 50 to 512. Although it is unlikely that anyone will enter a value that uses this amount of space, it takes into account foreign characters that may require additional bytes and databases that may require extra space to store characters.

i) 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.

- 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.



Value Selector

Type: Single HTML Parameter

Value: sys_title

Choices:
sys_authtype
sys_communityid
sys_contentid
sys_contenttypeid
sys_context
sys_folderid
sys_revision
sys_siteid
sys_slotid
sys_slotname
sys_stateid
sys_variantid
sys_workflowid

Figure 60: 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*.

Leave Mime type mode as Default, to allow the system to specify the mime type for the field. Note that when you choose *Default*, Mime type value remains grayed out. The system chooses Mime type value based on the Data Type and Storage Size. In this case Mime type value is *text/plain*.

j) Leave Enable searching for this field checked because users will want to search on the title of the field.

k) Check Required to make *displaytitle* a required field.

l) 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*.

m) Leave Show in preview checked so that users can see the *displaytitle* field when they preview templates of Content Items that contain it.

n) Do not click the [**All Properties**] button. You will leave the default values of the properties that it accesses.

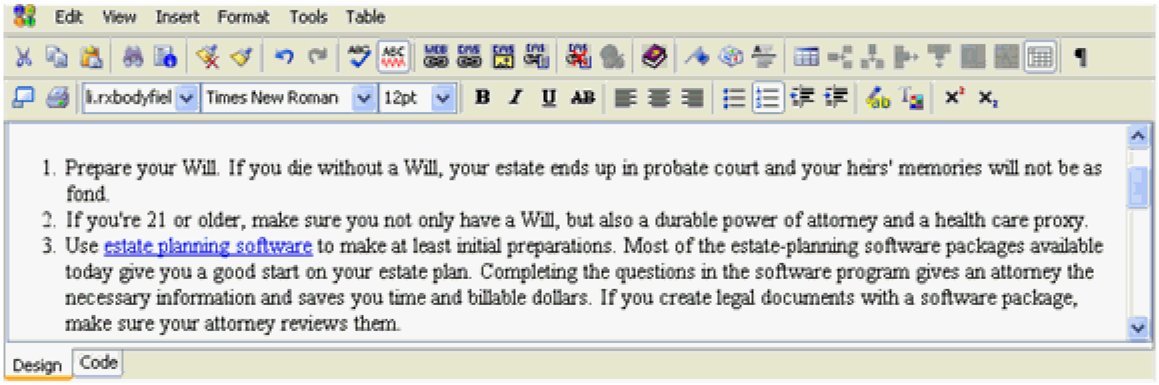
7 Next enter the *body* field.

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.

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.

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.



li.rxbodyfiel | Times New Roman | 12pt

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.

Figure 61: sys_EditLive control

Note that any number of fields in a Content Type can use the sys_EditLive editor, but when users view the fields in the content editor, only one sys_EditLive field can be active and edited at a time. When a user first opens a content editor including a sys_EditLive field, the user must click on the field to enable it. Once the user clicks on another sys_EditLive field, the first sys_EditLive field becomes disabled.

c) Click on the row for the *body* field to make sure you are displaying the **Field Properties** for this field.

d) 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) In **Storage size**, change the value to *max*. The value *max* indicates that the storage size is the maximum allowed for 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, 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.

Leave **Mime type mode** as *Default*, to allow the system to specify the mime type for the field. Note that when you choose *Default*, **Mime type value** remains grayed out. The system chooses **Mime type value** based on the **Data Type** and **Storage Size**. In this case **Mime type value** is *text/html*.

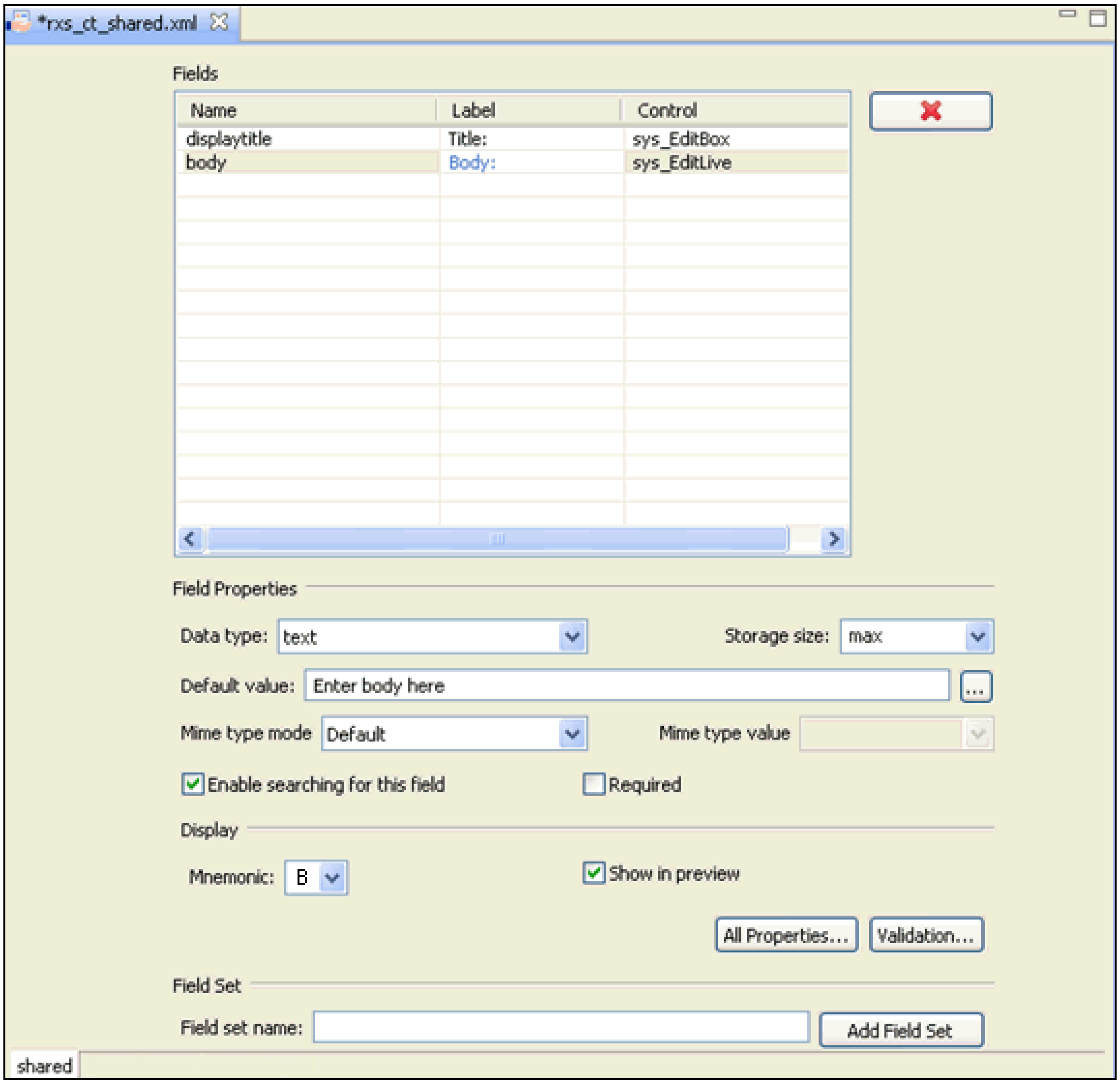
g) Leave **Enable searching for this field** checked.

h) Leave **Required** unchecked, so that *Body* is not a required field.

i) In Mnemonic, choose a key for accessing the field. Since the *body* field begins with the letter B, choose *B*.

j) 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:



Name	Label	Control
displaytitle	Title:	sys_EditBox
body	Body:	sys_EditLive

Data type: text

Storage size: max

Default value: Enter body here

Mime type mode: Default

Mime type value:

☑ Enable searching for this field

☐ Required

Mnemonic: B

☑ Show in preview

Field set name:

Figure 62: 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.

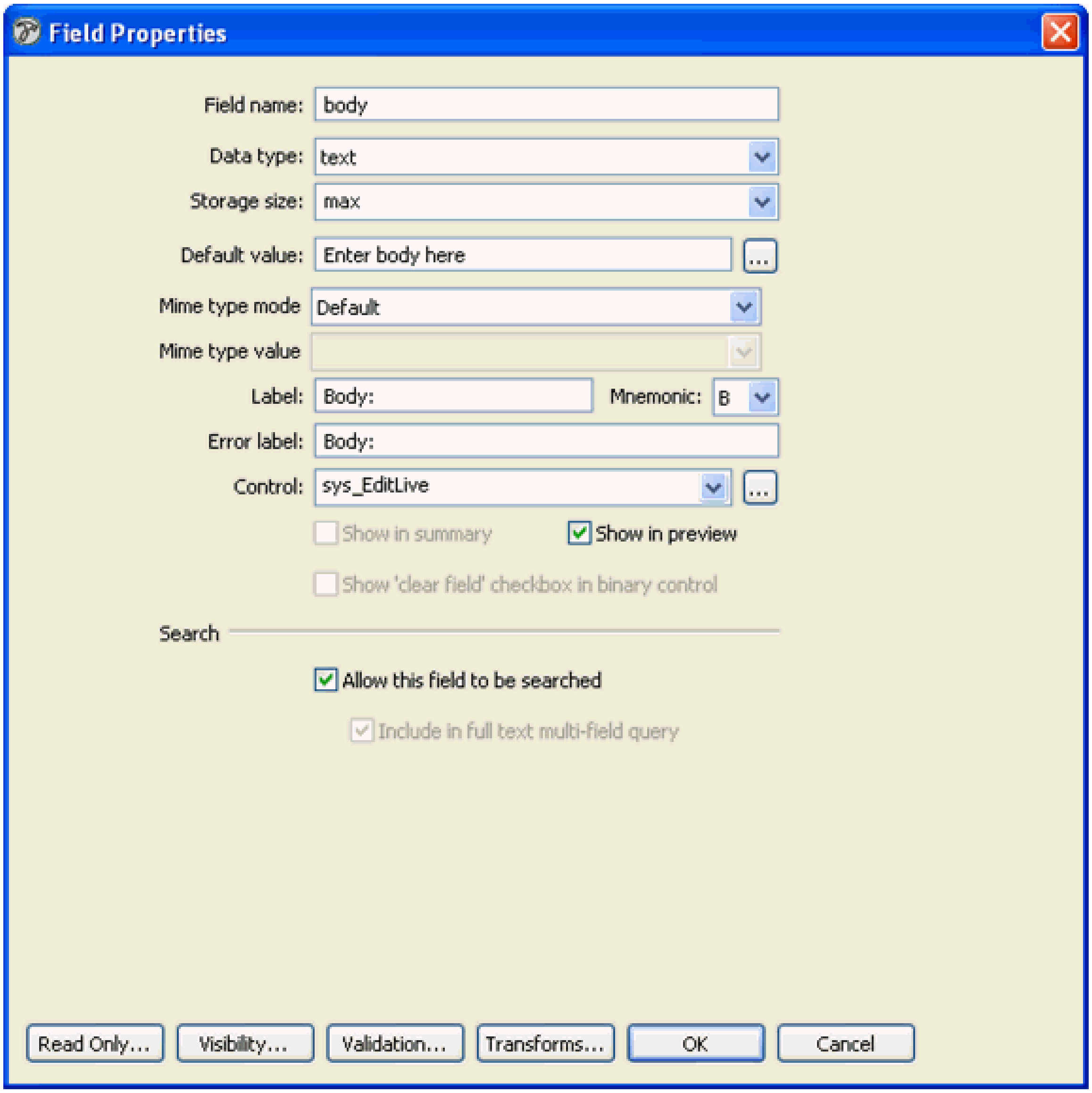
a) Leave the default value of *Body:* in Error label. Rhythmyx will concatenate any validation errors with the Error Label value and display them in the Content Editor.

b) 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.

c) Show "clear field" check box in binary control is disabled because the *body* field does not use a control that uploads binary files.

d) Include in full text multi-field query is checked to allow users to use the *body* field when creating searches in Content Explorer.

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



Field Properties

Field name: body

Data type: text

Storage size: max

Default value: Enter body here

Mime type mode Default

Mime type value

Label: Body:

Mnemonic: B

Error label: Body:

Control: sys_EditLive

□ Show in summary

☑ Show in preview

□ Show 'clear field' checkbox in binary control

Search

☑ Allow this field to be searched

☑ Include in full text multi-field query

Figure 63: Body Field Properties

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 448). 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 105).

12 Add the *doc_type* field in the next available row. Follow the instructions in the topic ***Implementing a List Control*** (see page 95).

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 98) 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 (if field is created in a Content Type): Local

Data type: Text

Storage size: 50

Mime type mode: Default

Enable searching for this field: unchecked

Required: unchecked

Show in preview: 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. (NOTE: Field names cannot include an initial lowercase letter followed by an uppercase letter. In other words aBcd is not a valid field name. abcd or ABCD would be valid field names.) 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:

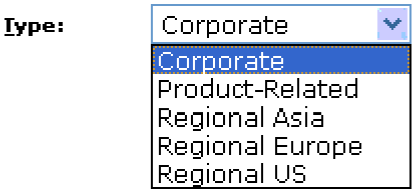


Figure 64: sys_DropDownSingle control

3 After choosing the drop-down single control, you must populate it with choices that a user can choose from.

a) Click [...] beside the control to open the Control Properties editor.

b) Click the Choices tab.

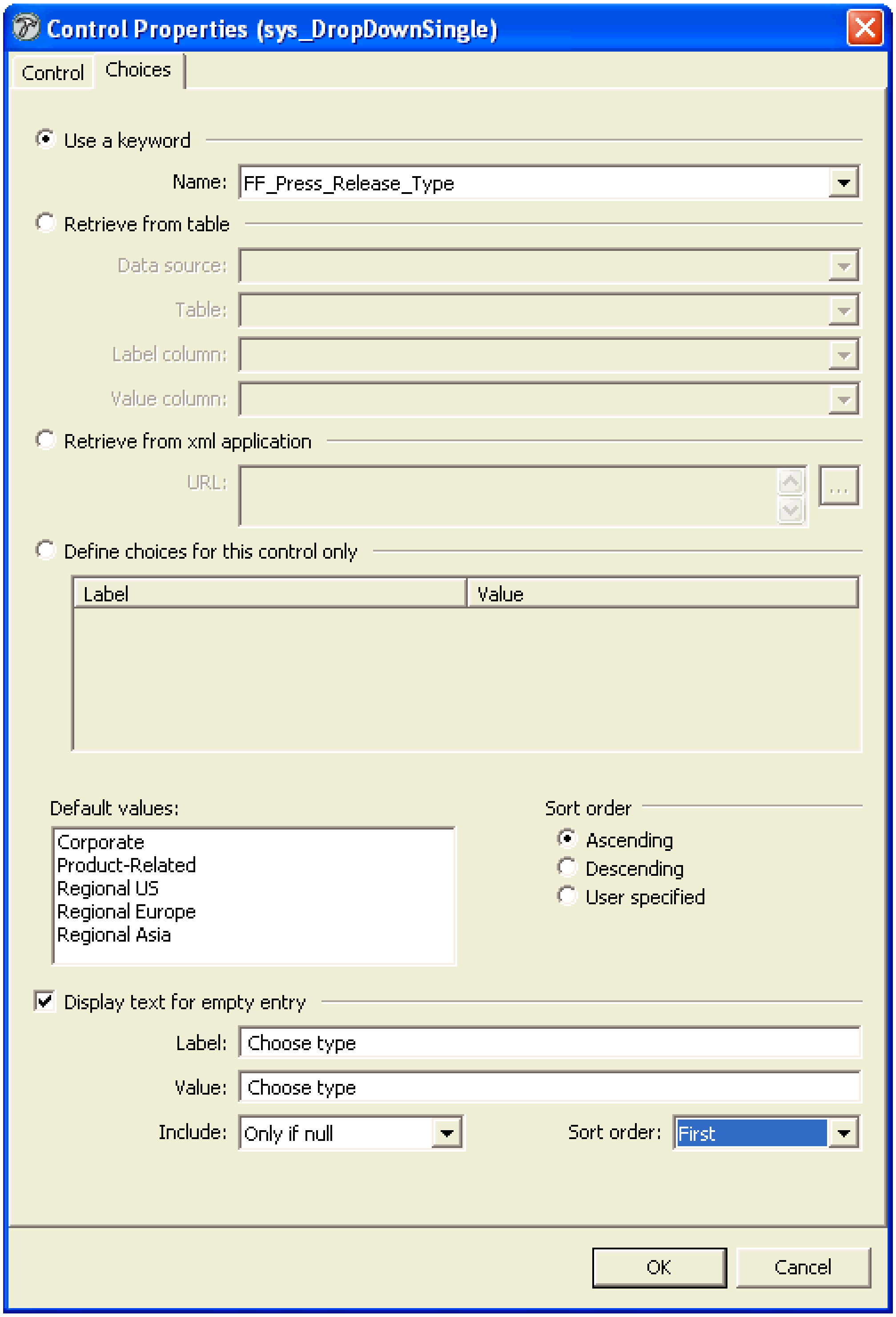
c) To use a Keyword's choices in a drop down control, leave the **Use a Keyword** radio button selected and choose *FF Press Release Type* in the **Name** drop list.

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

Note: See the section *Creating and Using Keywords* (see page 291) 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.

d) 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: FF_Press_Release_Type

○ 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: Choose type

Value: Choose type

Include: Only if null

Sort order: First

Figure 65: 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.

Leave Mime type mode as *Default*. The system chooses Mime type value based on the Data Type and Storage Size. In this case Mime type value is *text/plain*.

7 Leave Enable Searching for this field checked, since users may want to search on the value of the region.

8 Leave Required as unchecked.

9 Since users will have to manually choose a value for the field, choose the Mnemonic *T* from the drop list.

10 Check Show in preview so that users can see the region in any preview templates that include it.

11 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.

12 To add the *sharedimage* field set to the rxs_ct_shared.xml shared field object, do not close the editor. Instead proceed to the next section, ***Implementing the "sharedimage" Field Set*** (see page 98).

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.

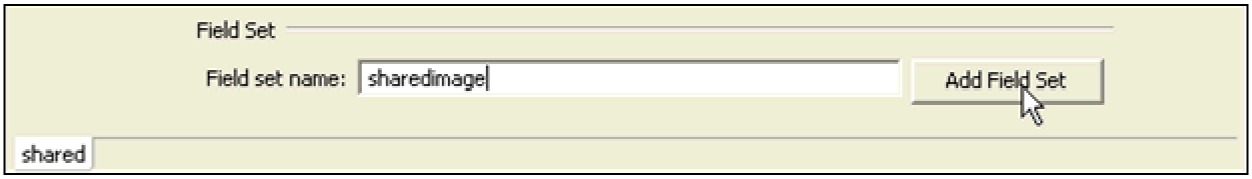
This topic focuses on the entry of the *img1* and *img1_filename* fields to demonstrate when to use the sys_File control, and the options Treat Data as Binary, and Show "clear field" checkbox in binary control as well as others. After entering the *img1* and *img1_filename* fields, you are instructed 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 105).
- 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**].



Field Set

Field set name: sharedimage

Figure 66: Adding another field set

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

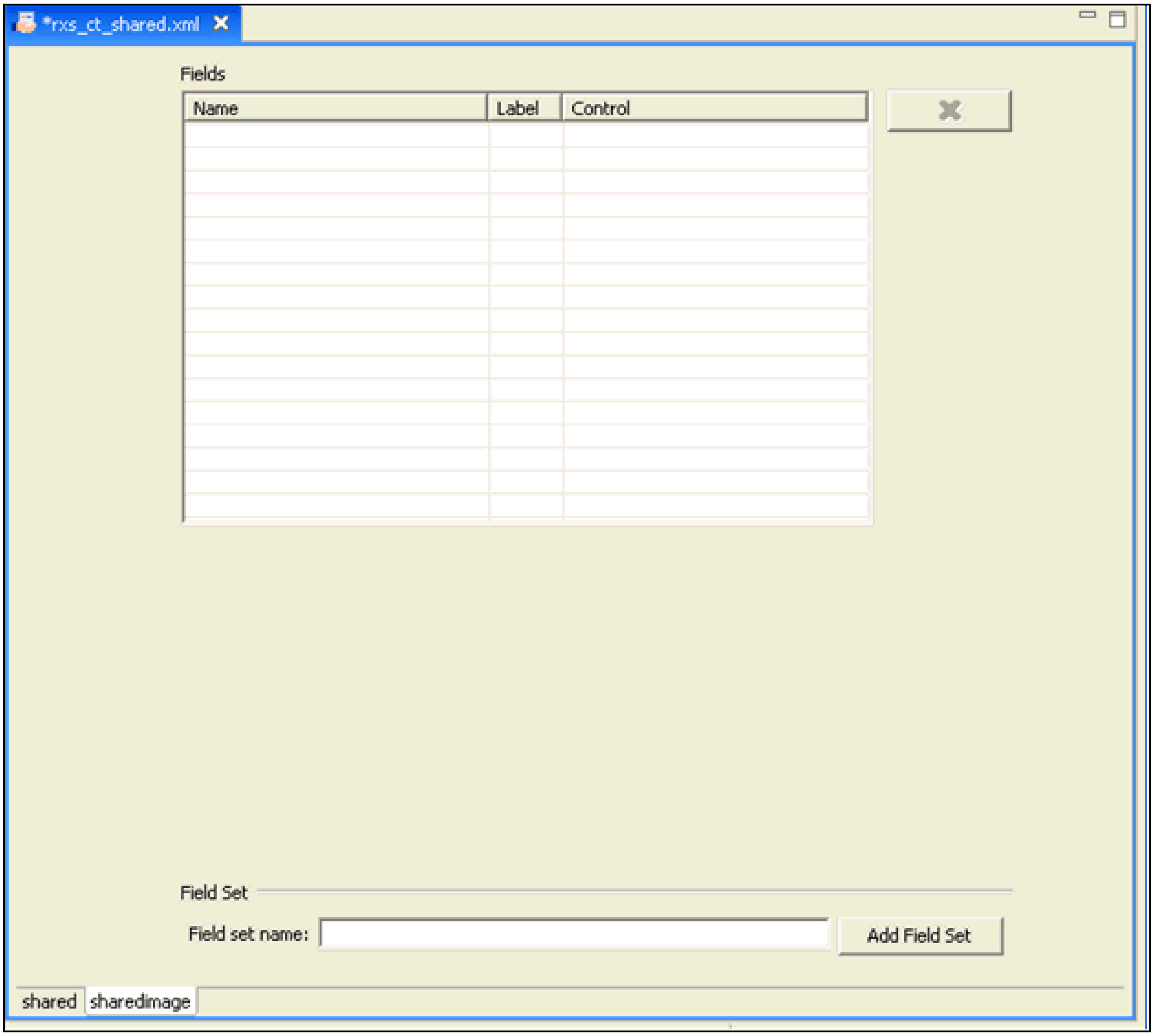


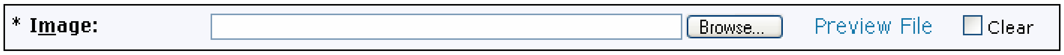
Figure 67: sharedimage field set

2 Now enter the fields specified in the topic *sharedimage* ***Field Set specification*** (see page 449).

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.



* Image: Browse... Preview File □ Clear

Figure 68: sys_File control

In FastForward, Content Types with fields that use the sys_File control include the sys_FileInfo and sys_ImageInfoExtractor extensions as 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 only use sys_ImageInfoExtractor because it encompasses the functionality of sys_FileInfo. The metadata that it extracts includes file name, Mime type, character length, file encoding, file height, and file width. It returns 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 449) for names of fields that these extensions use in the *sharedimage* field set.

Note that sys_ImageInfoExtractor 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 Editor if they are useful for users to see or for processing. See *sys_ImageInfoExtractor* in the *Rhythmyx Technical Reference* for more information and other metadata it 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 the Mime type mode, which is automatically filled in, as *Default.* In this case, the system sees that the Data Type is binary, and maps the uploaded file's extension to a mime type. The system fills the Mime type value with the mime type that mapped to the extension.

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) Check Required since the image file is the main content of the item.

h) Choose the Mnemonic *I* for the field.

i) Leave Show in Preview checked, so that the image is visible in previews.

j) Click [**All Properties**] to set additional properties for the field.

The Field Properties dialog opens.

k) Leave the default value of *Image:* in Error label. Rhythmyx will concatenate any validation errors with the Error Label value and display them in the Content Editor.

l) 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.

m) The search fields are unchecked because you unchecked Enable searching for this field in the previous dialog. Leave the search fields unchecked.

n) 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 the Mime type mode, which is automatically filled in, as *Default.* The system sets the Mime type value according to the Data Type and Storage size. In this case, Rhythmyx automatically fills the Mime type value with *text/plain* but does not display it on the screen.

f) Leave Enable searching for this field checked, since advanced users may want to search on the filename of the image.

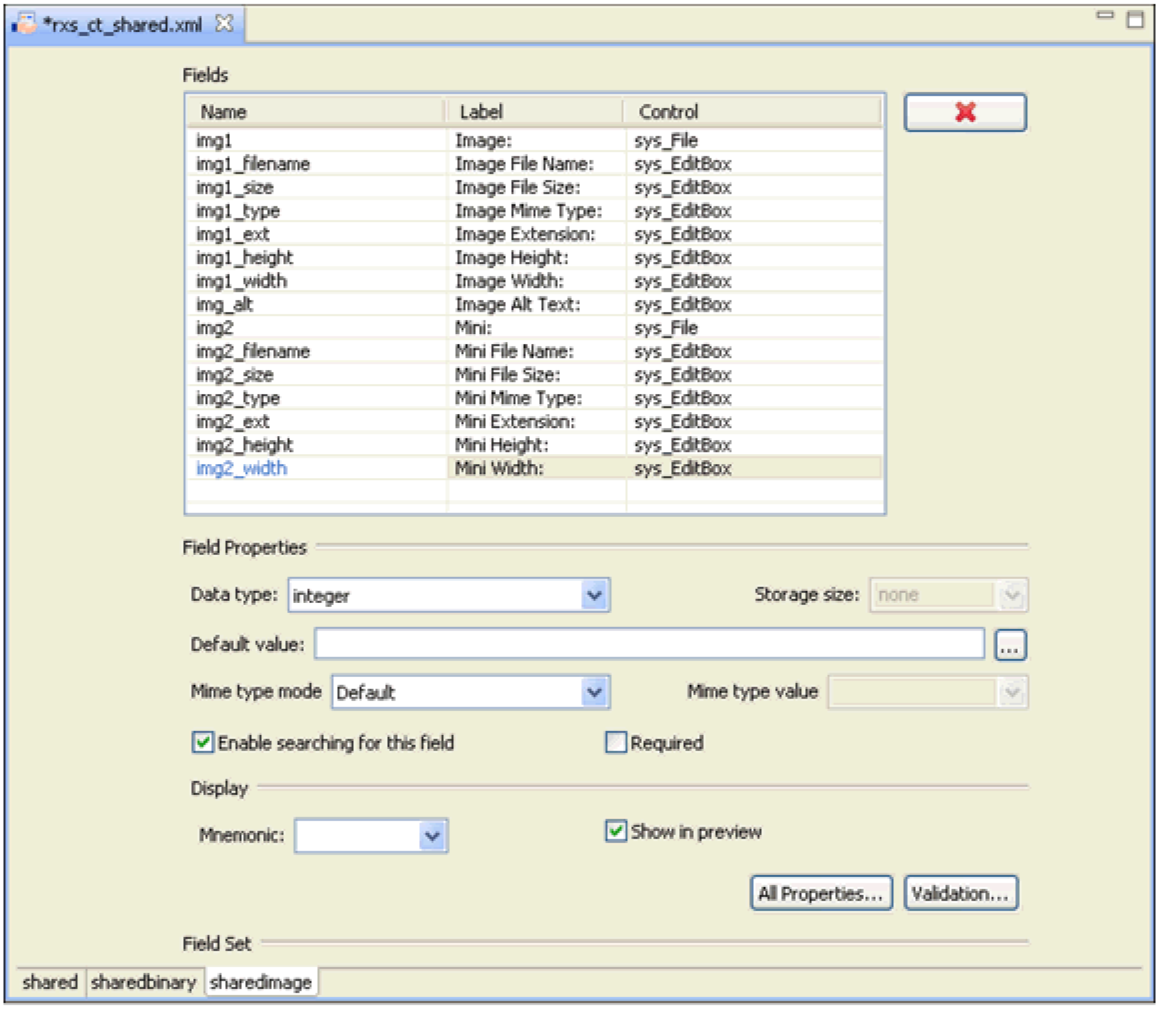
g) Check Required since the field should be filled automatically when the file is uploaded.

h) 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.

i) Leave Show in Preview checked so that the field is shown in previews of Templates of Content Items of this type.

j) Do not click [**All Properties**] because we will leave the default values on the Field Properties dialog.

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 449) 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 105). When you are done, the table should appear as:



Fields

Name	Label	Control
img1	Image:	sys_File
img1_filename	Image File Name:	sys_EditBox
img1_size	Image File Size:	sys_EditBox
img1_type	Image Mime Type:	sys_EditBox
img1_ext	Image Extension:	sys_EditBox
img1_height	Image Height:	sys_EditBox
img1_width	Image Width:	sys_EditBox
img_alt	Image Alt Text:	sys_EditBox
img2	Mini:	sys_File
img2_filename	Mini File Name:	sys_EditBox
img2_size	Mini File Size:	sys_EditBox
img2_type	Mini Mime Type:	sys_EditBox
img2_ext	Mini Extension:	sys_EditBox
img2_height	Mini Height:	sys_EditBox
img2_width	Mini Width:	sys_EditBox

Field Properties

Data type: integer

Storage size: none

Default value:

Mime type mode Default

Mime type value

☑ Enable searching for this field

□ Required

Display

Mnemonic:

☑ Show in preview

Figure 69: Shared Image Field Set

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:

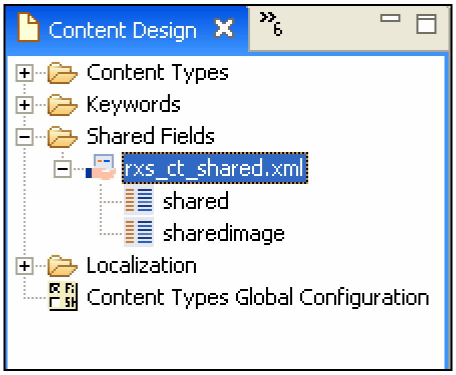
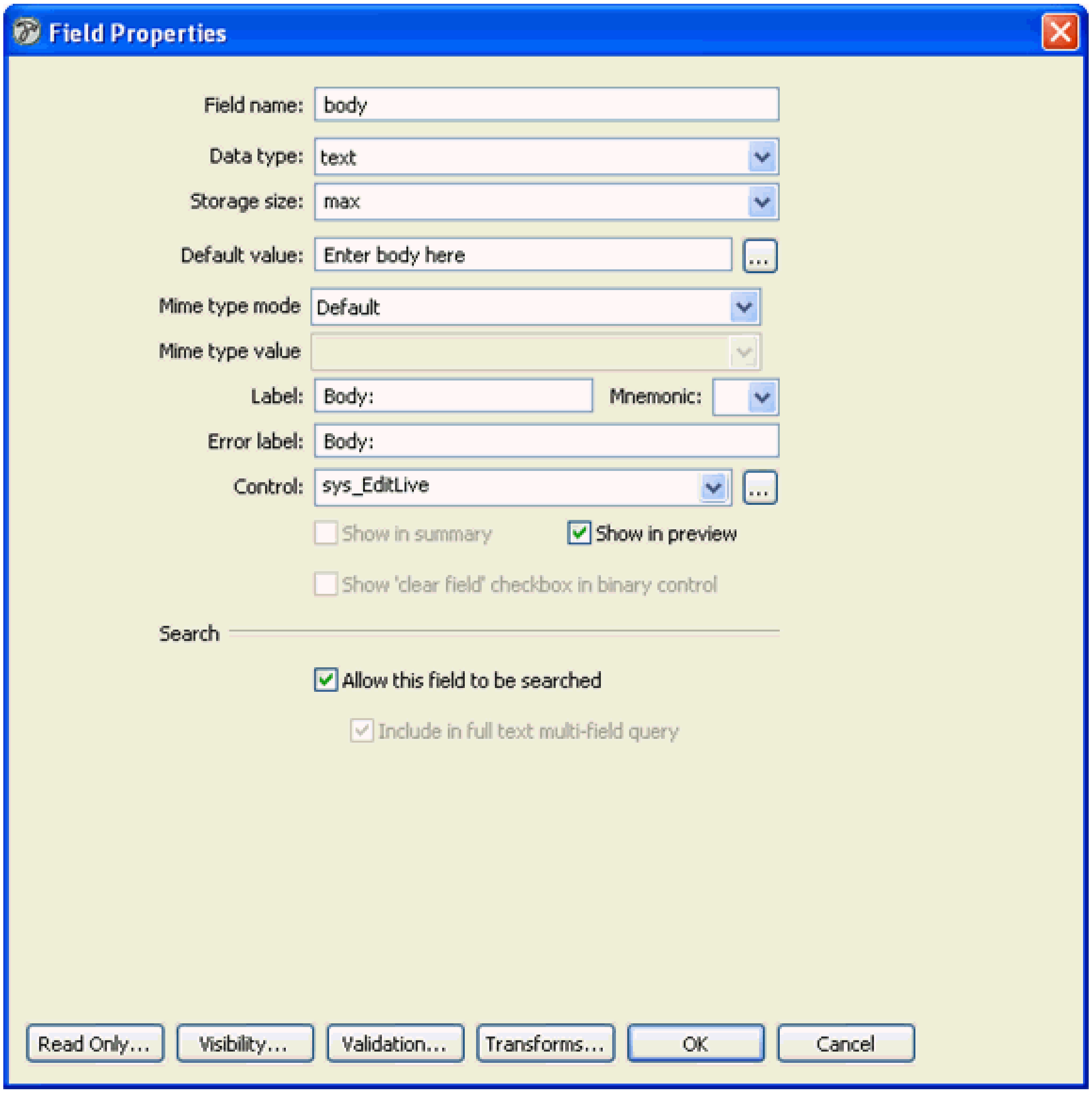


Figure 70: test_shared.xml now appears in Content Design view

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 name: body

Data type: text

Storage size: max

Default value: Enter body here

Mime type mode Default

Mime type value

Label: Body: Mnemonic:

Error label: Body:

Control: sys_EditLive

□ Show in summary ☑ Show in preview

□ Show 'clear field' checkbox in binary control

Search

☑ Allow this field to be searched

☑ Include in full text multi-field query

Figure 71: Field Properties dialog

The table below explains the functions of these rules and give some examples. The following topic, ***Adding a Field Visibility Rule*** (see page 105), 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.	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. 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.
validation	Specify what values for a field are valid.	A validation rule could specify that a value must be entered in a field. A validation rule could specify that a credit card number entered in a field must be located in an external database.
transformation	Input transformation rules specify how Rhythmyx should modify a value for storage in the repository when it is entered in a Content Editor field. NOTE: An input transformation rule should always be preceded by a validation rule to ensure that the data submitted is valid for the transformation. Output transformation rules specify how Rhythmyx should modify a value when it retrieves it from the repository to display in a Content Editor field.	An input transformation rule could specify that a date entered in the format Month dd, yyyy is stored in the format yyyymmdd. 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.

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 85).

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**].

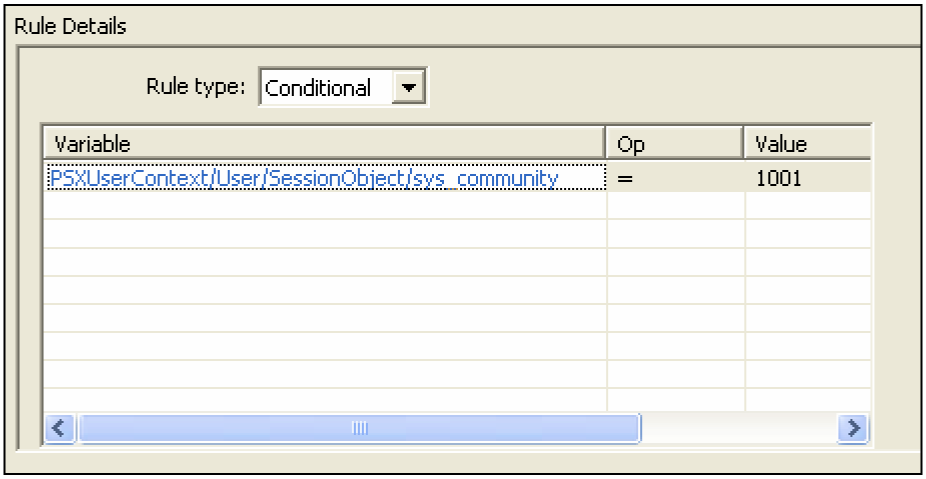
The Field Visibility editor opens.

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 users in 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_community = 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

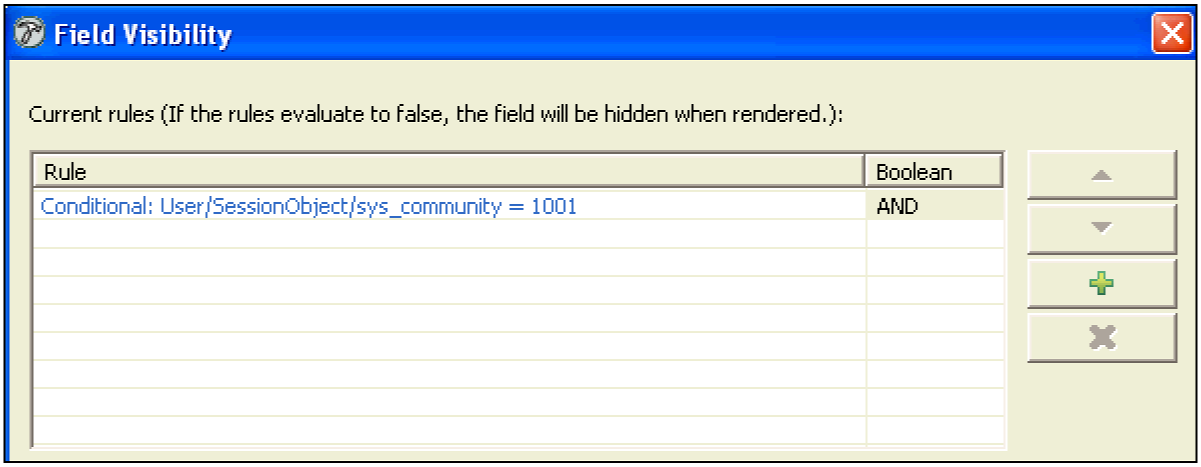
Rule type: Conditional

Variable	Op	Value
PSXUserContext/User/SessionObject/sys_community	=	1001

Figure 72: Rule Details

7 Click [**Add**].

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



Field Visibility

Current rules (If the rules evaluate to false, the field will be hidden when rendered.):

Rule	Boolean
Conditional: User/SessionObject/sys_community = 1001	AND

Figure 73: 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.

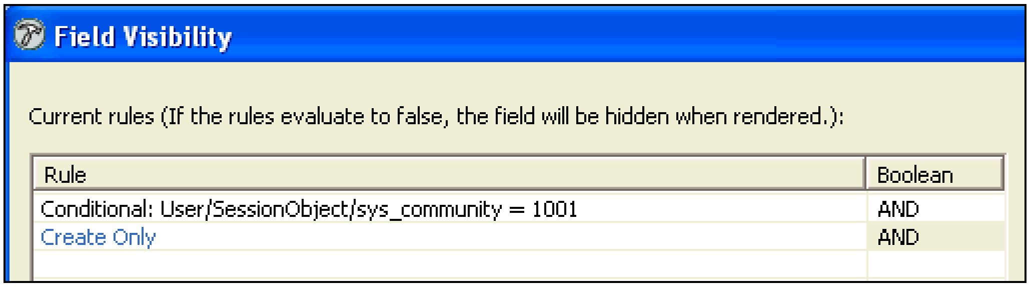
This rule that makes the field visible when the Content Item is being created, but hides it when the Content Item is being edited.

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.



Field Visibility

Current rules (If the rules evaluate to false, the field will be hidden when rendered.):

Rule	Boolean
Conditional: User/SessionObject/sys_community = 1001	AND
Create Only	AND

Figure 74: 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:

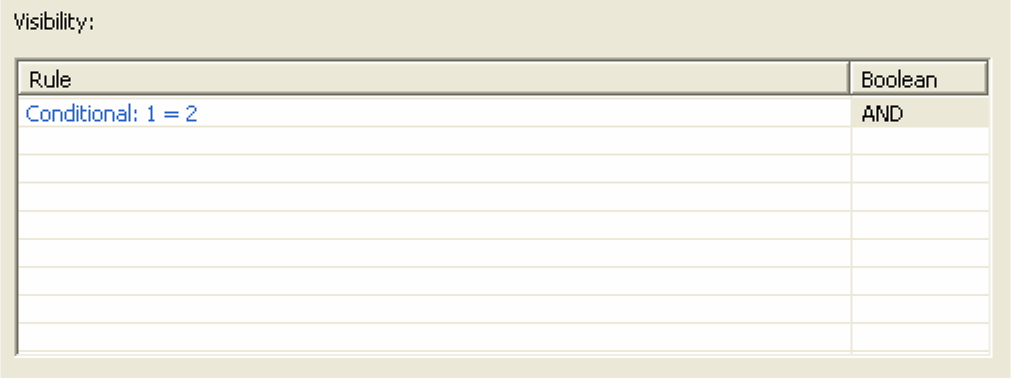
1 Open the Field Visibility editor for the *img1_ext* field.

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.



Visibility:

Rule	Boolean
Conditional: 1 = 2	AND

Figure 75: 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	

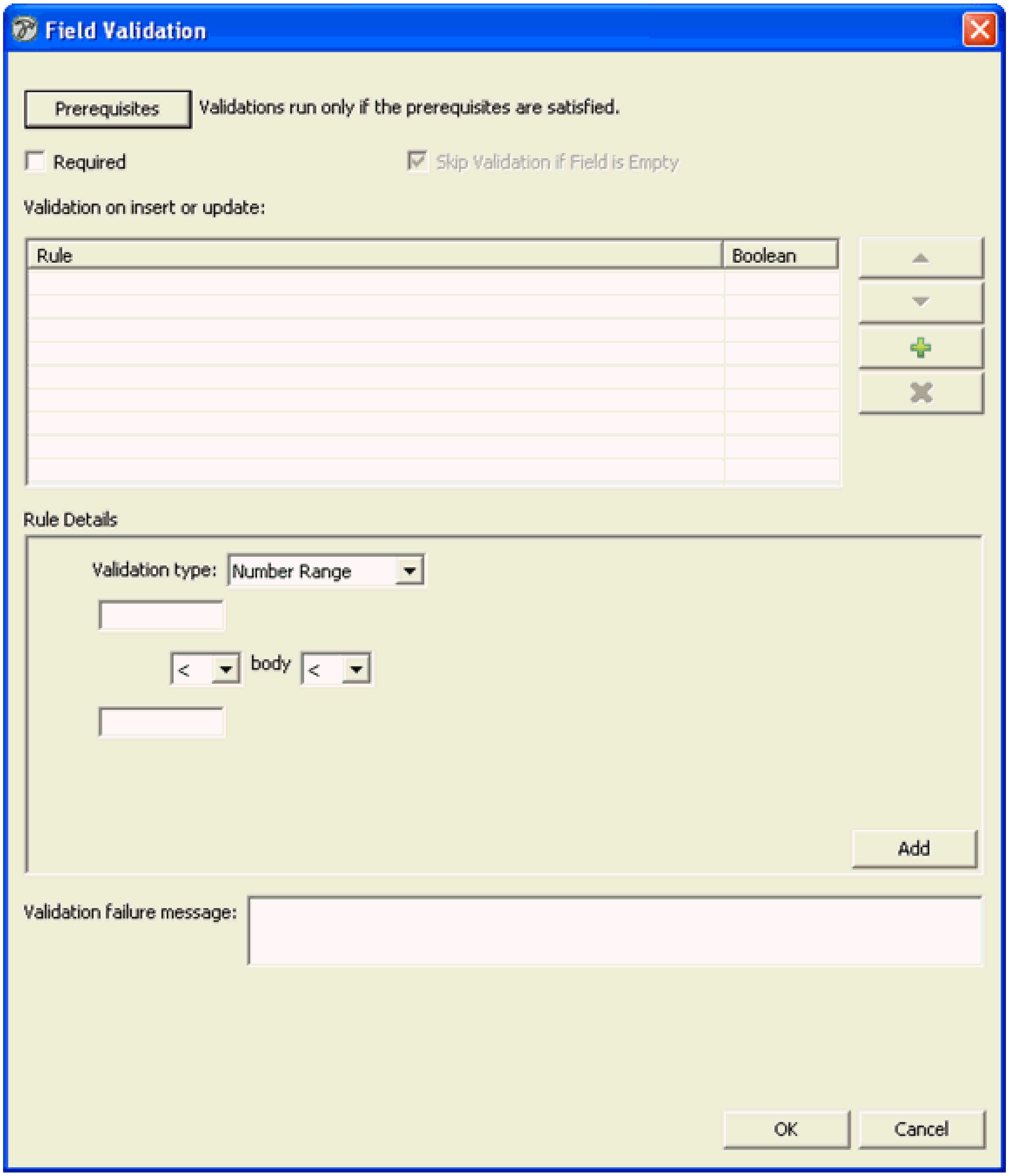
Adding a Field Validation Rule

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.

□ Required ☑ Skip Validation if Field is Empty

Validation on insert or update:

Rule	Boolean

Rule Details

Validation type: Number Range

< body <

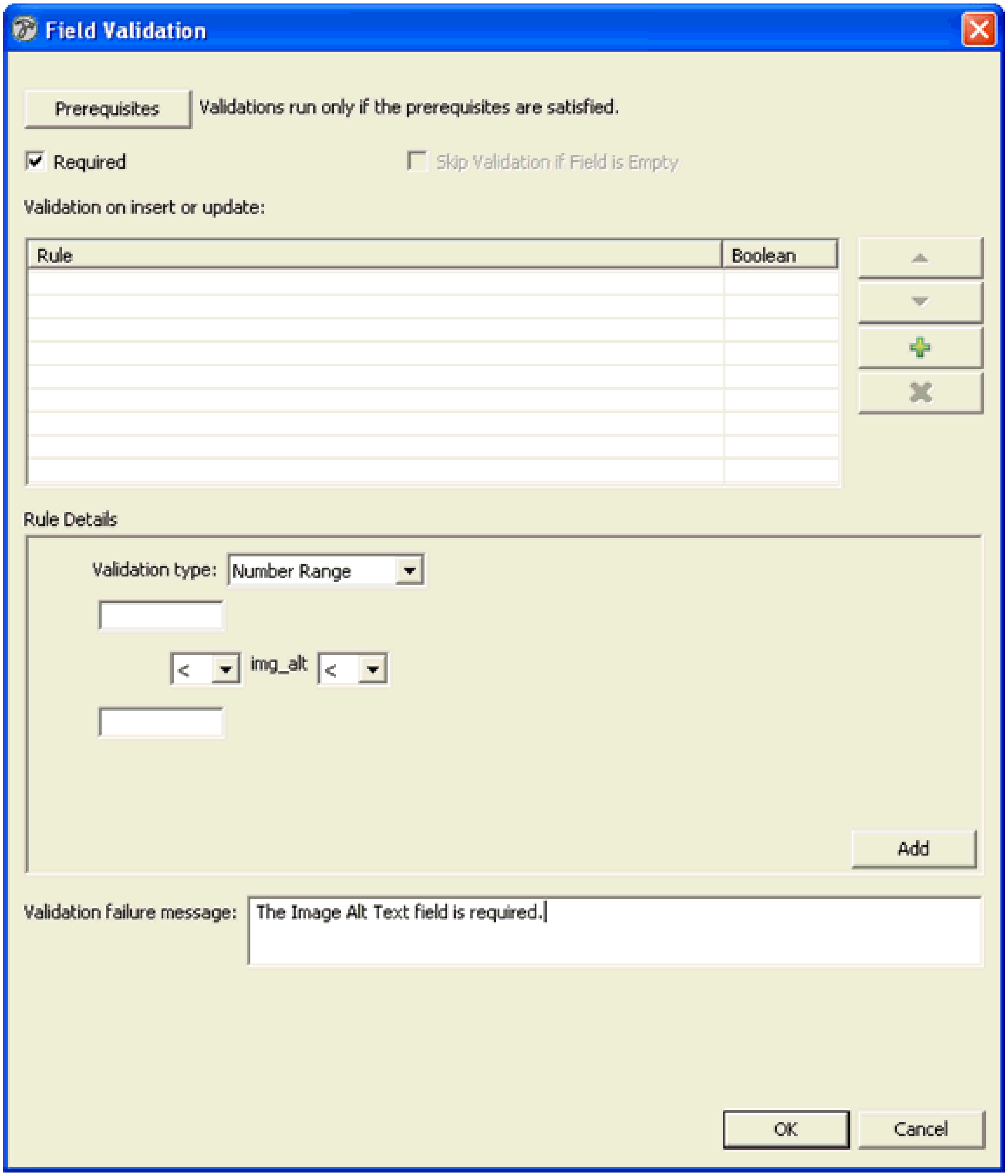
Validation failure message:

Figure 76: Field Validation Editor

2 No prerequisite conditions must be met before the validation occurs, so do not press the [**Prerequisites**] button.

3 Check *Required* since this is the validation that you want to use.

4 In the Validation Failure Message text box, enter *The Image Alt Text field is required*.



Field Validation

Prerequisites Validations run only if the prerequisites are satisfied.

☑ Required □ Skip Validation if Field is Empty

Validation on insert or update:

Rule	Boolean

Rule Details

Validation type: Number Range

< img_alt <

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

Figure 77: Field Validation dialog

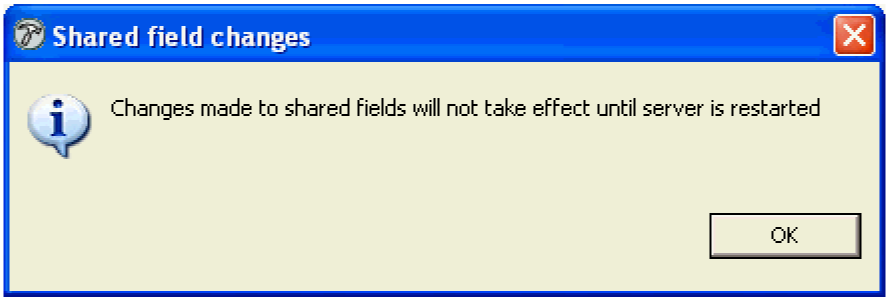
5 Click [**OK**].

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.

6 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.



Shared field changes

Changes made to shared fields will not take effect until server is restarted

Figure 78: Shared field changes dialog

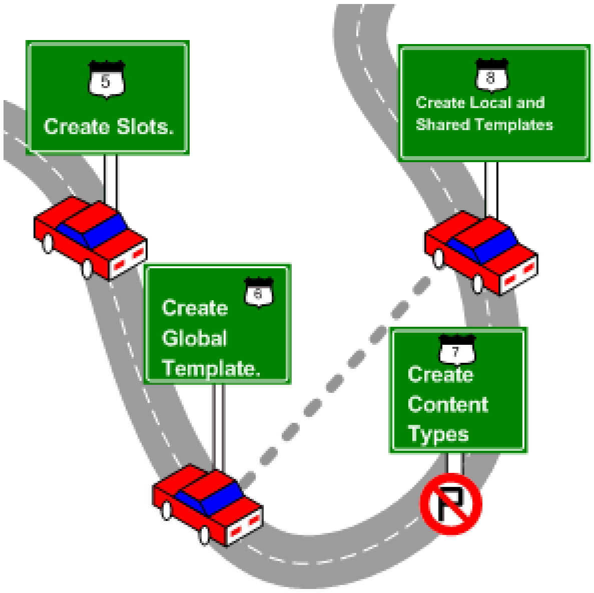
7 Click [**OK**].

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 215); 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.

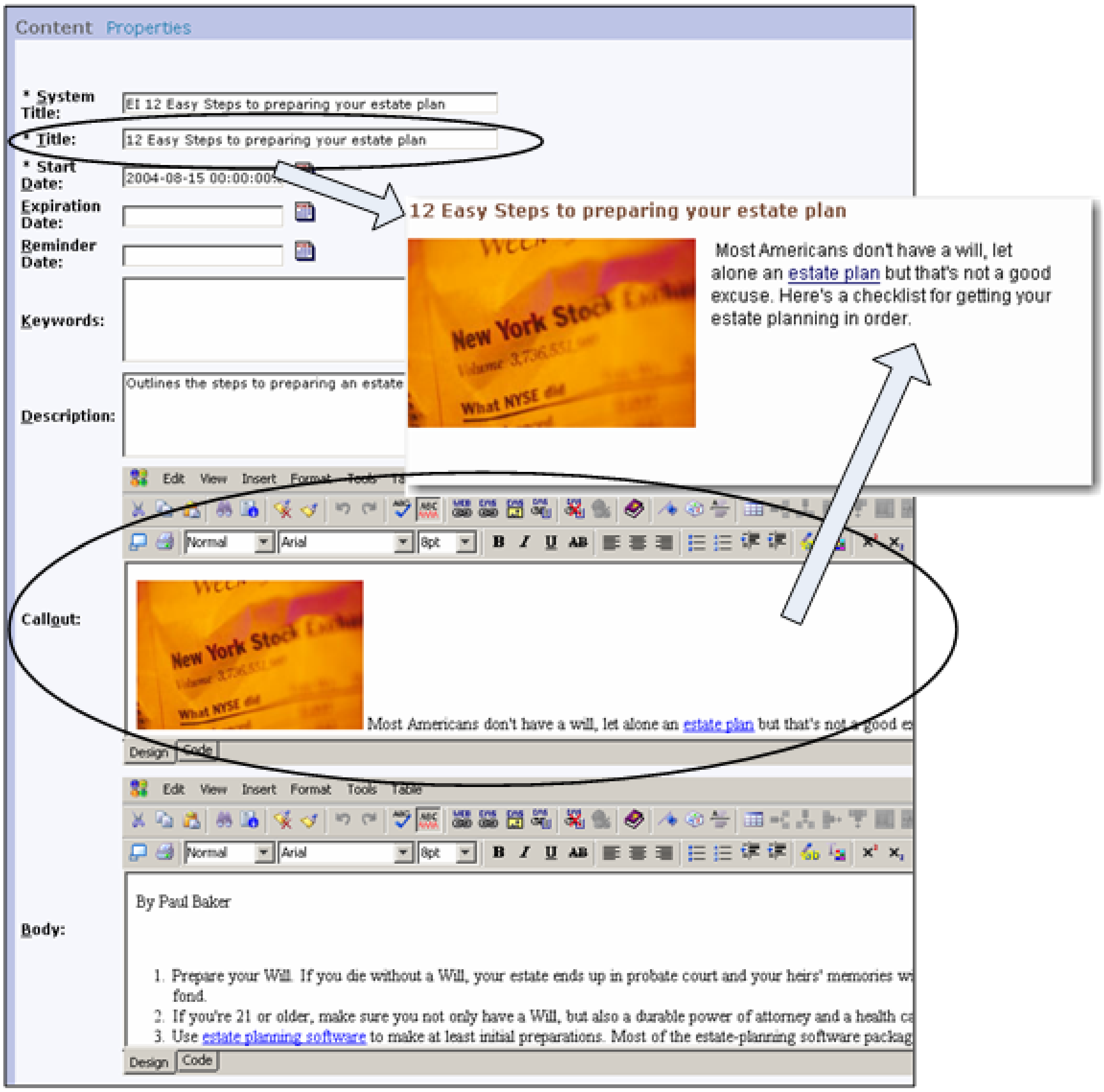
During modeling and design, at least one Template is specified for each Content Type defined in the implementation. The Template specifies

- 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.



* System Title: EI 12 Easy Steps to preparing your estate plan

* Title: 12 Easy Steps to preparing your estate plan

* Start Date: 2004-08-15 00:00:00

Expiration Date:

Reminder Date:

Keywords:

Description: Outlines the steps to preparing an estate

Callout: Most Americans don't have a will, let alone an estate plan but that's not a good ex

Body: By Paul Baker

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
3. Use estate planning software to make at least initial preparations. Most of the estate-planning software packag

12 Easy Steps to preparing your estate plan

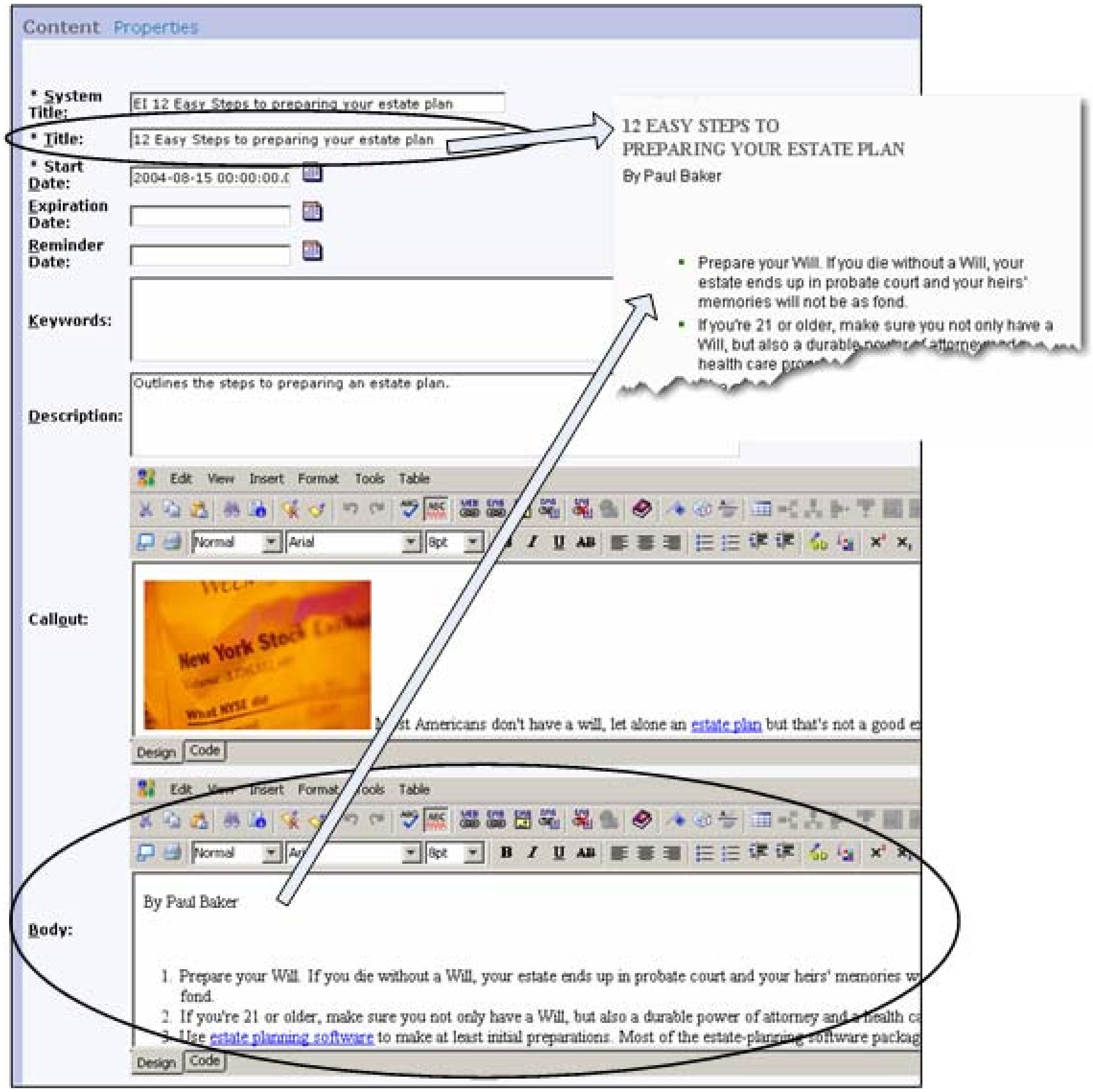
Most Americans don't have a will, let alone an estate plan but that's not a good excuse. Here's a checklist for getting your estate planning in order.

Figure 79: 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.



* System Title: EI 12 Easy Steps to preparing your estate plan

* Title: 12 Easy Steps to preparing your estate plan

* Start Date: 2004-08-15 00:00:00.(

Expiration Date:

Reminder Date:

Keywords:

Description: Outlines the steps to preparing an estate plan.

Callout: Most Americans don't have a will, let alone an estate plan but that's not a good e

Body:

By Paul Baker

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
3. Use estate planning software to make at least initial preparations. Most of the estate-planning software packag

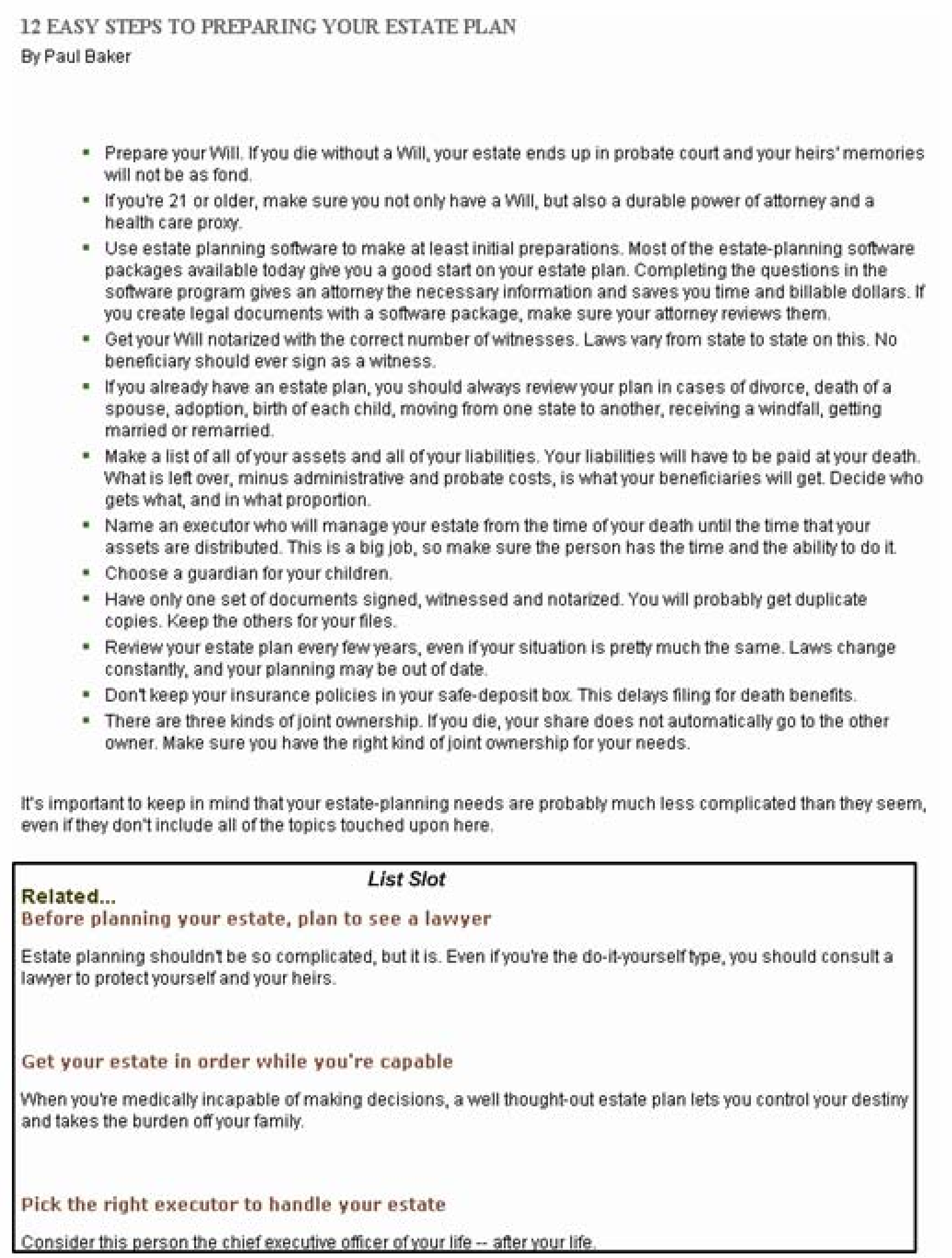
12 EASY STEPS TO
PREPARING YOUR ESTATE PLAN

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 pro

Figure 80: 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.

Related...

Before planning your estate, plan to see a lawyer

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.

Get your estate in order while you're capable

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.

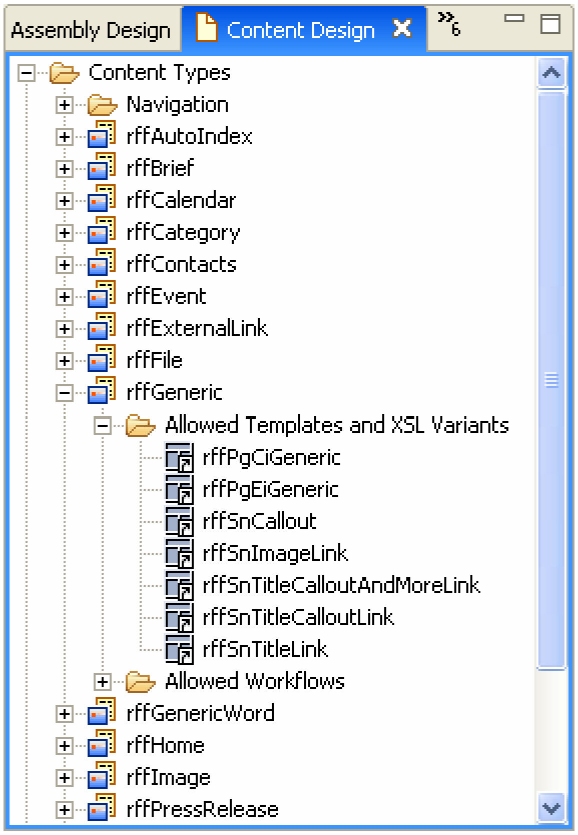
Pick the right executor to handle your estate

Consider this person the chief executive officer of your life -- after your life.

Figure 81: 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.

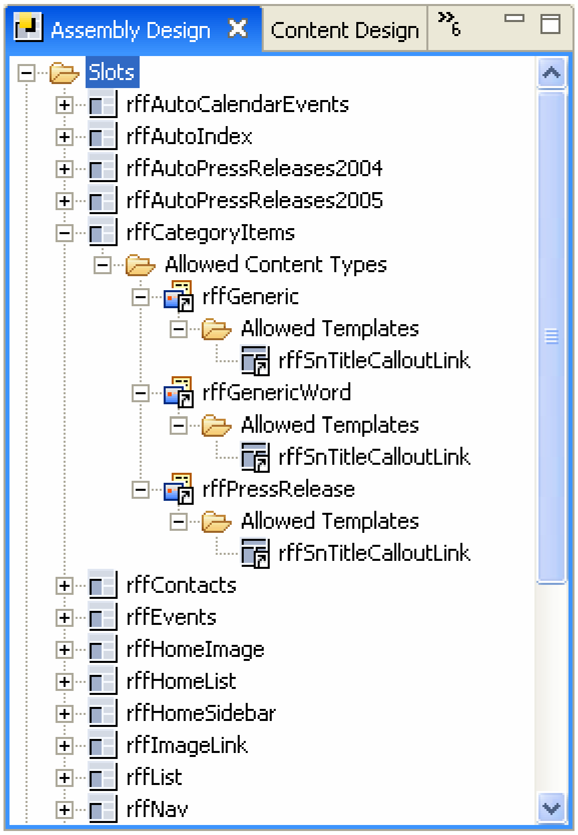
The Workbench illustrates the relationships among these objects. For example, in Content View, you can list the Templates associated with each Content Type. For each Template, you can show the list of associated Slots. Note, however, that the View does not show the complete recursion. The Slots used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



- Content Types
  - Navigation
  - rffAutoIndex
  - rffBrief
  - rffCalendar
  - rffCategory
  - rffContacts
  - rffEvent
  - rffExternalLink
  - rffFile
  - rffGeneric
    - Allowed Templates and XSL Variants
      - rffPgCiGeneric
      - rffPgEiGeneric
      - rffSnCallout
      - rffSnImageLink
      - rffSnTitleCalloutAndMoreLink
      - rffSnTitleCalloutLink
      - rffSnTitleLink
    - Allowed Workflows
  - rffGenericWord
  - rffHome
  - rffImage
  - rffPressRelease

Figure 82: 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.



- Slots
  - rffAutoCalendarEvents
  - rffAutoIndex
  - rffAutoPressReleases2004
  - rffAutoPressReleases2005
  - rffCategoryItems
    - Allowed Content Types
      - rffGeneric
        - Allowed Templates
          - rffSnTitleCalloutLink
      - rffGenericWord
        - Allowed Templates
          - rffSnTitleCalloutLink
      - rffPressRelease
        - Allowed Templates
          - rffSnTitleCalloutLink
  - rffContacts
  - rffEvents
  - rffHomeImage
  - rffHomeList
  - rffHomeSidebar
  - rffImageLink
  - rffList
  - rffNav

Figure 83: 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.

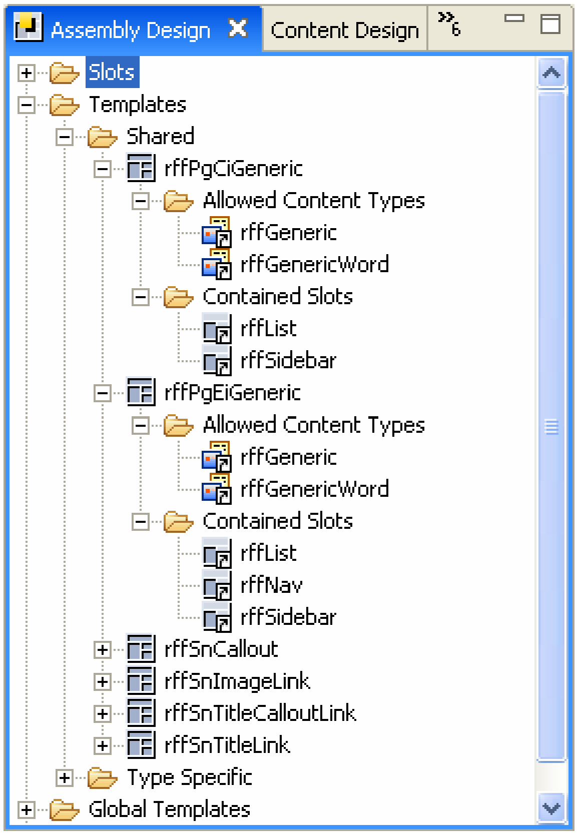


Figure 84: Assembly View showing Templates, associated Content Types and Slots

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 279).

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 192).

  - 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**
rffSnImageLink	General Slot for Images	Active Assembly	sys_RelationshipContentFinder

The following Content Types are allowed in this Slot:

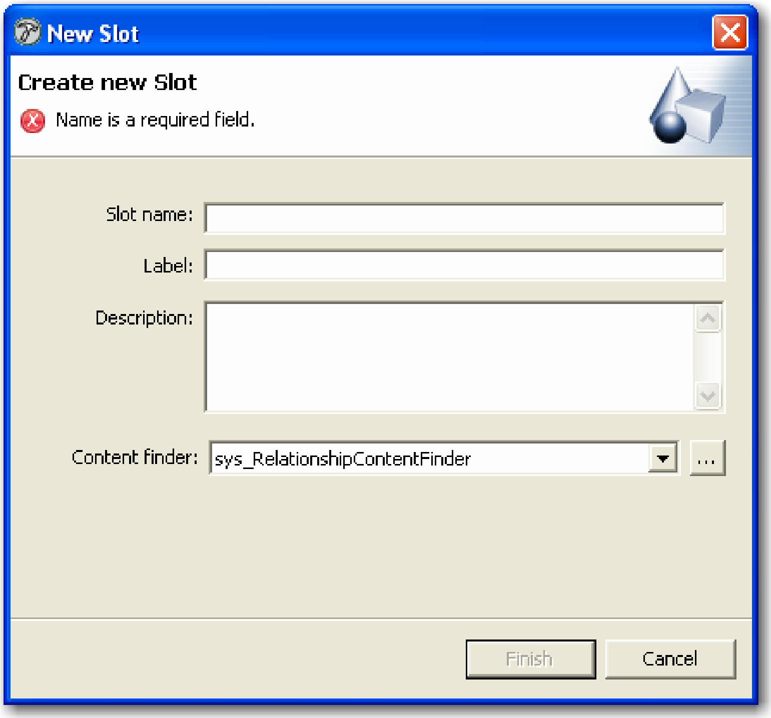
Content Type	**Template**
Image	rffSnImage
Image	rffSnImageandTitle

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.



Create new Slot

Name is a required field.

Slot name:

Label:

Description:

Content finder: sys_RelationshipContentFinder

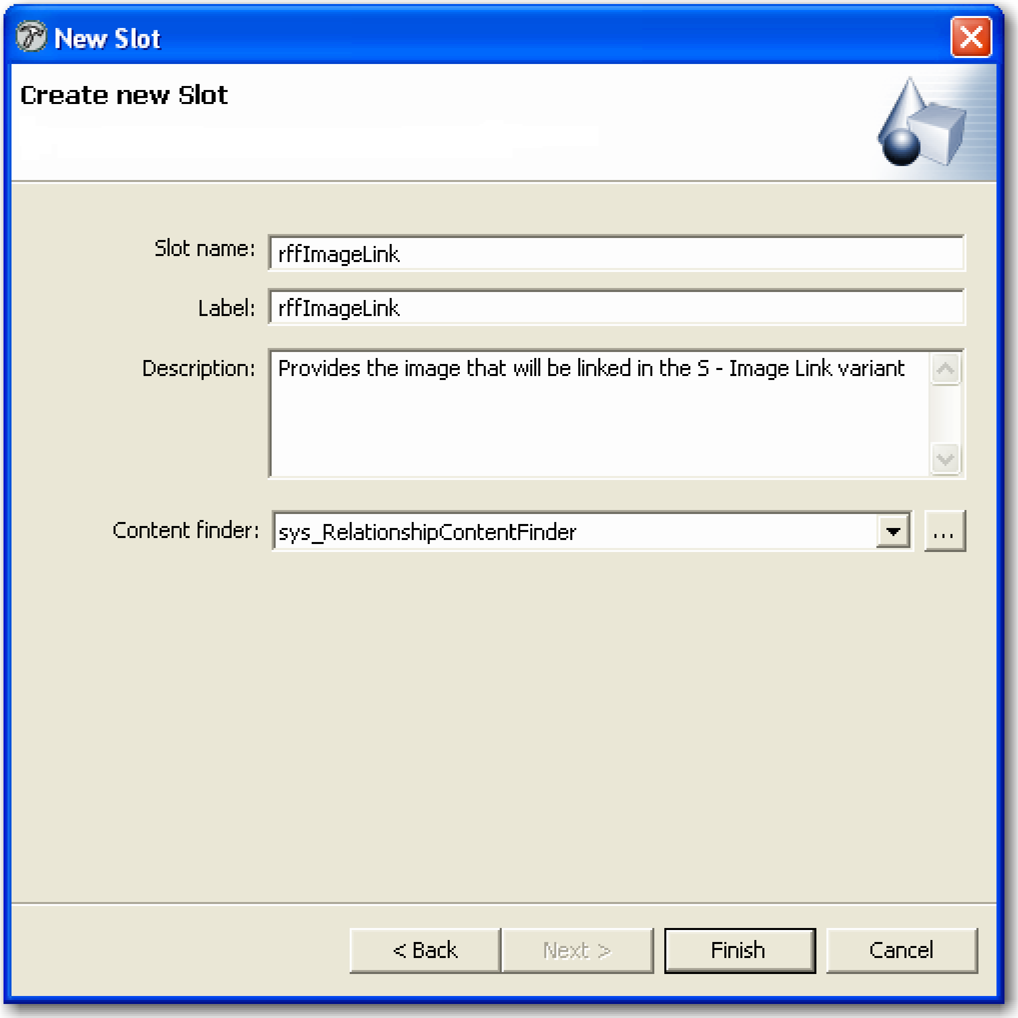
Figure 85: Slot Wizard

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 120).



Create new Slot

Slot name: rffImageLink

Label: rffImageLink

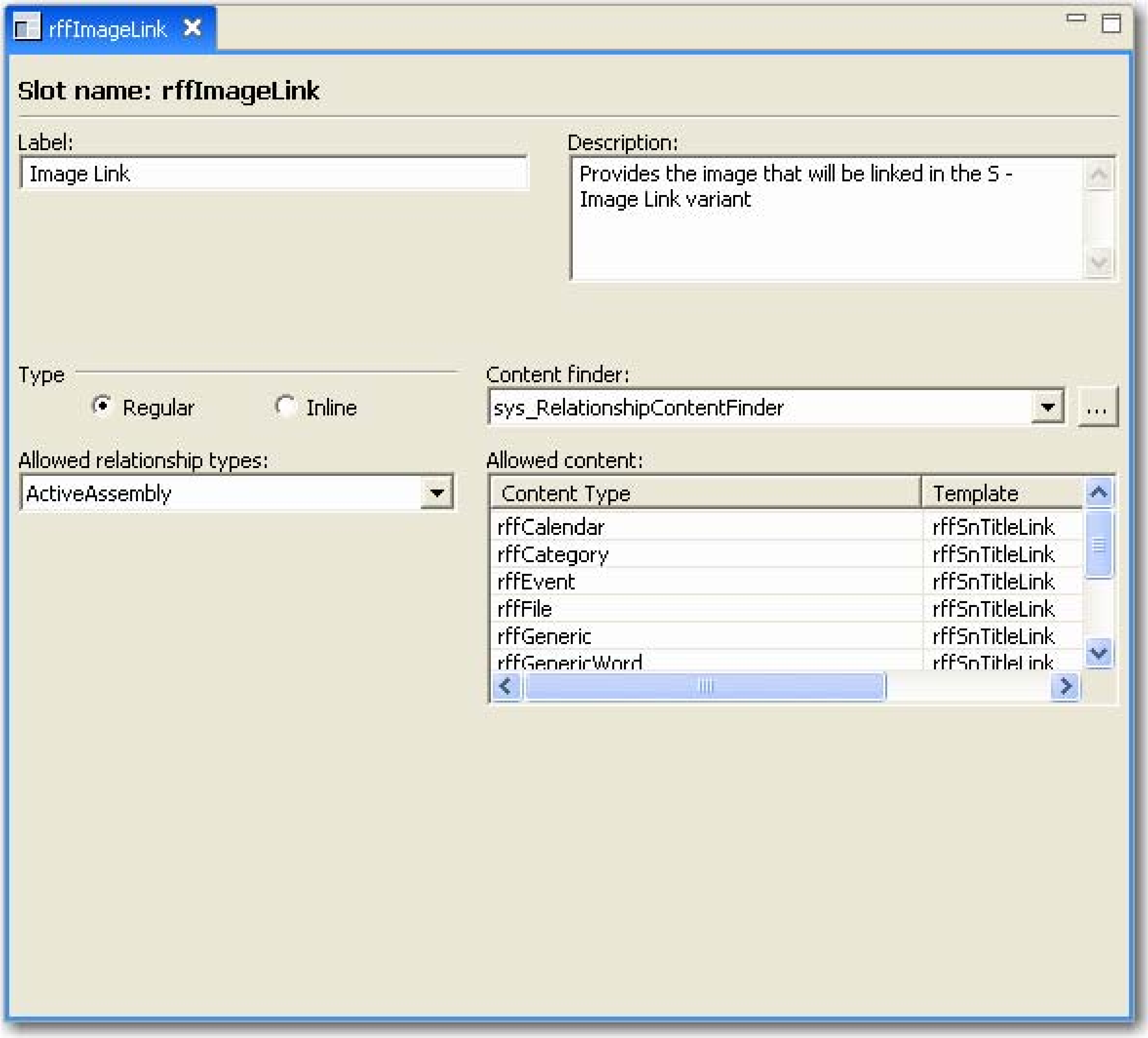
Description: Provides the image that will be linked in the S - Image Link variant

Content finder: sys_RelationshipContentFinder

Figure 86: 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.



Slot name: rffImageLink

Label: Image Link

Description: Provides the image that will be linked in the S - Image Link variant

Type: ● Regular ○ Inline

Content finder: sys_RelationshipContentFinder

Allowed relationship types: ActiveAssembly

Allowed content:

Content Type	Template
rffCalendar	rffSnTitleLink
rffCategory	rffSnTitleLink
rffEvent	rffSnTitleLink
rffFile	rffSnTitleLink
rffGeneric	rffSnTitleLink
rffGenericWord	rffSnTitleLink

Figure 87: 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

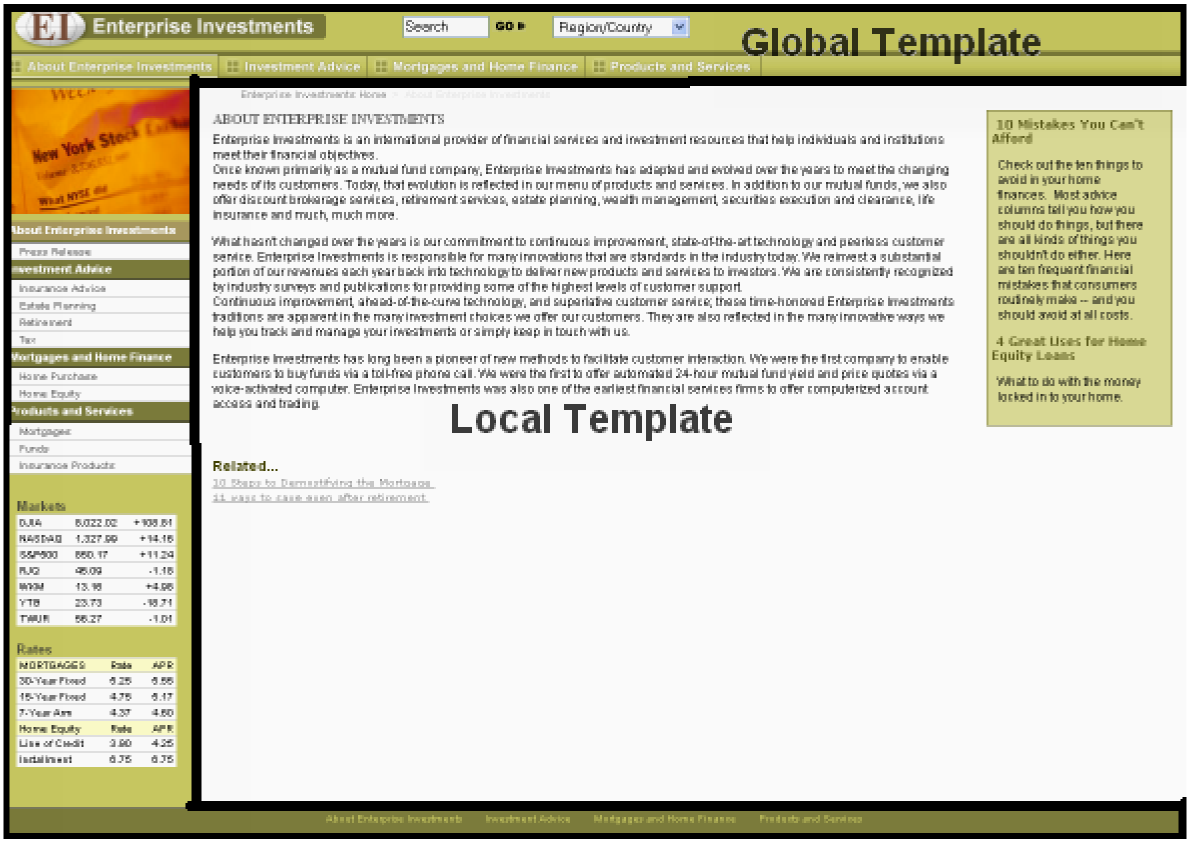
In some cases, you may want more control over the contents of a Slot. For example:

- 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 (desc). 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.



ABOUT ENTERPRISE INVESTMENTS

Enterprise Investments is an international provider of financial services and investment resources that help individuals and institutions meet their financial objectives.
Once known primarily as a mutual fund company, Enterprise Investments has adapted and evolved over the years to meet the changing needs of its customers. Today, that evolution is reflected in our menu of products and services. In addition to our mutual funds, we also offer discount brokerage services, retirement services, estate planning, wealth management, securities execution and clearance, life insurance and much, much more.

What hasn't changed over the years is our commitment to continuous improvement, state-of-the-art technology and peerless customer service. Enterprise Investments is responsible for many innovations that are standards in the industry today. We reinvest a substantial portion of our revenues each year back into technology to deliver new products and services to investors. We are consistently recognized by industry surveys and publications for providing some of the highest levels of customer support.
Continuous improvement, ahead-of-the-curve technology, and superlative customer service; these time-honored Enterprise Investments traditions are apparent in the many investment choices we offer our customers. They are also reflected in the many innovative ways we help you track and manage your investments or simply keep in touch with us.

Enterprise Investments has long been a pioneer of new methods to facilitate customer interaction. We were the first company to enable customers to buy funds via a toll-free phone call. We were the first to offer automated 24-hour mutual fund yield and price quotes via a voice-activated computer. Enterprise Investments was also one of the earliest financial services firms to offer computerized account access and trading.

Related...
10 Steps to Demystifying the Mortgage
11 ways to ease exen after retirement

10 Mistakes You Can't Afford

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.

4 Great Uses for Home Equity Loans

What to do with the money locked in to your home.

Markets

DJIA	8,022.02	+108.81
NASDAQ	1,327.99	+14.16
S&P500	850.17	+11.24
RJG	45.09	-1.16
WKM	13.16	+4.96
YTB	23.73	-16.71
TWUR	56.27	-1.01

Rates

MORTGAGES	Rate	APR
30-Year Fixed	6.25	6.56
15-Year Fixed	4.75	5.17
7-Year Arm	4.37	4.80
Home Equity	Rate	APR
Line of Credit	3.90	4.25
Installment	6.75	6.75

Figure 88: 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 363).)

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***, 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 141).

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.

In a Snippet Template, all content within the HTML `<body>` tag should be wrapped in a `<div>` or `<span>` tag.

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: If a Snippet Template 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.

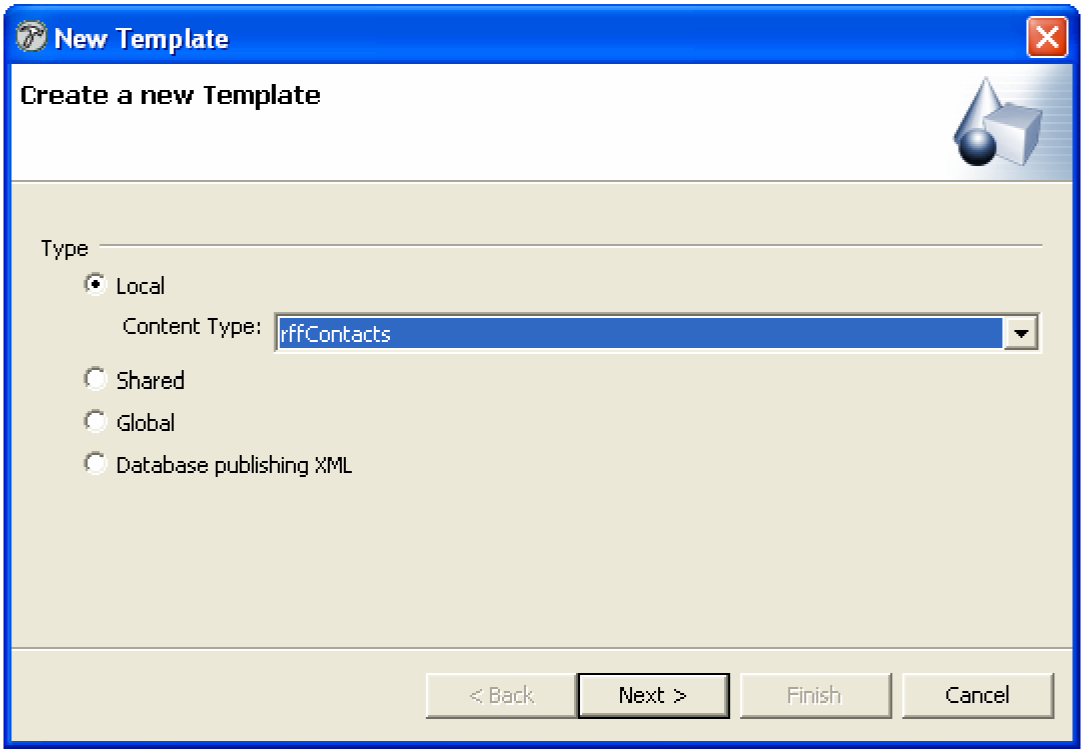
Creating a Text Template Object

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.



Type

● Local

Content Type: rffContacts

○ Shared

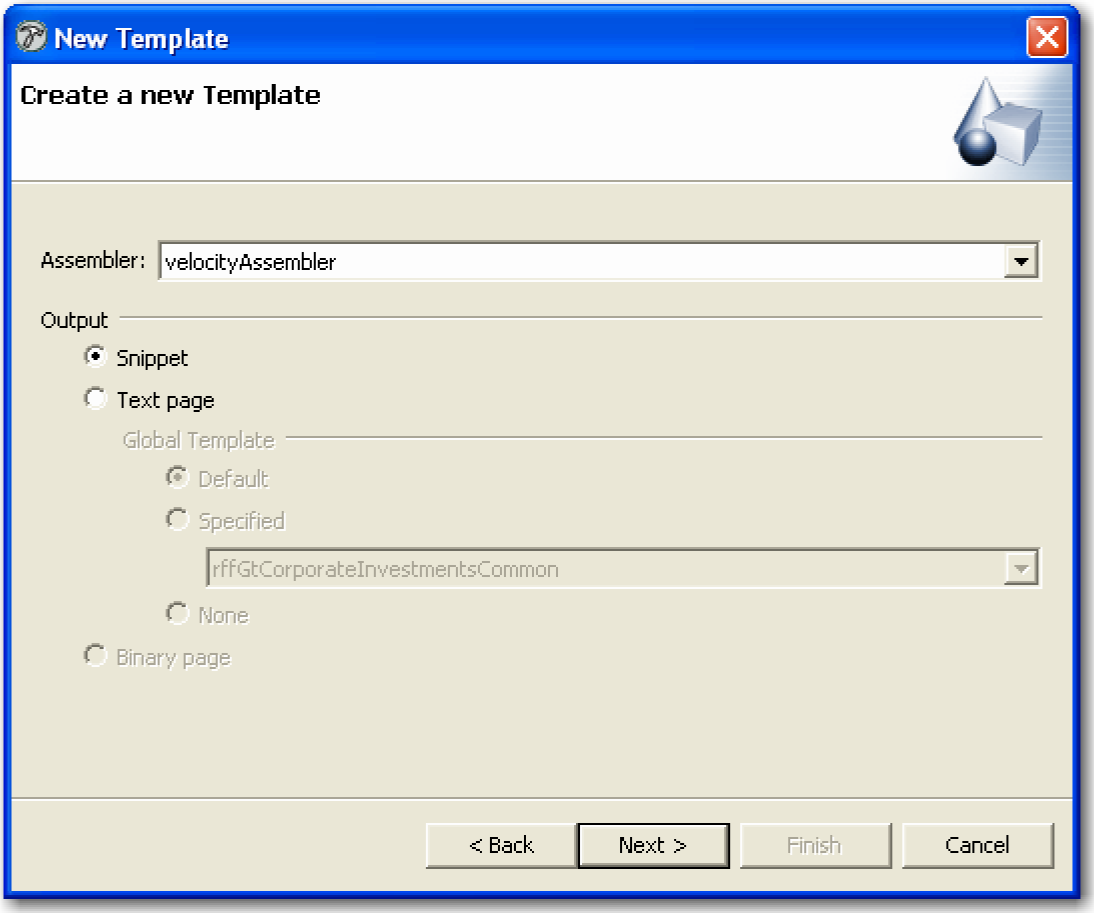
○ Global

○ Database publishing XML

Figure 89: 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.



New Template

Create a new Template

Assembler: velocityAssembler

Output

● Snippet

○ Text page

Global Template

● Default

○ Specified

rffGtCorporateInvestmentsCommon

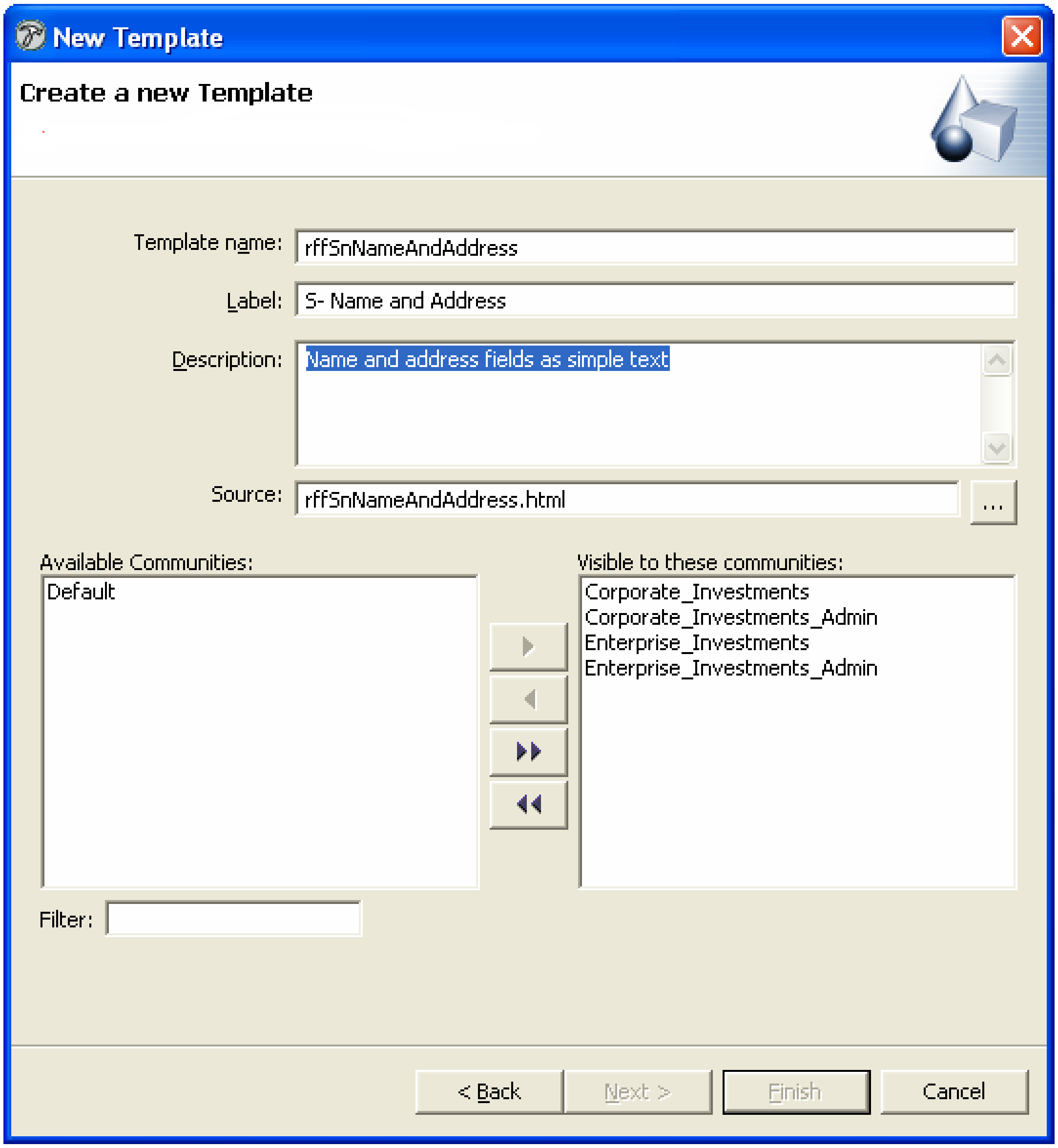
○ None

○ Binary page

Figure 90: 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.



Template name: rffSnNameAndAddress

Label: S- Name and Address

Description: Name and address fields as simple text

Source: rffSnNameAndAddress.html

Available Communities:

Default

Visible to these communities:

Corporate_Investments
Corporate_Investments_Admin
Enterprise_Investments
Enterprise_Investments_Admin

Filter:

Figure 91: 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.

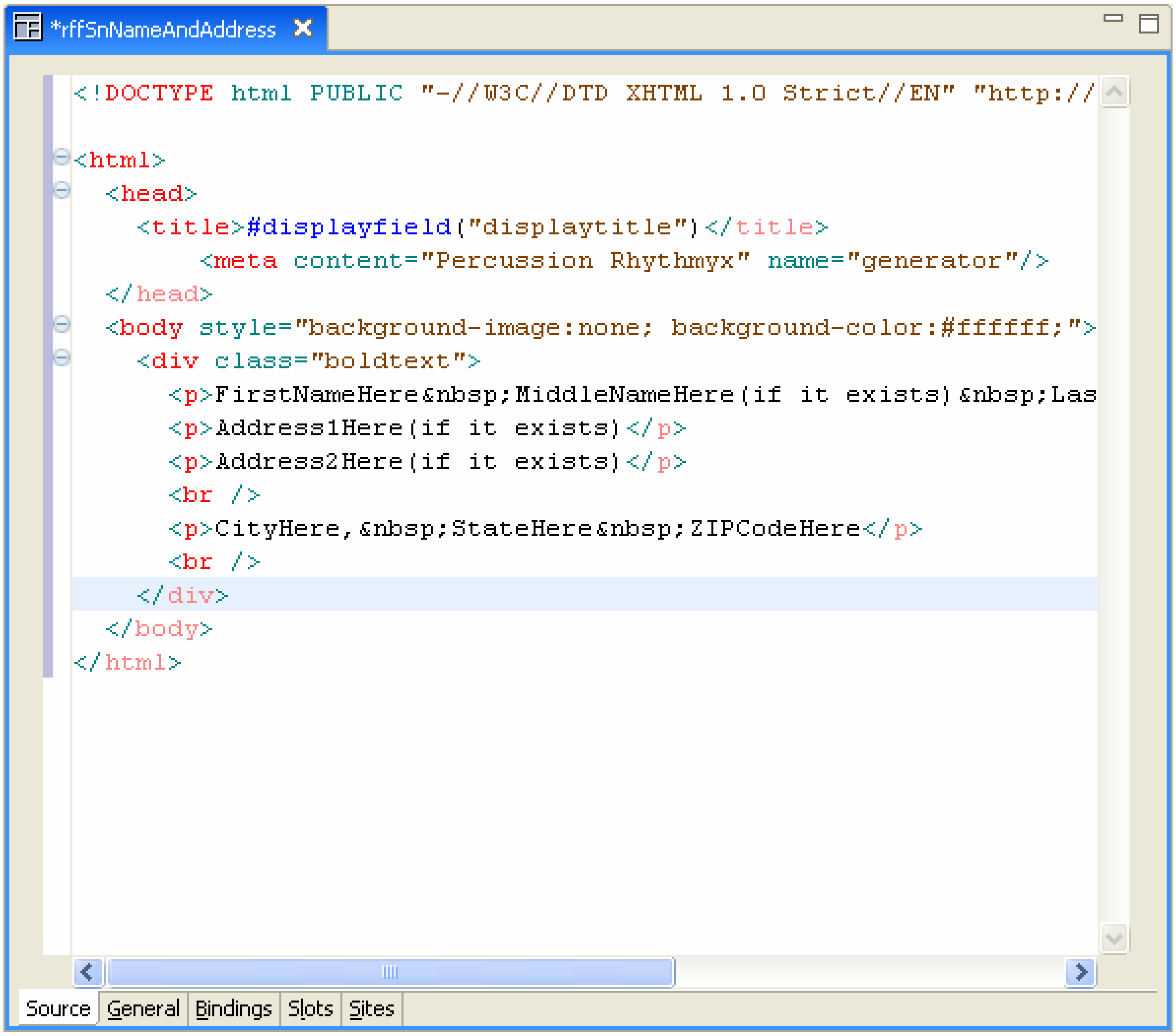
The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

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&nbsp;MiddleNameHere(if it exists)&nbsp;Las
      <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 92: 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:

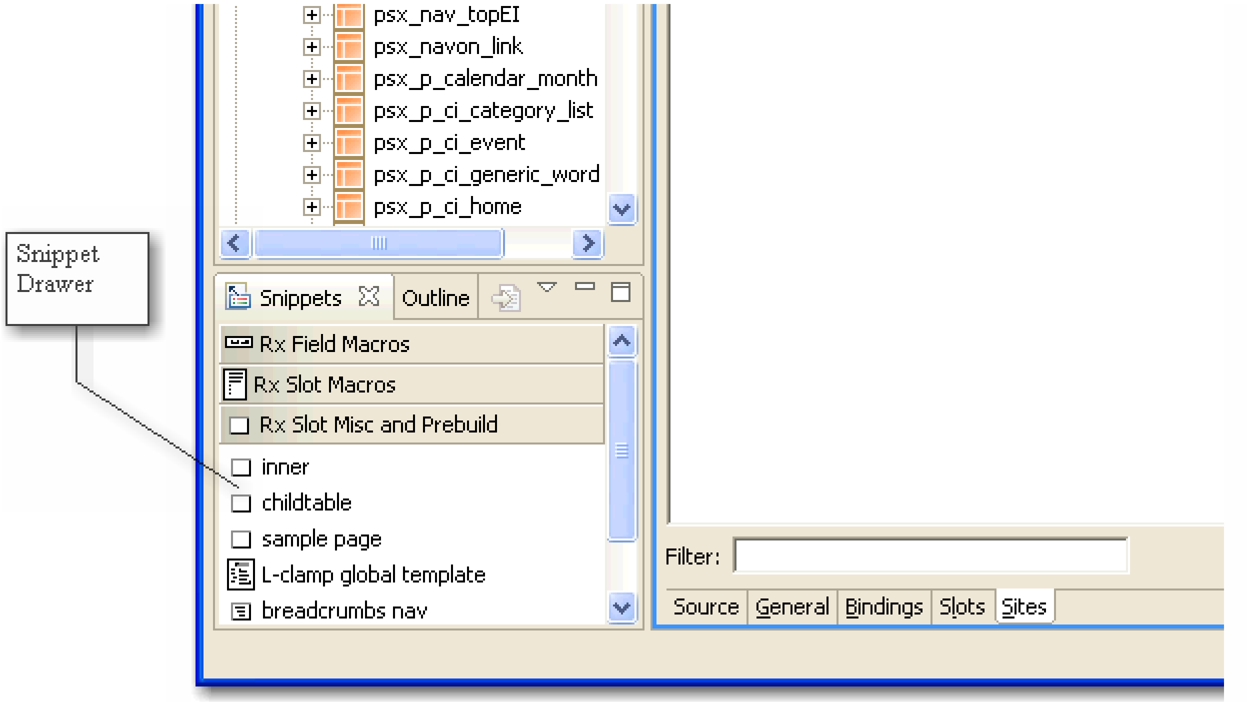


Figure 93: Snippet Drawer

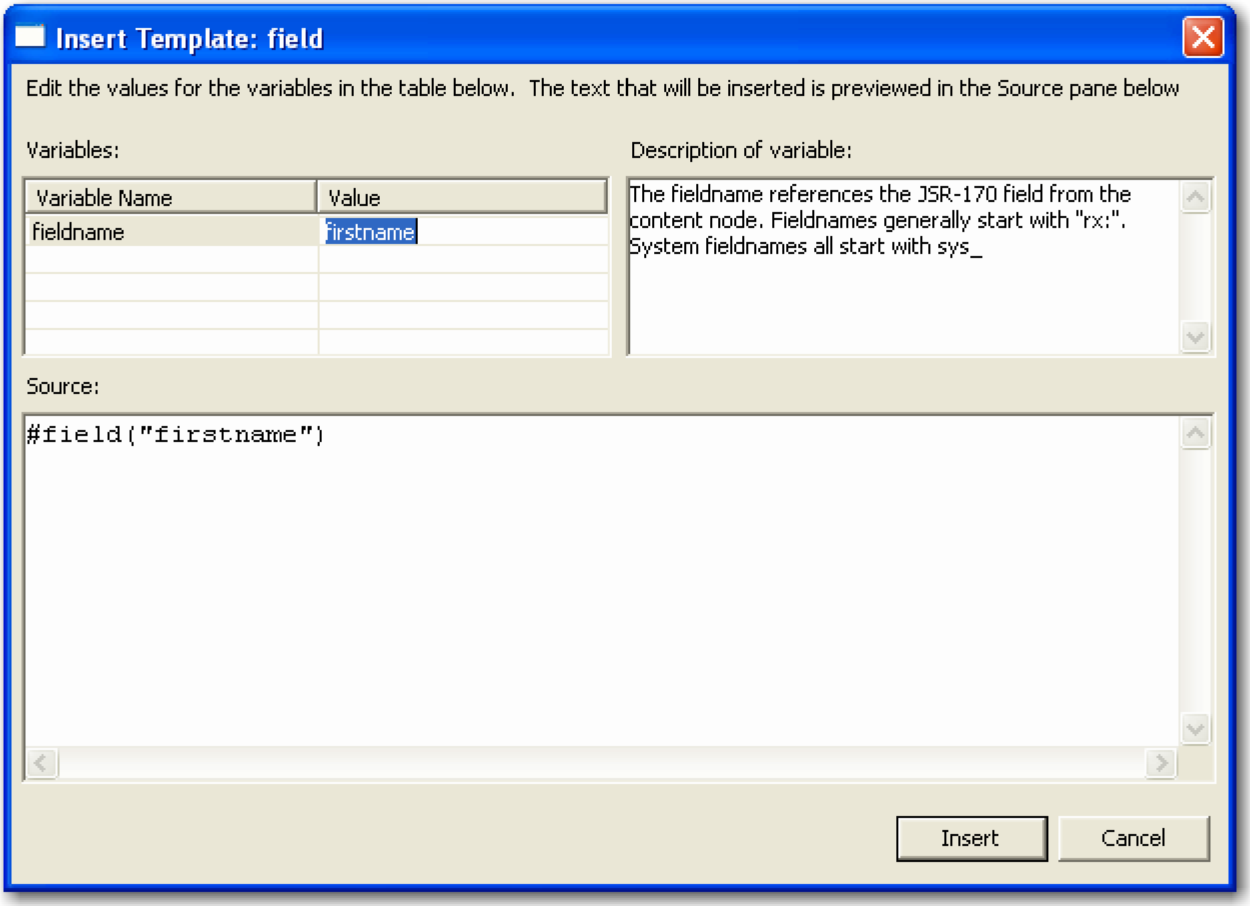
NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

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*.



Insert Template: field

Edit the values for the variables in the table below. The text that will be inserted is previewed in the Source pane below

Variables:

Variable Name	Value
fieldname	firstname

Description of variable:

The fieldname references the JSR-170 field from the content node. Fieldnames generally start with "rx:". System fieldnames all start with sys_

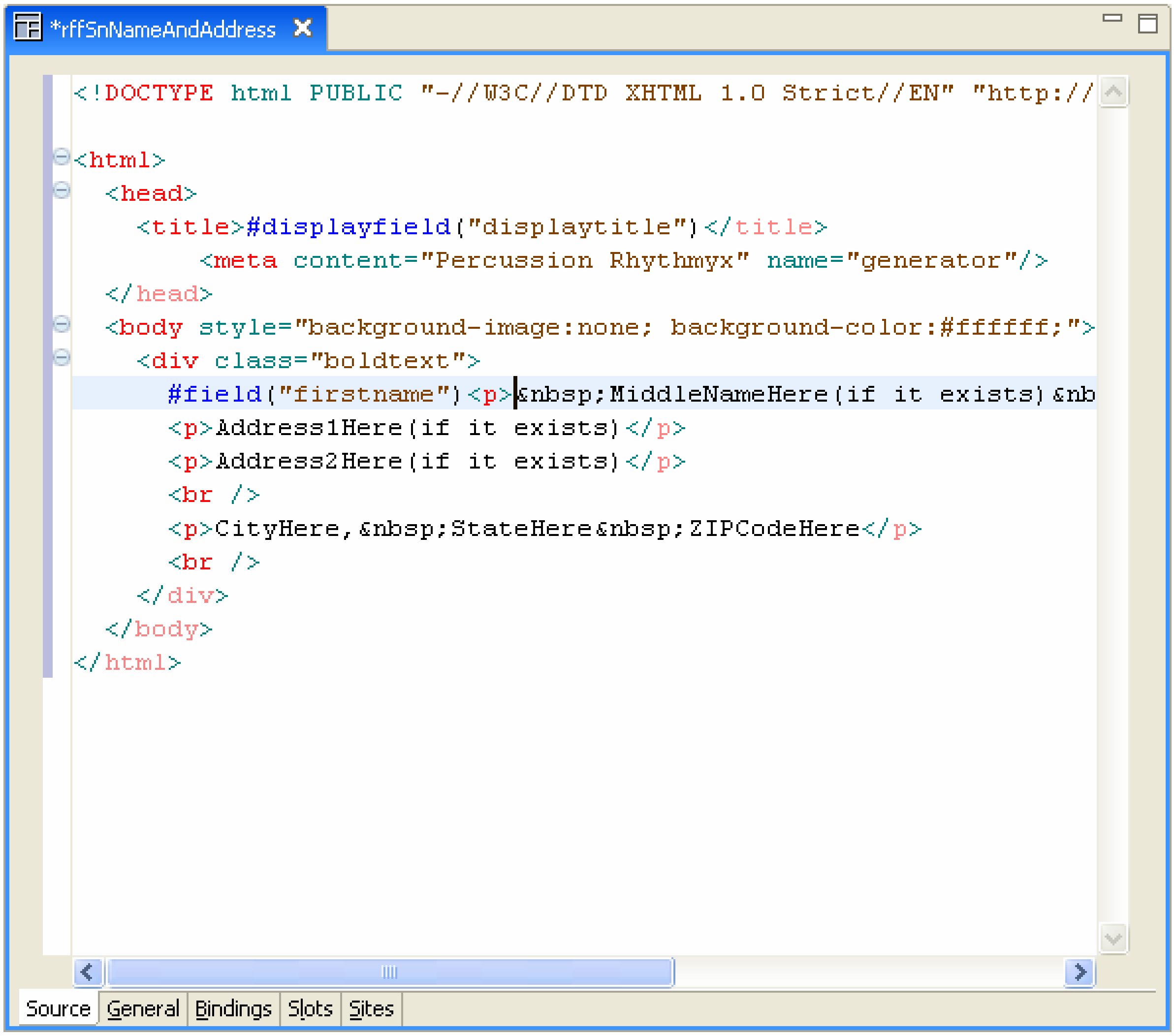
Source:

```
#field("firstname")
```

Figure 94: 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 95: 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.

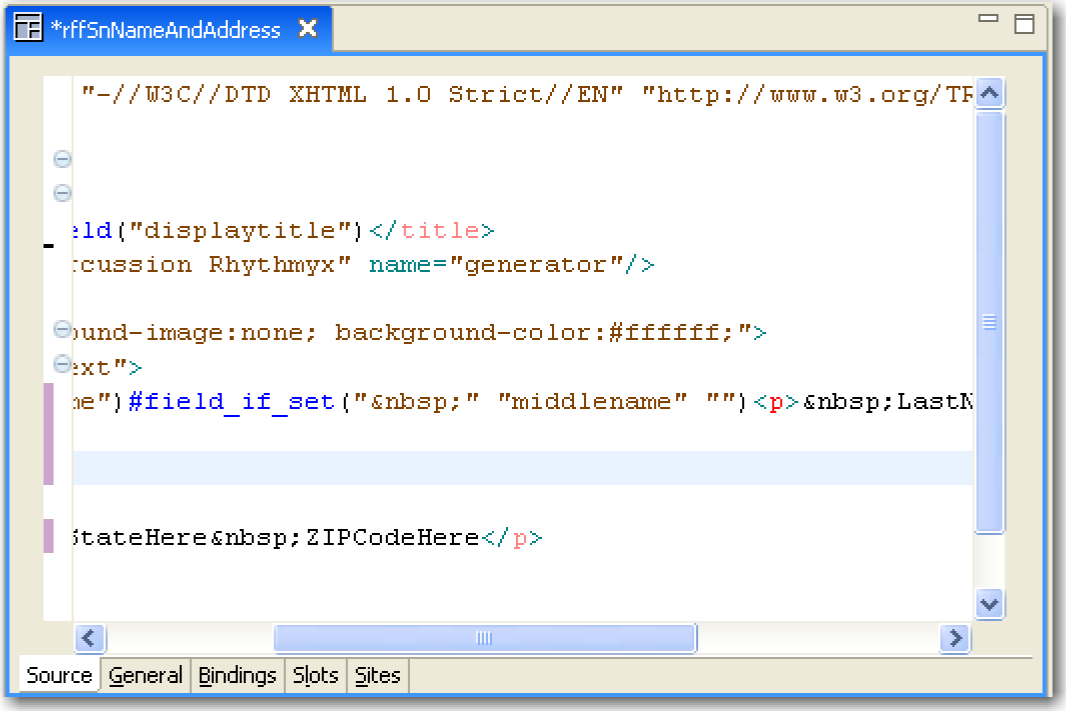
- after

  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:



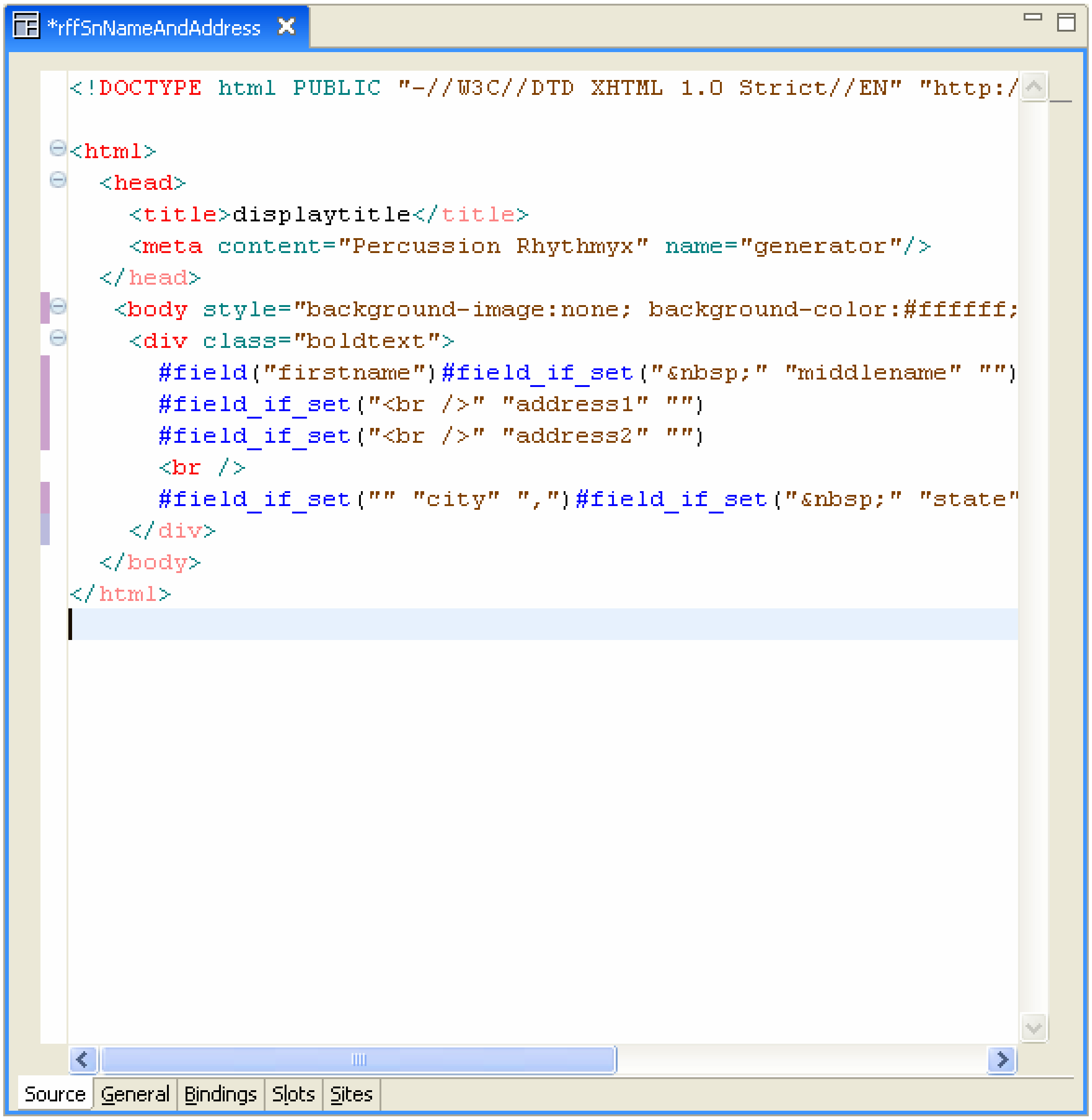
```
"-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR



eld("displaytitle")</title>
cussion Rhythmyx" name="generator"/>

ound-image:none; background-color:#ffffff;">
ext">
ne")#field_if_set("&nbsp;" "middlename" "")<p>&nbsp;LastN



StateHere&nbsp;ZIPCodeHere</p>
```

Figure 96: 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	#field_if_set("&nbsp;" "lastname" "")
address1	#field_if_set("<br />" "address1" "")
address2	#field_if_set("<br />" "address2" "")
city	#field_if_set("" "city" ",")
state	#field_if_set("&nbsp;" "state" "")
zipcode	#field_if_set("&nbsp;" "zipcode" "")

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>
```

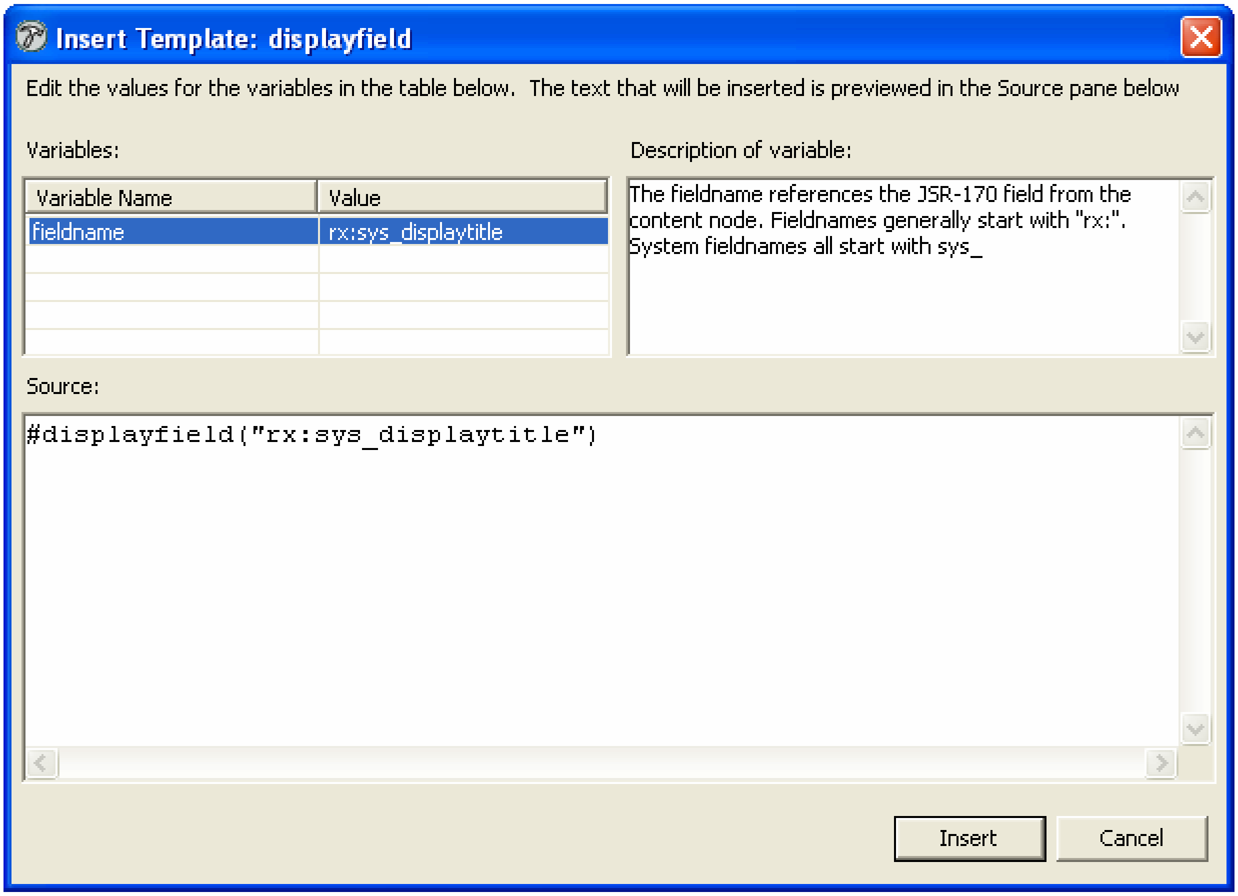
Figure 97: rffNameAndAddress HTML with all fields marked up

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:



Insert Template: displayfield

Edit the values for the variables in the table below. The text that will be inserted is previewed in the Source pane below

Variables:

Variable Name	Value
fieldname	rx:sys_displaytitle

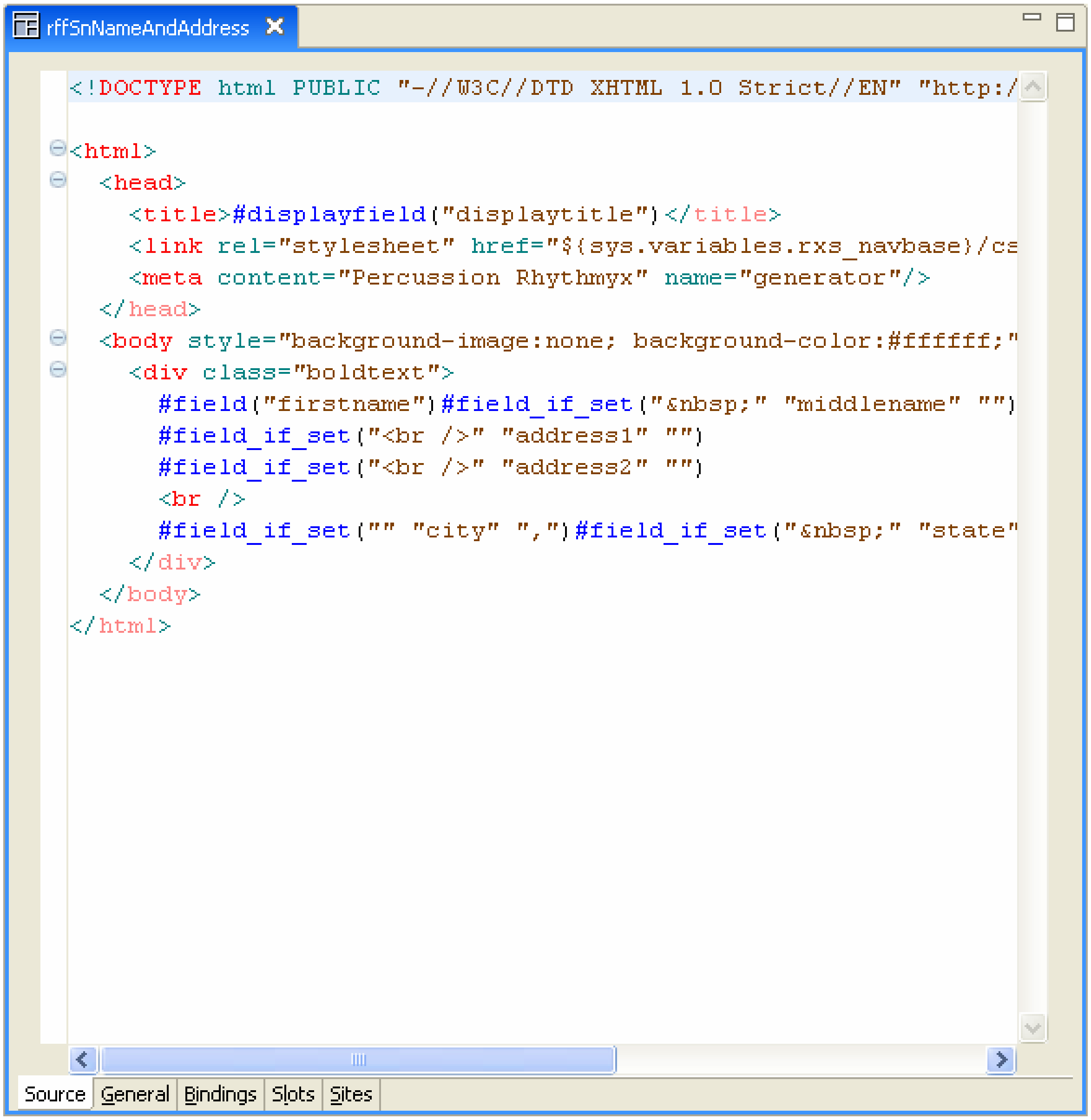
Description of variable:

The fieldname references the JSR-170 field from the content node. Fieldnames generally start with "rx:". System fieldnames all start with sys_

Source:

```
#displayfield("rx:sys_displaytitle")
```

Figure 98: Insert Template dialog showing the #displayfield macro

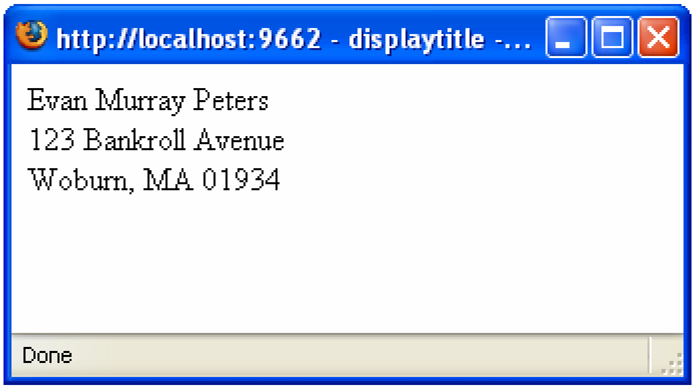


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http:/

<html>
  <head>
    <title>#displayfield("displaytitle")</title>
    <link rel="stylesheet" href="${sys.variables.rxs_navbase}/cs
    <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 99: 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 180).

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



Evan Murray Peters
123 Bankroll Avenue
Woburn, MA 01934

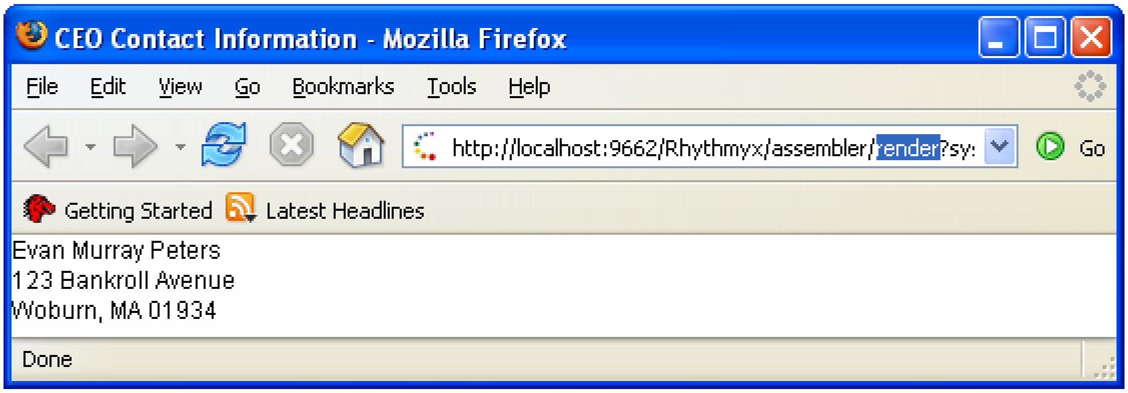
Figure 100: rffSnNameAndAddress preview

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`, `#displaydatefield`, and `#datefield_if_set` macros.

Debugging Templates

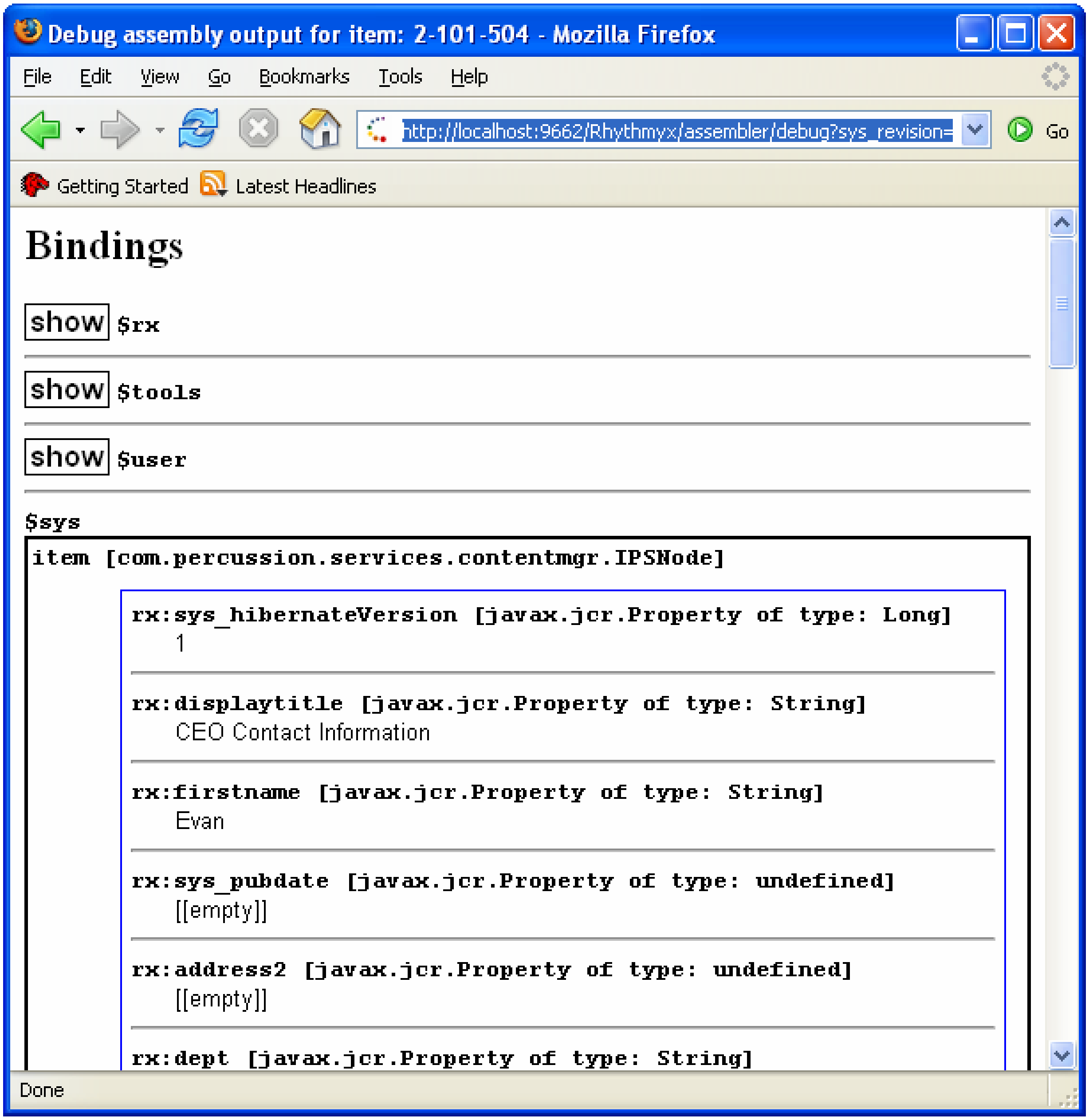
A debugging output is available to help diagnose Templates that generate errors. To see the debug output, in the URL of the preview, change `/assembler/render` to `/assembler/debug`.


CEO Contact Information - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
http://localhost:9662/Rhythmyx/assembler/render?sy:
Go
Getting Started
Latest Headlines
Done

Evan Murray Peters
123 Bankroll Avenue
Woburn, MA 01934

Figure 101: 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.


Debug assembly output for item: 2-101-504 - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
http://localhost:9662/Rhythmyx/assembler/debug?sys_revision=
Go
Getting Started
Latest Headlines
Done

Bindings

show $rx

show $tools

show $user

```
$sys
item [com.percussion.services.contentmgr.IPSNode]
        rx:sys_hibernateVersion [javax.jcr.Property of type: Long]
            1
        rx:displaytitle [javax.jcr.Property of type: String]
            CEO Contact Information
        rx:firstname [javax.jcr.Property of type: String]
            Evan
        rx:sys_pubdate [javax.jcr.Property of type: undefined]
            [[empty]]
        rx:address2 [javax.jcr.Property of type: undefined]
            [[empty]]
        rx:dept [javax.jcr.Property of type: String]
```

Figure 102: 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 407).

- a value definition

  The value definition is a 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://) 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
```

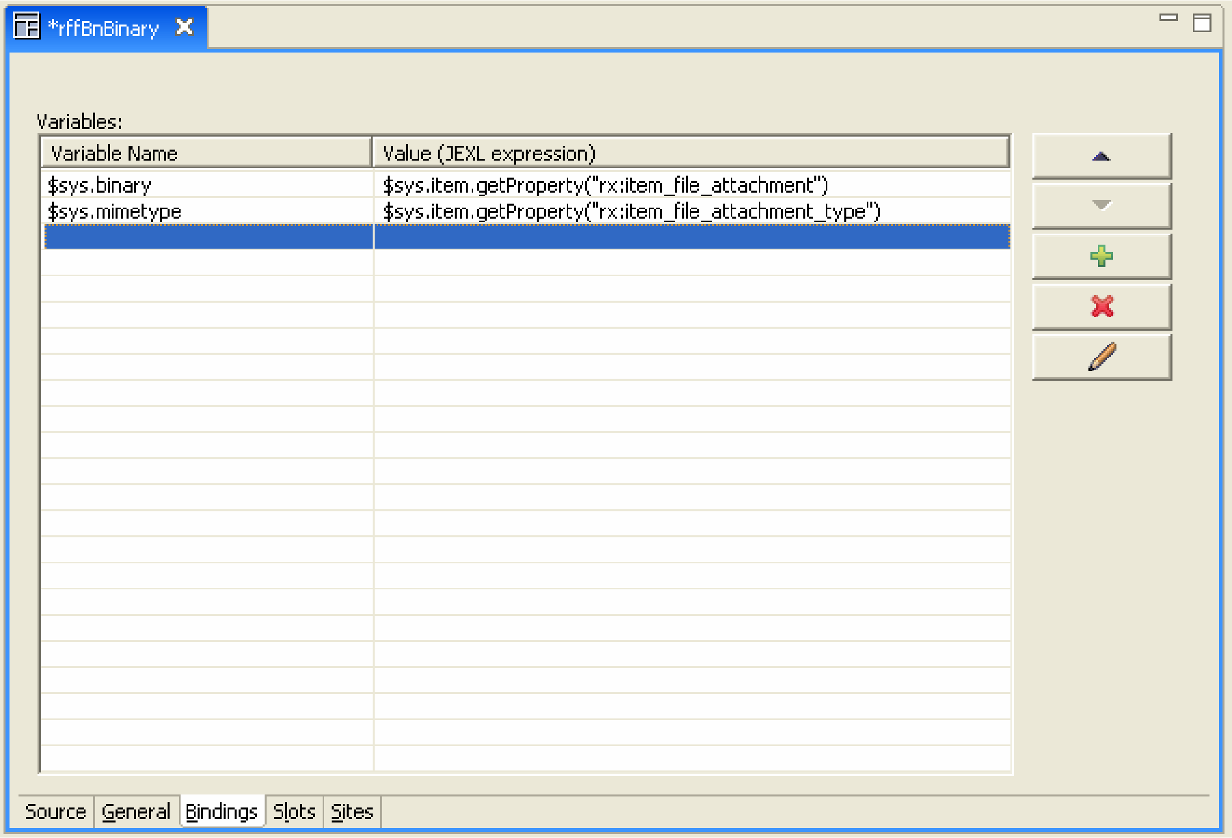
This code defines the macro “displayfield” which requires one argument, “fieldname”, In this case, the system uses a method of another internally-defined binding (the getProperty method of the $sys.item binding) to retrieve the value of the field specified. The value is returned as a string.

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.



Variables:

Variable Name	Value (JEXL expression)
$sys.binary	$sys.item.getProperty("rx:item_file_attachment")
$sys.mimetype	$sys.item.getProperty("rx:item_file_attachment_type")

Figure 103: 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.

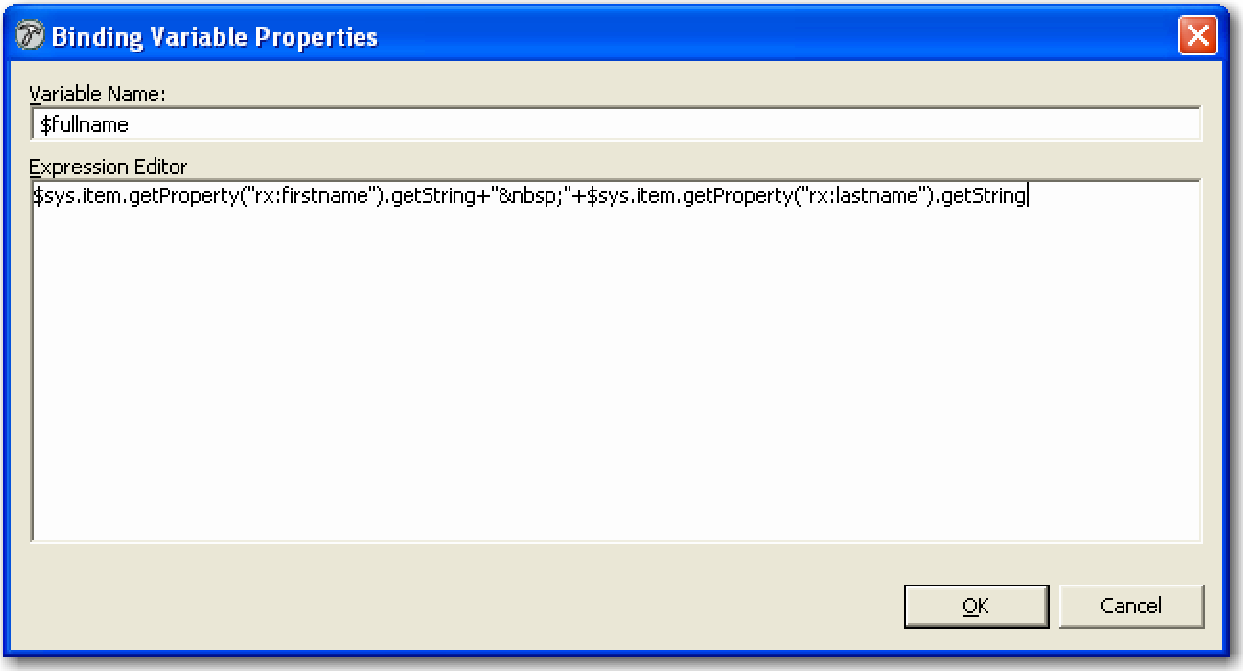
   b) Enter *$sys.item.getProperty("rx:firstname").getString*.

   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:



$fullname

```
$sys.item.getProperty("rx:firstname").getString+"&nbsp;"+$sys.item.getProperty("rx:lastname").getString
```

Figure 104: $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 variable `$sys.binary` (mapped to the value `$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.

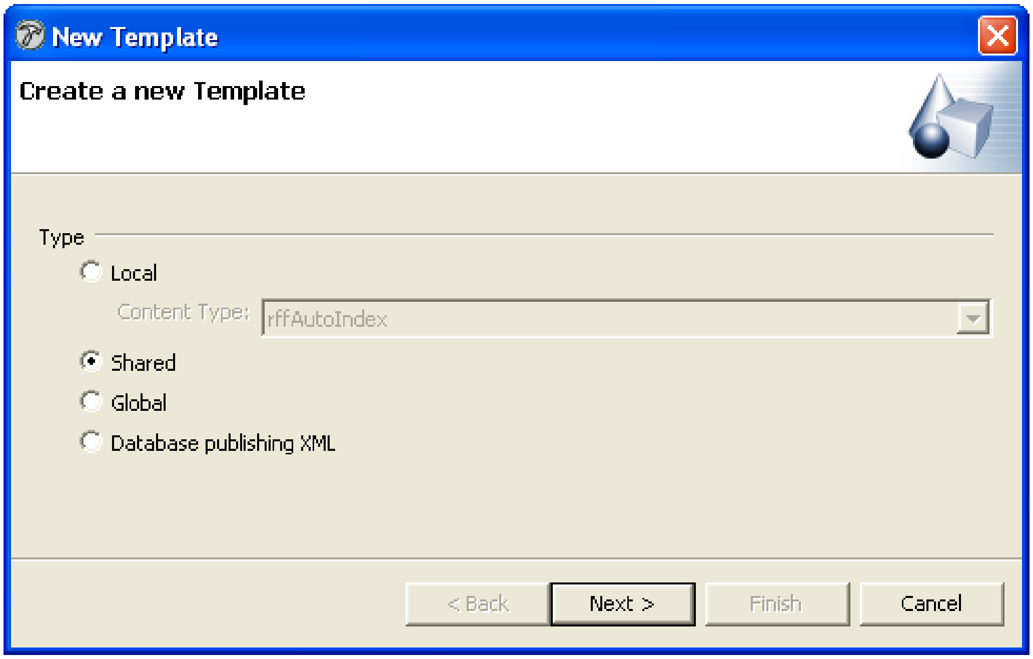


Figure 105: Creating a Binary Template as a Shared Template

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.

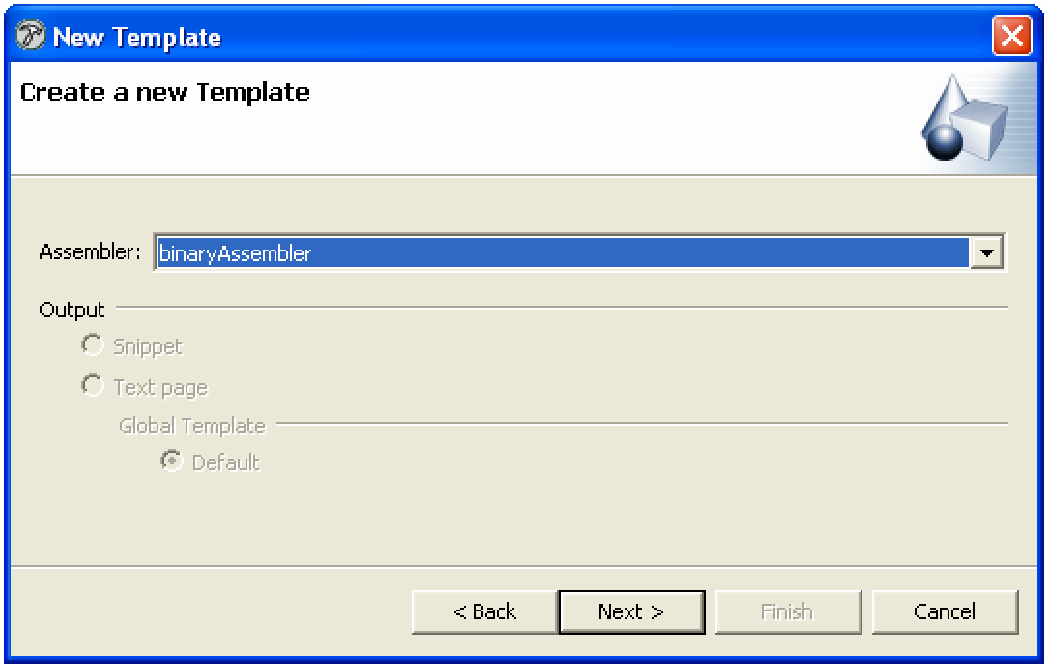


Figure 106: Choosing the binary Assembler for the Binary Template

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 Bindings 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.

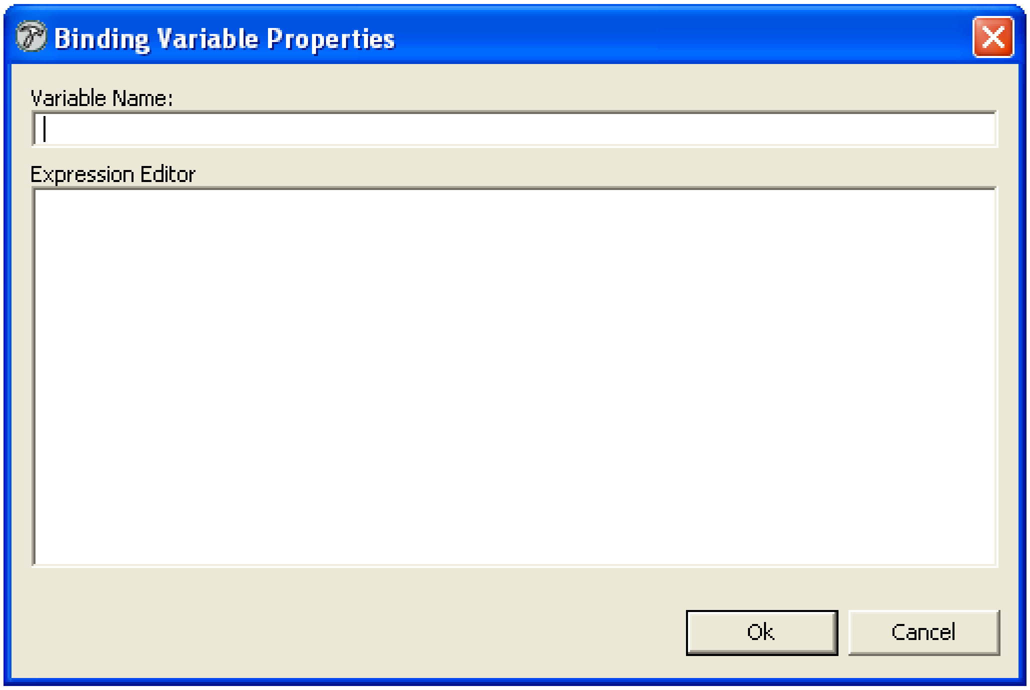
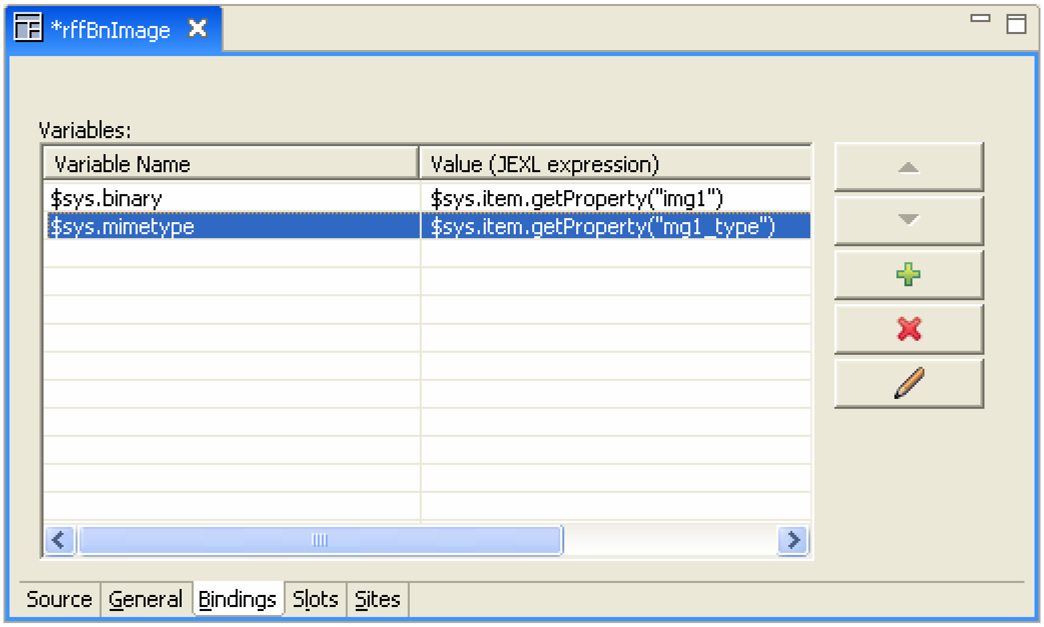


Figure 107: Binding Variable Properties dialog

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.



Variables:

Variable Name	Value (JEXL expression)
$sys.binary	$sys.item.getProperty("img1")
$sys.mimetype	$sys.item.getProperty("mg1_type")

Figure 108: 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

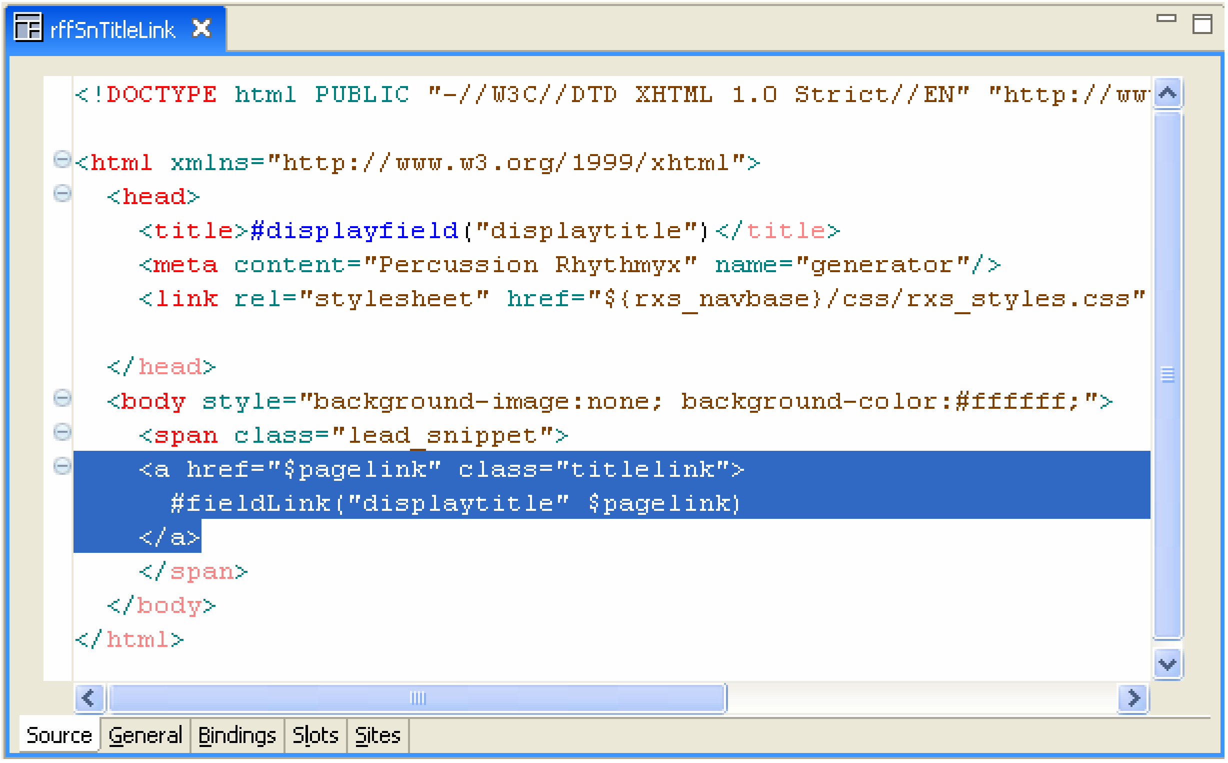
    Click the ▶ button to move these Content Types to the Associated Content Types field.
11. Click the [**Finish**] button.

    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 131).
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.

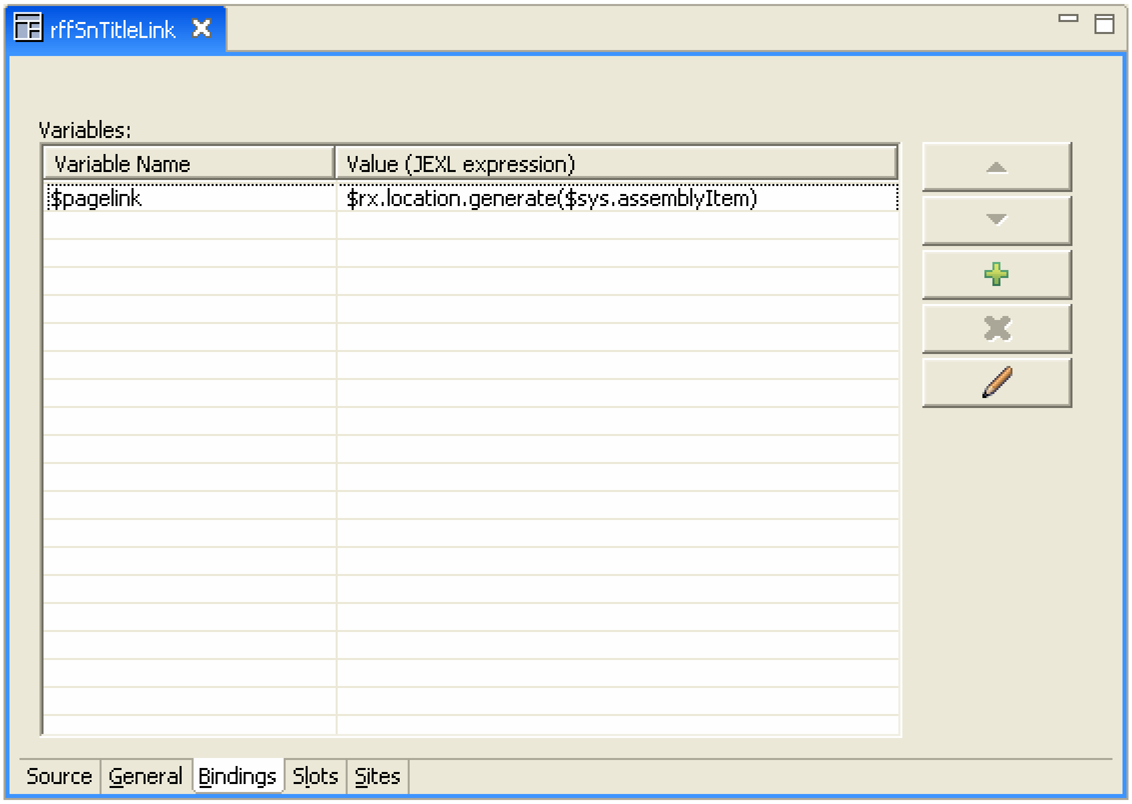
When you are finished, the Snippet Template HTML code resembles the following screenshot:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://ww

<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 109: 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)`.



Variables:

Variable Name	Value (JEXL expression)
$pagelink	$rx.location.generate($sys.assemblyItem)

Figure 110: 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 131).

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

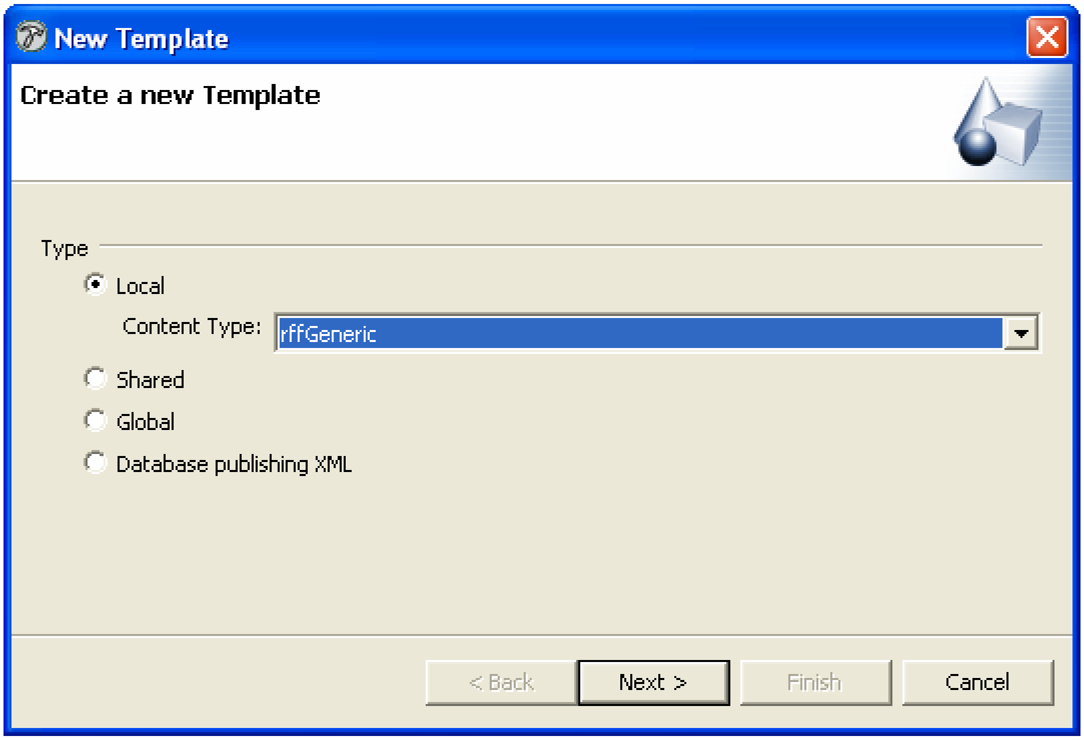
We will assume that the HTML for this Snippet is stored in an HTML file named rffPgEIGeneric.html, which was created during the modeling and design process.

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.



New Template

Create a new Template

Type

● Local

Content Type: rffGeneric

○ Shared

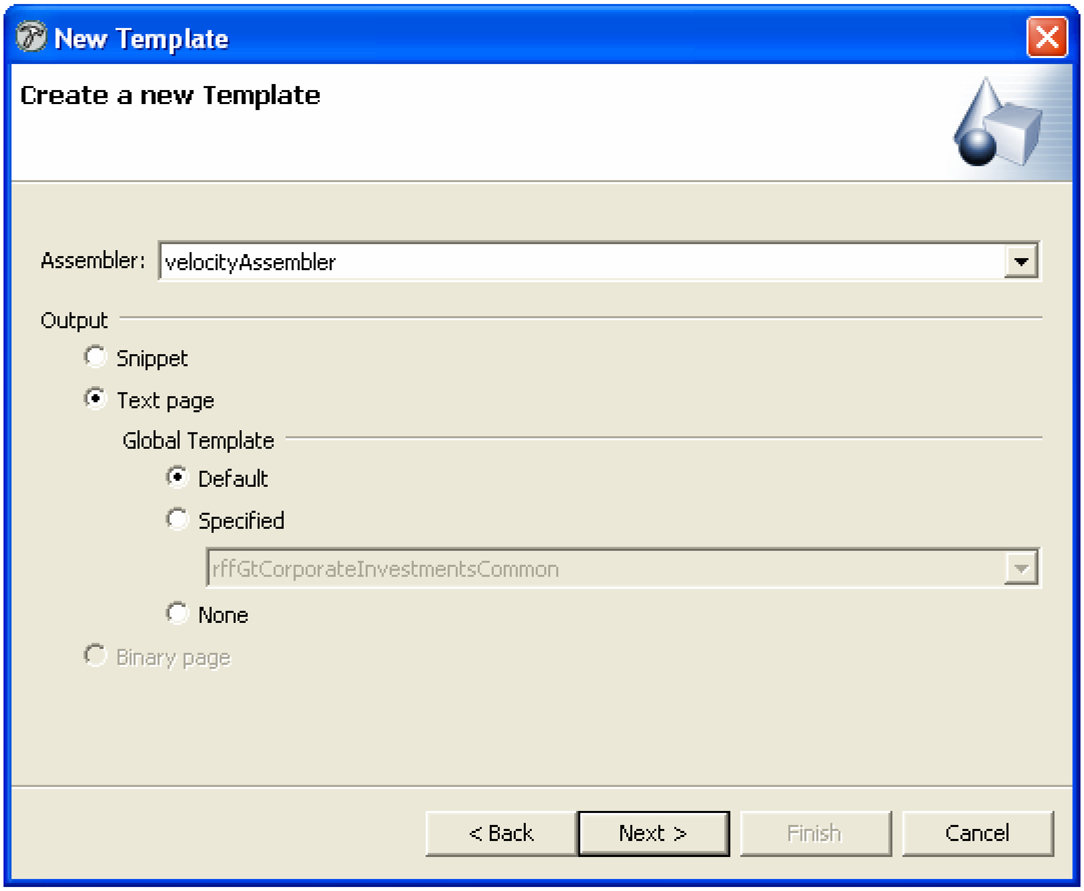
○ Global

○ Database publishing XML

Figure 111: 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.



New Template

Create a new Template

Assembler: velocityAssembler

Output

○ Snippet

● Text page

Global Template

● Default

○ Specified

rffGtCorporateInvestmentsCommon

○ None

○ Binary page

Figure 112: 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.

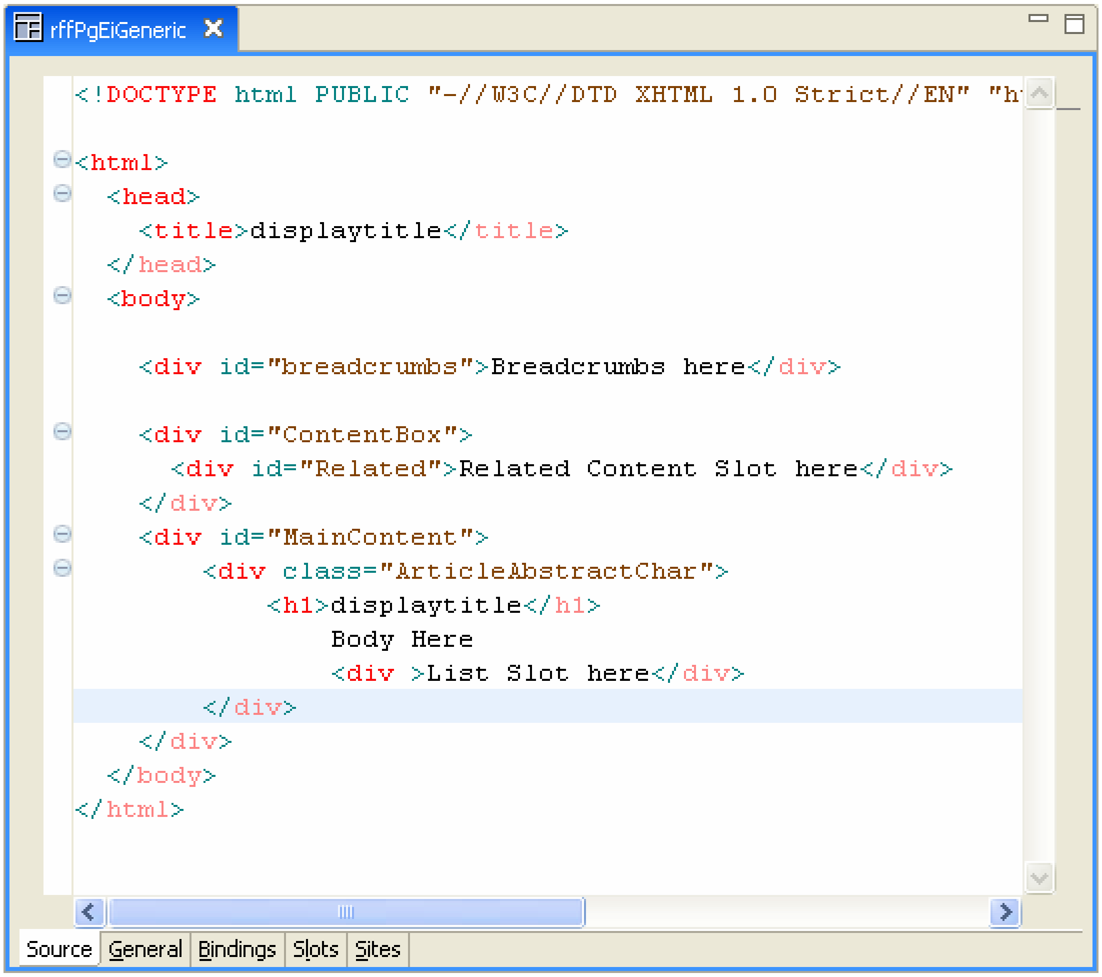
9. Select the List Slot and Sidebar Slot and the click [>] button to add them to the Template.

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

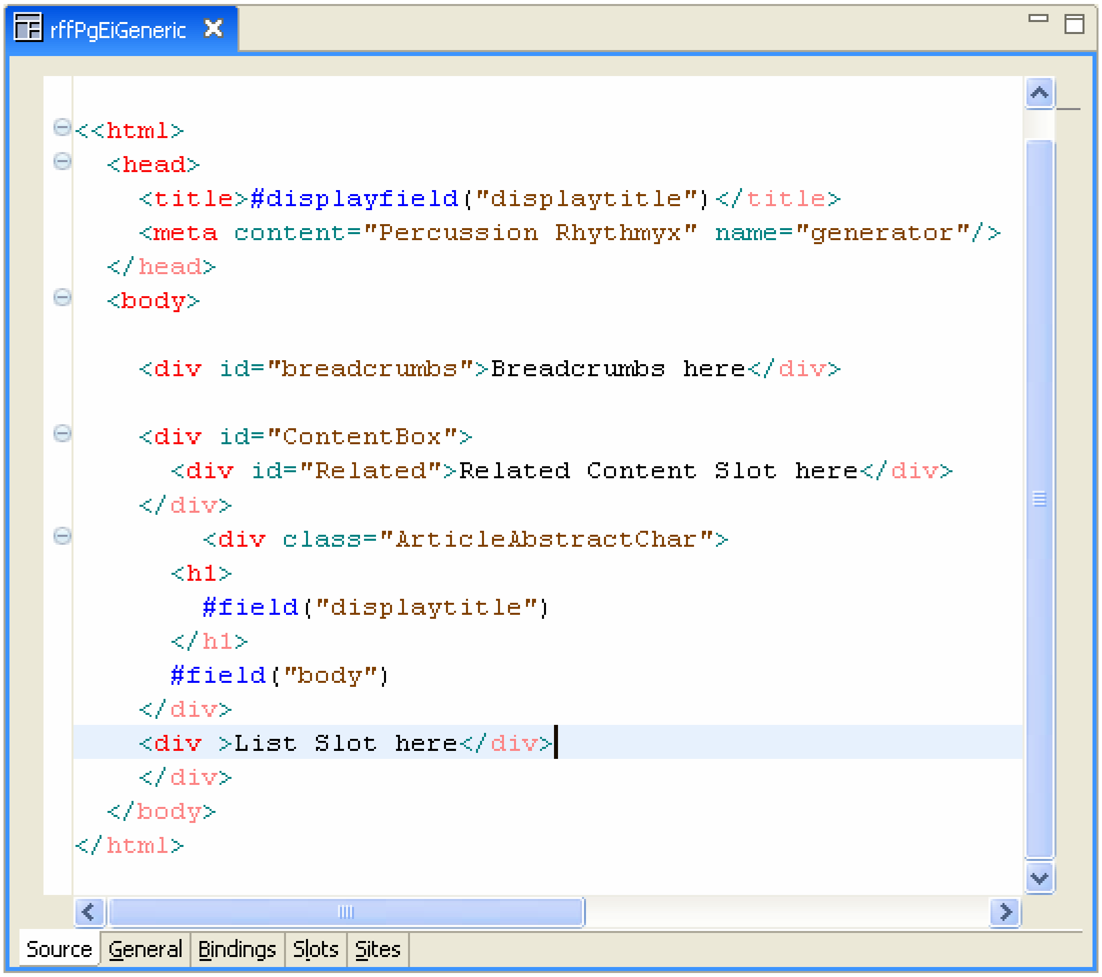
The following screenshot illustrates the rffPgGeneric HTML before adding Velocity markup.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "h

<html>
  <head>
    <title>displaytitle</title>
  </head>
  <body>

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

    <div id="ContentBox">
      <div id="Related">Related Content Slot here</div>
    </div>
    <div id="MainContent">
        <div class="ArticleAbstractChar">
            <h1>displaytitle</h1>
                Body Here
                <div >List Slot here</div>
        </div>
    </div>
  </body>
</html>
```

Figure 113: 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 131). 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 114: HTML of the rffPgEIGeneric Template with field macros

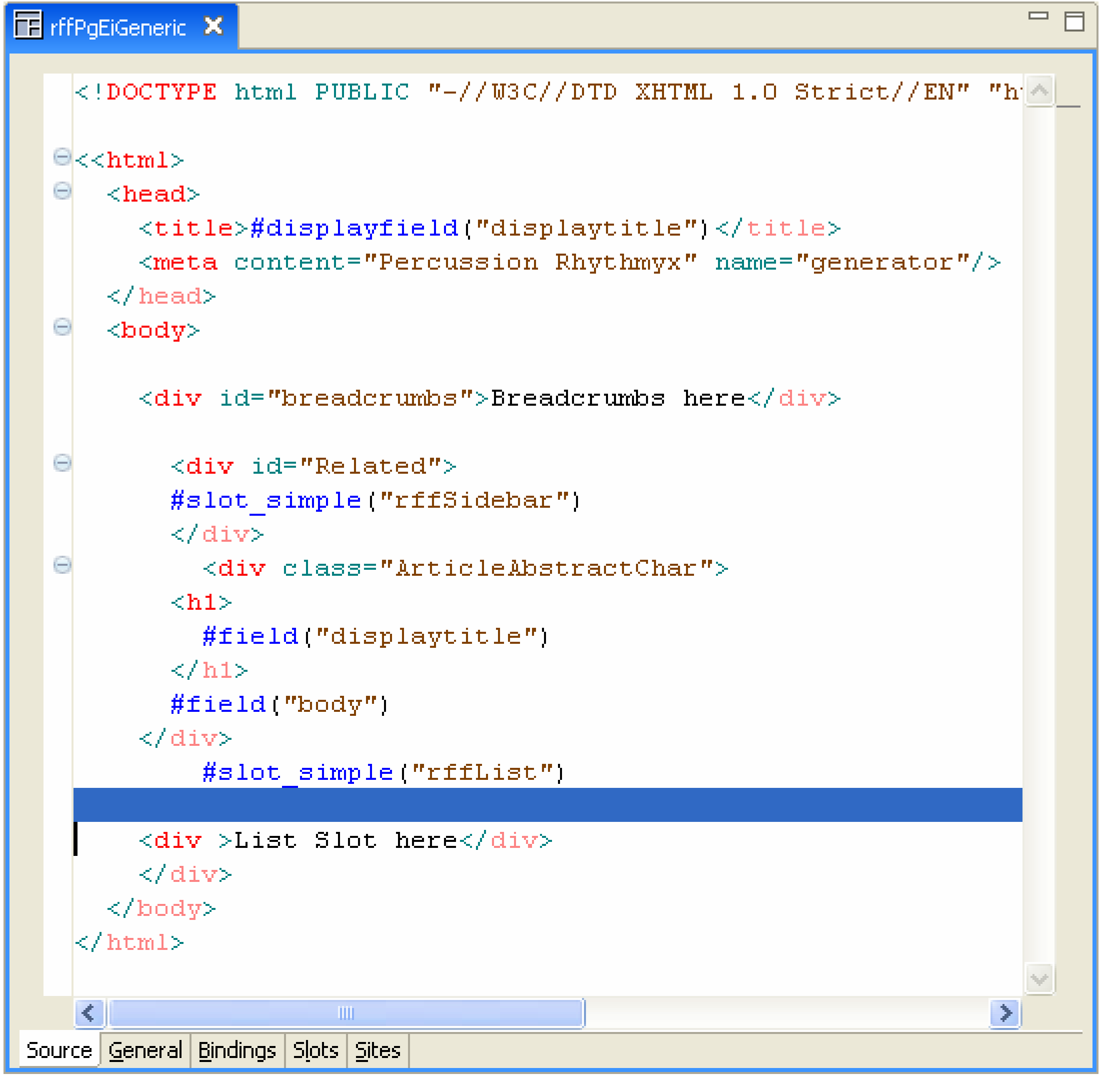
Rhythmyx includes three predefined macros for Slots. The simplest Slot macro is the #slot_simple macro.

```
#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")
```

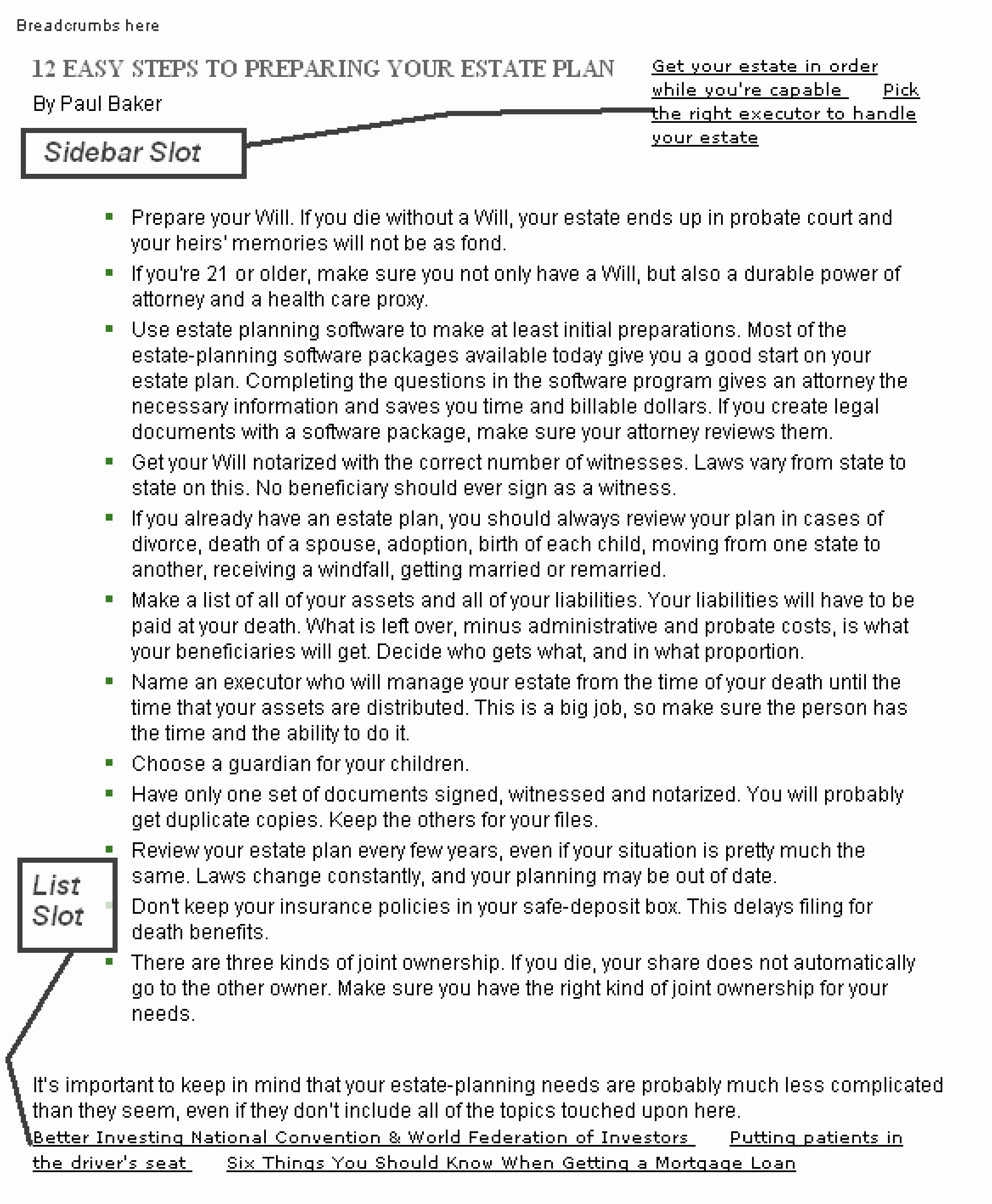
```
#slot_simple(“List Slot”)
```



```
<!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 115: rffPgIEGeneric Template with #slot_simple macros

This markup produces the following output:



Breadcrumbs here

12 EASY STEPS TO PREPARING YOUR ESTATE PLAN

By Paul Baker

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.

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 116: 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:

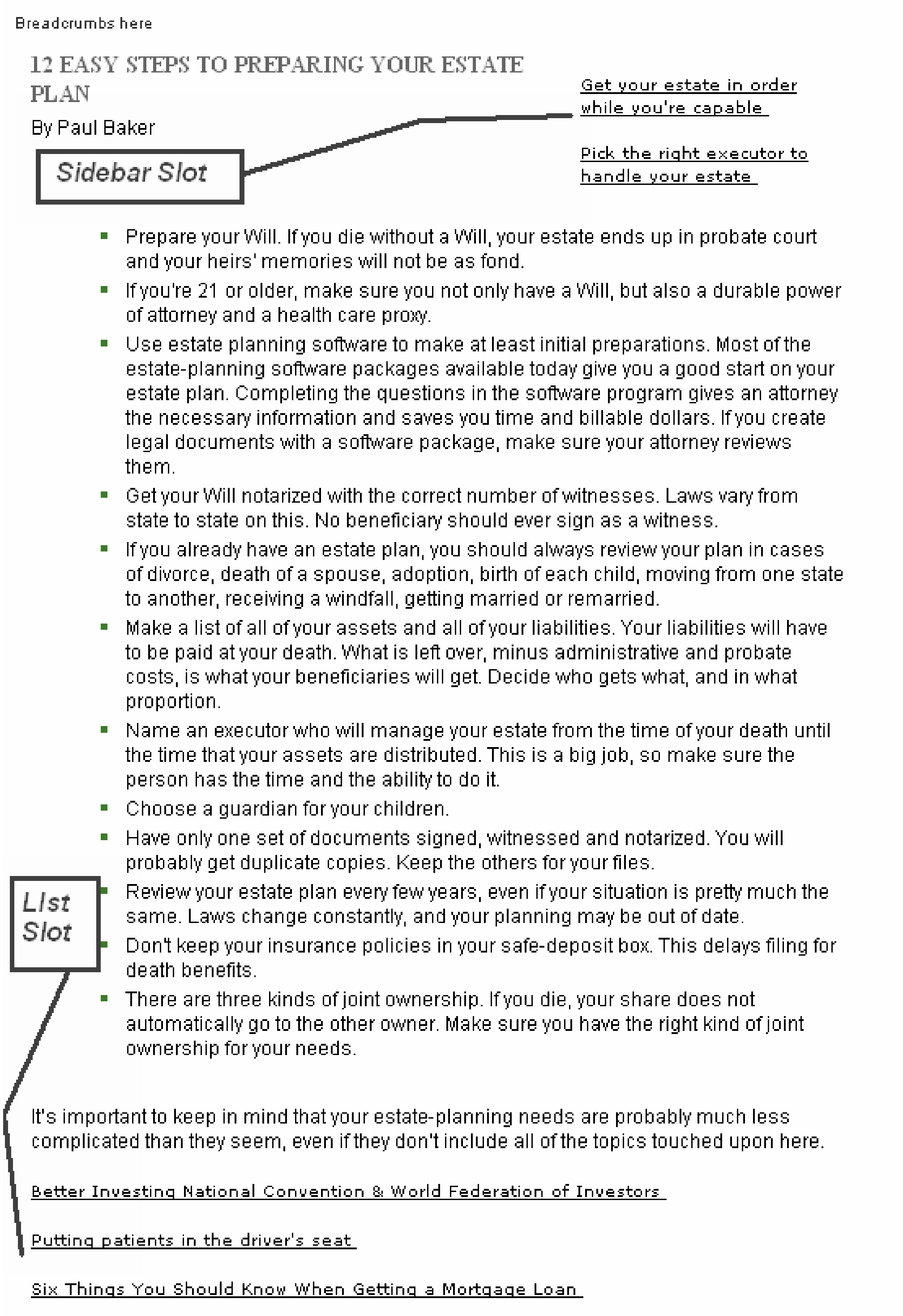
```
#slot_wrapped(slotname startslottext endslottext)
```

where startslotext and endslottext 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

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.

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 117: 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 `startslottext` and `endslottext` attributes of the `#slot_wrapped` macro.)

The richest Slot macro is the #slot macro:

```
#slot(slotname header before after footer params)
```

Where

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

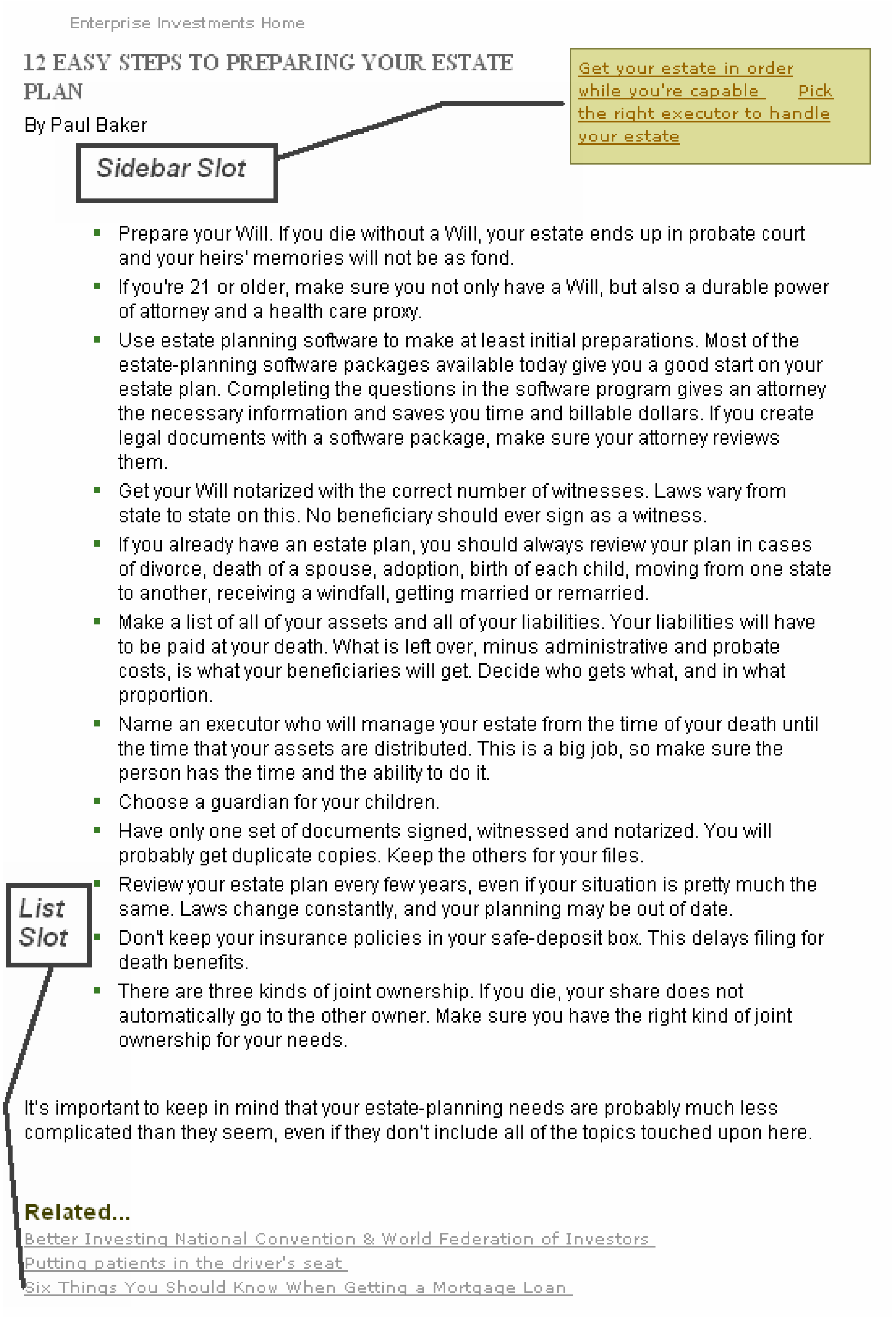
`“<br/>”` is the `aftertext` for each Content Item in the Slot

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

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.

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 118: 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 250) 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:

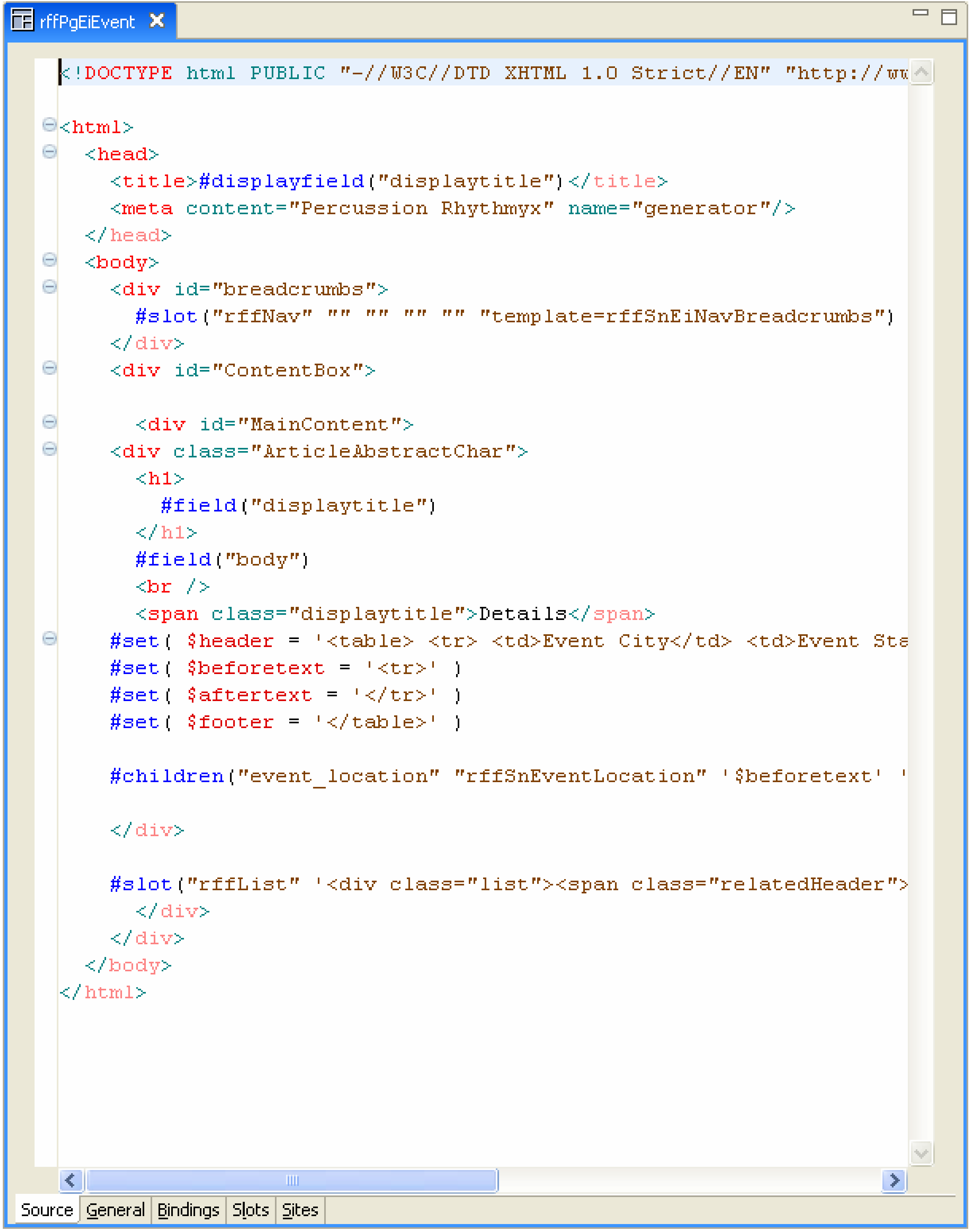
```
      <td>#field("rx:event_address"),
   #field("rx:event_city"),#field("rx:event_state")</td>
      <td> contact:#field("rx:event_contact")</td>
```

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>' )
```

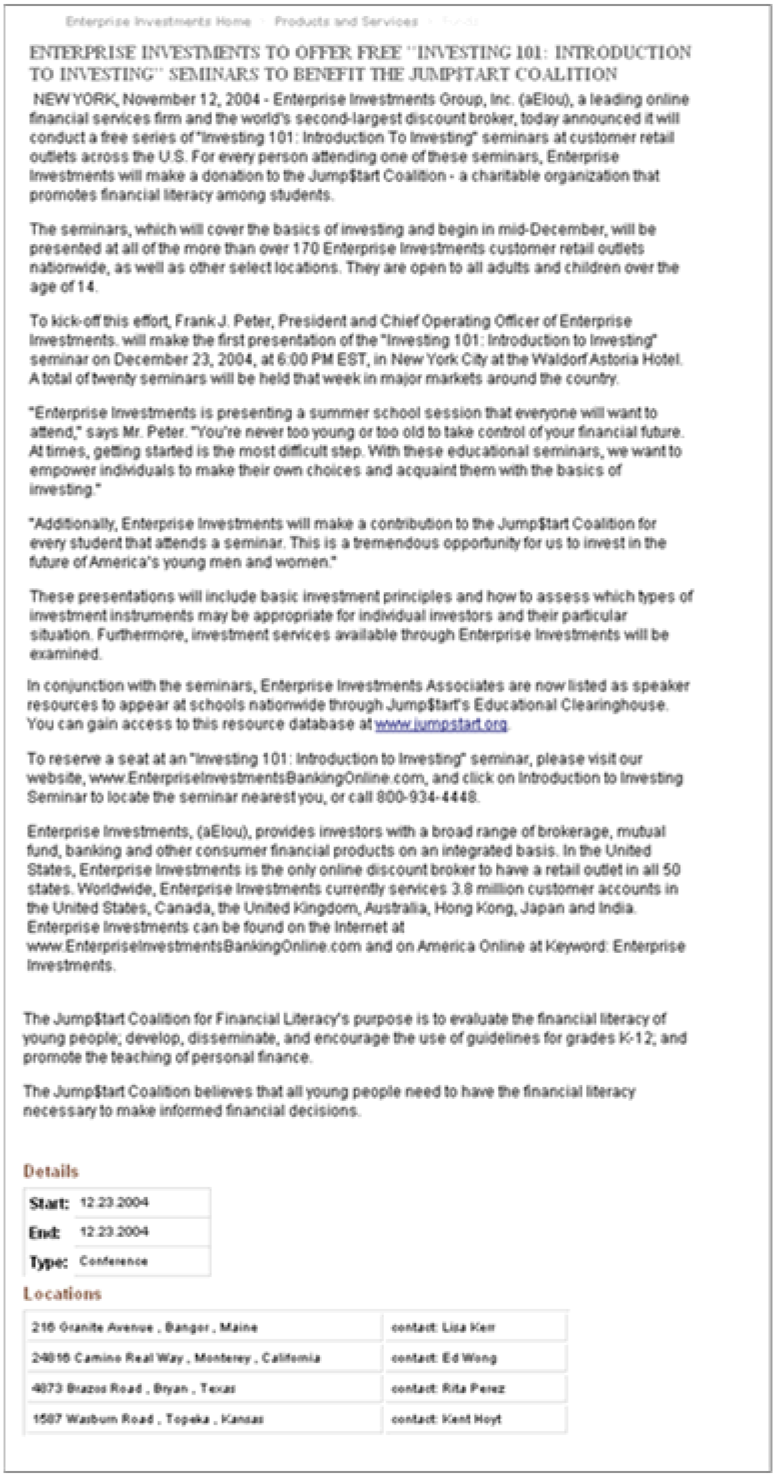
```
        #children("event_location" "rffSnEventLocation" $header
$beforetext $aftertext $footer )
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://ww

<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>
```

Figure 119: rffPgEIEvent Template with #children Macro

This code results in the following output:



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.

To reserve a seat at an "Investing 101: Introduction to Investing" seminar, please visit our website, 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 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

Start:	12.23.2004
End:	12.23.2004
Type:	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 120: Preview of Event Content Item showing child data table

Adding Paging Support

When a Content Item includes a large block of text or when a Slot includes a large number of related Content Items, you may want to implement paging support to break up the output HTML page into several pages rather than outputting one long page.

Paging support can be added for both Content Item fields and for Slots, but paging should only be implemented once in a Template. In other words, you should not implement paging support for a field and for a Slot on the same Template, nor should you implement paging support for more than one field or more than one Slot on a Template. More than one instance of paging on a Template will result in inconsistent results on the output pages.

Content Items with pagination can only be previewed in the Preview Context; preview in publish Contexts returns errors. (Pages are published correctly.)

Implementing Paging Support for a Field

Paging is supported in rich-text fields (fields maintained using the sys_EditLive control). Paging support should only be added to one field per Content Type. Adding paging support to multiple fields may result in inconsistent behavior of HTML pages that include pagination.

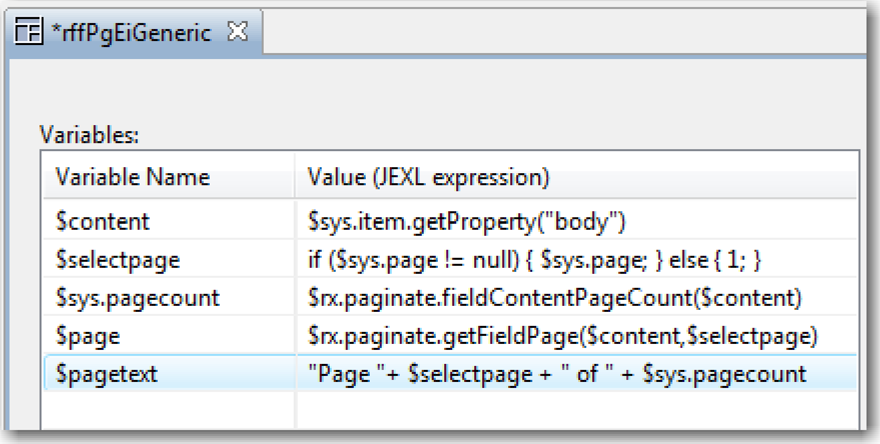
To add paging support to a field, add paging bindings to the Page Template object and paging markup to the Template source code.

Paging for a field requires the following bindings:

Binding Variable	Description	Value
$content	Retrieves the content of the paginated field for later processing.	$sys.item.getProperty("fieldname"); for example, $sys.item.getProperty("body") NOTE: The value should NOT include a conversion to a string value (in other words, DO NOT append ".getString" to the value of the binding). The value of this binding is passed as a parameter into functions used in later bindings, and the functions do not take string values.
$selectpage	Page number of the page to be rendered	if ($sys.page != null) { $sys.page; } else { 1; }
$sys.pagecount	Count of total pages of the paginated field to be rendered. Uses the $rx.paginate.fieldContentPageCount function, which returns the number of pages to be generated based on the number of page breaks added to the specified field. This binding variable must be included to trigger pagination processing.	$rx.paginate.fieldContentPageCount ($content)
$page	Page to be rendered. Uses the $rx.paginate.getFieldPage function to return the content of the individual pages.	$rx.paginate.getFieldPage($content, $selectpage)

Binding Variable	Description	Value
$pagetext	Text to be rendered on the output page to provide paging, such as "page 1 of 3"	"Page "+ $selectpage + " of " + $sys.pagecount

The following graphic illustrates an example set of bindings:



Variables:

Variable Name	Value (JEXL expression)
$content	$sys.item.getProperty("body")
$selectpage	if ($sys.page != null) { $sys.page; } else { 1; }
$sys.pagecount	$rx.paginate.fieldContentPageCount($content)
$page	$rx.paginate.getFieldPage($content,$selectpage)
$pagetext	"Page "+ $selectpage + " of " + $sys.pagecount

Figure 121: Example bindings to support field paging

In the Velocity code of the Template, replace the field macro with the following code:

```
$page
#pager($sys.pagecount $sys.page "<" $pagetext ">")
```

For example, in the rffPgEiGeneric Template, to provide pagination support for the body field, replace the macro for the body field:

```
#field("body")
```

When assembling the Content Item, Rhythmyx will break the Content Item up into the specified number of pages.

Implementing Paging Support for a Slot

Paging support can be added to any Slot other than an inline Slot. (An inline Slot that uses pagination will result in errors.) The pagination implementation defines the number of related Content Items that will be included on each page when the Content Item is paginated.

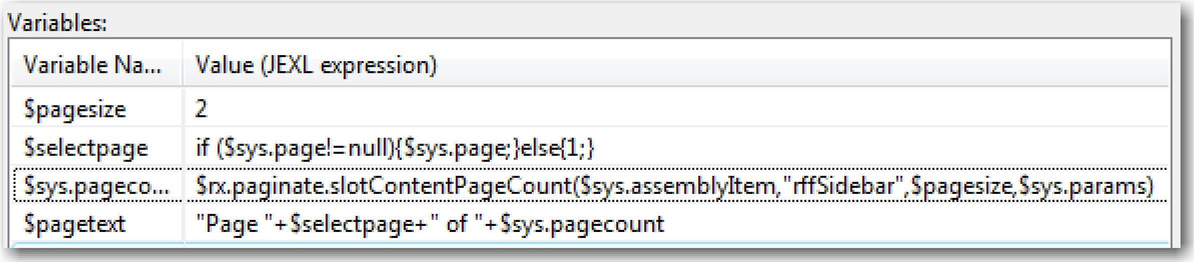
To add paging support for a Slot, add paging bindings to the Page Template object and paging markup to the Template source code.

Paging for a Slot requires the following bindings:

Binding Variable	Description	Value
$pagesize	Defines the number of related Content Items included on each page	A literal numeric value, such as 3 or 5.
$selectpage	Page number of the page to be rendered	if ($sys.page != null) { $sys.page; } else { 1; }

Binding Variable	Description	Value
$sys.pagecount	Count of total pages of the paginated Slot to be rendered. Uses the $rx.paginate.slotContentPageCount function, which returns the number of pages to be generated based on the value in the $pagesize binding variable. This binding variable must be included to trigger pagination processing.	$rx.paginate.slotContentPageCount($sys.assemblyItem,"rffList",$pagesize,$sys.params)
$pagetext	Text to be rendered on the output page to provide paging, such as "page 1 of 3"	"Page "+ $selectpage + " of " + $sys.pagecount

The following graphic illustrates an example set of bindings:



Variables:

Variable Na...	Value (JEXL expression)
$pagesize	2
$selectpage	if ($sys.page!=null){$sys.page;}else{1;}
$sys.pageco...	$rx.paginate.slotContentPageCount($sys.assemblyItem,"rffSidebar",$pagesize,$sys.params)
$pagetext	"Page "+$selectpage+" of "+$sys.pagecount

pr

In the Velocity code of the Template, replace the Slot macro with the following code:

```
   #slot_page("Slotname" "Header" "Beforetext" "Aftertext" "Footer"
"Params" "ItemsPerPage" "Page Number")
#pager($sys.pagecount $selectpage "<" $pagetext ">")
```

For example, in the rffPgEiGeneric Template, to provide paging support for the Sidebar Slot, replace the

```
#slot("rffSidebar" '<div class="SideContent">' '' '<br/>' '</div>' '')
```

with the code:

```
   #slot_page("rffSidebar" $start_slot $start_snippet $end_snippet
$end_slot '' $pagesize $selectpage)
#pager($sys.pagecount $selectpage "<" $pagetext ">")
```

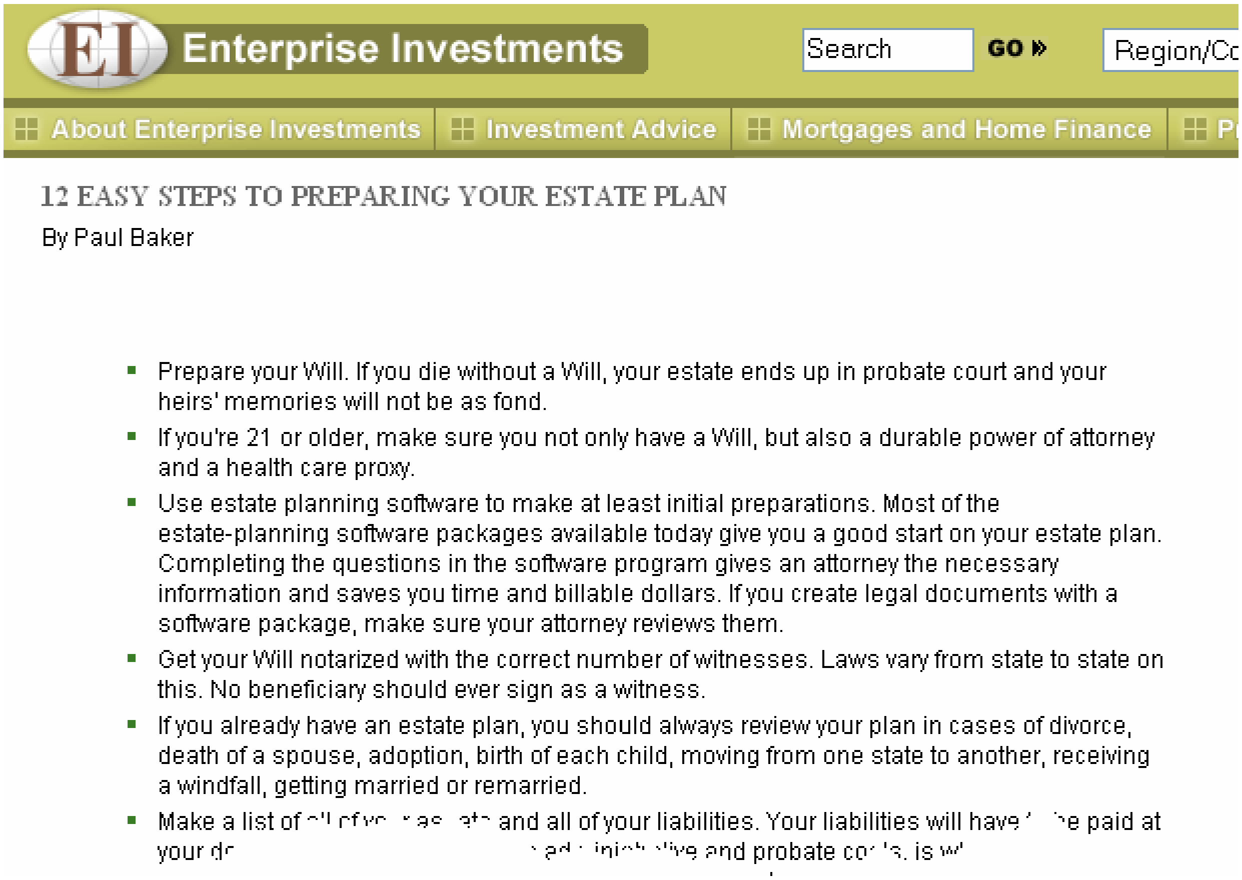
When assembling the Content Item, Rhythmyx will chunk the Content Items in the Sidebar Slot into groups of the specified size (2 Content Items per group in this case), and publish the number of HTML pages required to include the complete list of Content Items. For example, in this case, if the Sidebar Slot includes only one or two Content Items, only one HTML page will be published. If the Sidebar Slot includes five Content Items, three HTML pages will be published, two each with two Content Items in the Slot, and the last page with one. Of the Sidebar Slot include eight Content Items, four HTML pages will be published, each with two Content Items in the Sidebar Slot.

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

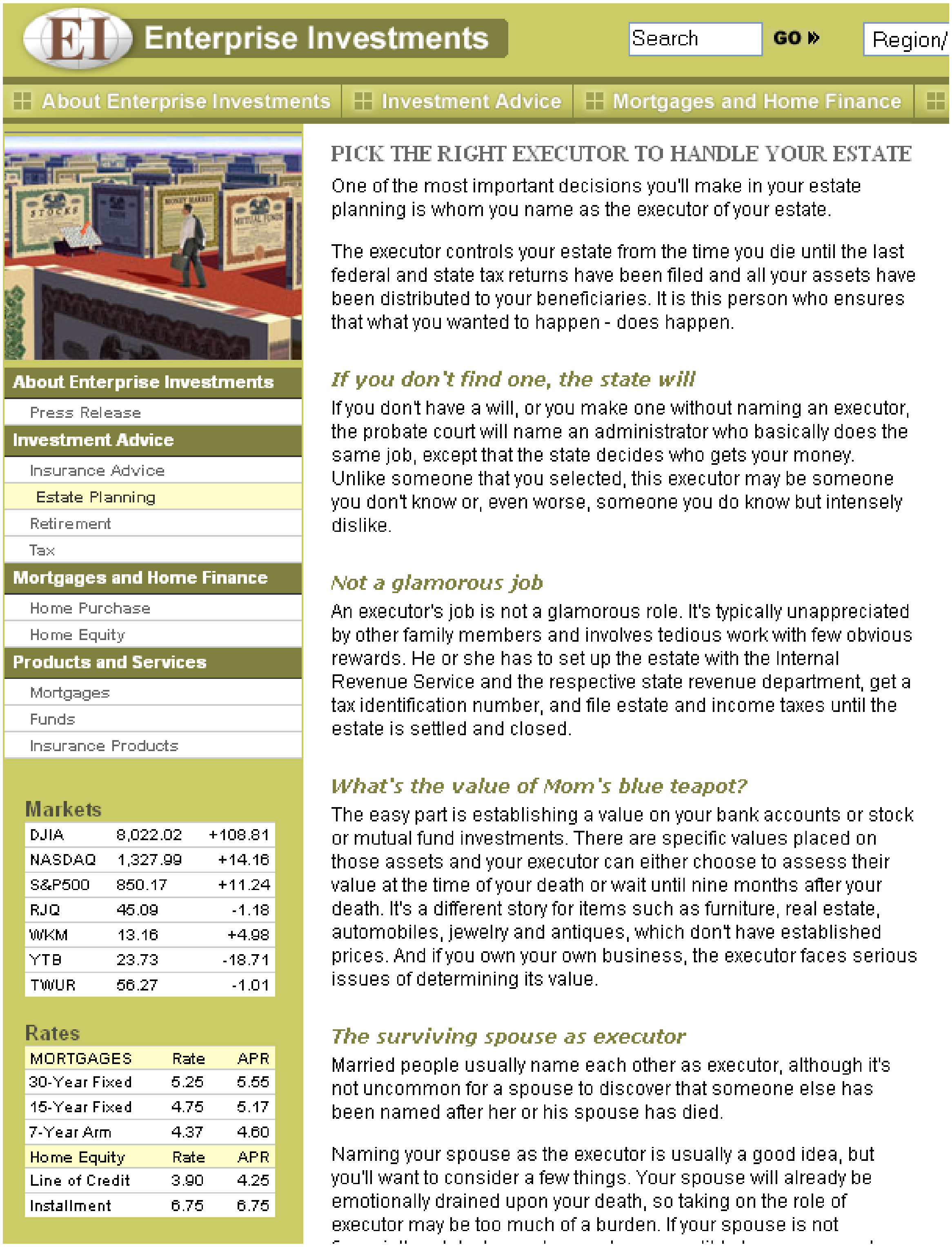
By Paul Baker

- 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 d[illegible] administrative and probate costs, is w[illegible]

Figure 122: 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.



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

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

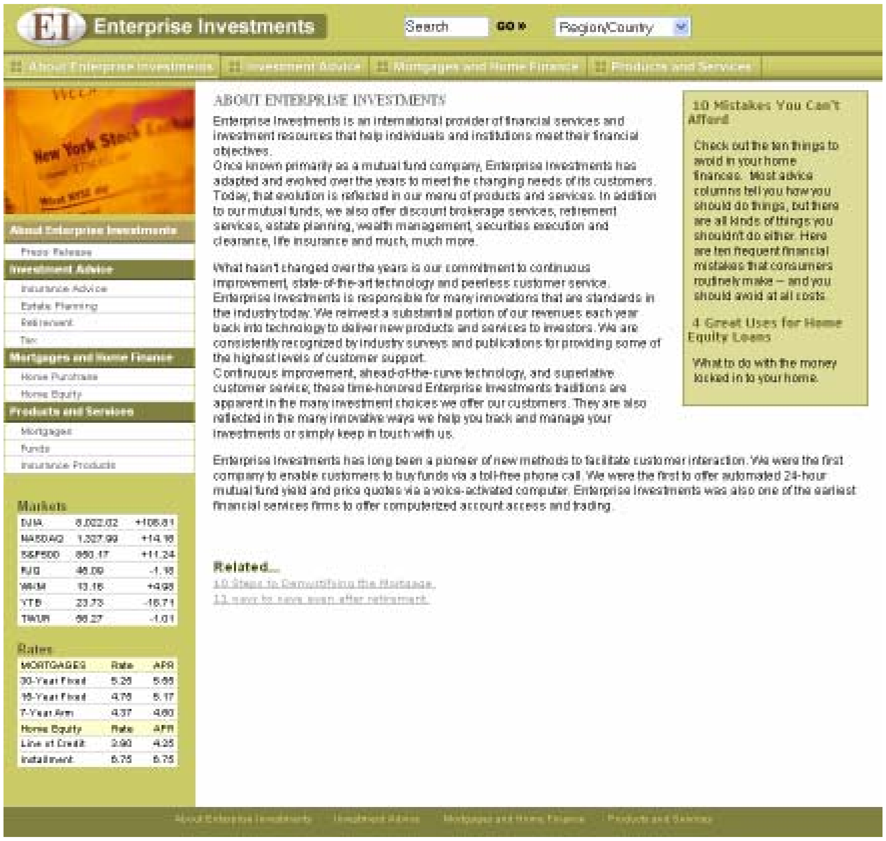
Married people usually name each other as executor, although it's not uncommon for a spouse to discover that someone else has been named after her or his spouse has died.

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

Figure 123: 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".



ABOUT ENTERPRISE INVESTMENTS

Enterprise Investments is an international provider of financial services and investment resources that help individuals and institutions meet their financial objectives.
Once known primarily as a mutual fund company, Enterprise Investments has adapted and evolved over the years to meet the changing needs of its customers. Today, that evolution is reflected in our menu of products and services. In addition to our mutual funds, we also offer discount brokerage services, retirement services, estate planning, wealth management, securities execution and clearance, life insurance and much, much more.

What hasn't changed over the years is our commitment to continuous improvement, state-of-the-art technology and peerless customer service. Enterprise Investments is responsible for many innovations that are standards in the industry today. We reinvest a substantial portion of our revenues each year back into technology to deliver new products and services to investors. We are consistently recognized by industry surveys and publications for providing some of the highest levels of customer support.
Continuous improvement, ahead-of-the-curve technology, and superlative customer service; these time-honored Enterprise Investments traditions are apparent in the many investment choices we offer our customers. They are also reflected in the many innovative ways we help you track and manage your investments or simply keep in touch with us.

Enterprise Investments has long been a pioneer of new methods to facilitate customer interaction. We were the first company to enable customers to buy funds via a toll-free phone call. We were the first to offer automated 24-hour mutual fund yield and price quotes via a voice-activated computer. Enterprise Investments was also one of the earliest financial services firms to offer computerized account access and trading.

Related...
10 Steps to Demystifying the Mortgage.
11 ways to save even after retirement.

10 Mistakes You Can't Afford

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.

4 Great Uses for Home Equity Loans

What to do with the money locked in to your home.

Markets

DJIA	8,022.02	+106.81
NASDAQ	1,327.99	+14.16
S&P500	880.17	+11.24
RJG	46.09	-1.16
WHM	13.16	+4.98
YTB	23.73	-16.71
TWUR	56.27	-1.01

Rates

MORTGAGES	Rate	APR
30-Year Fixed	5.26	5.66
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 124: 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.

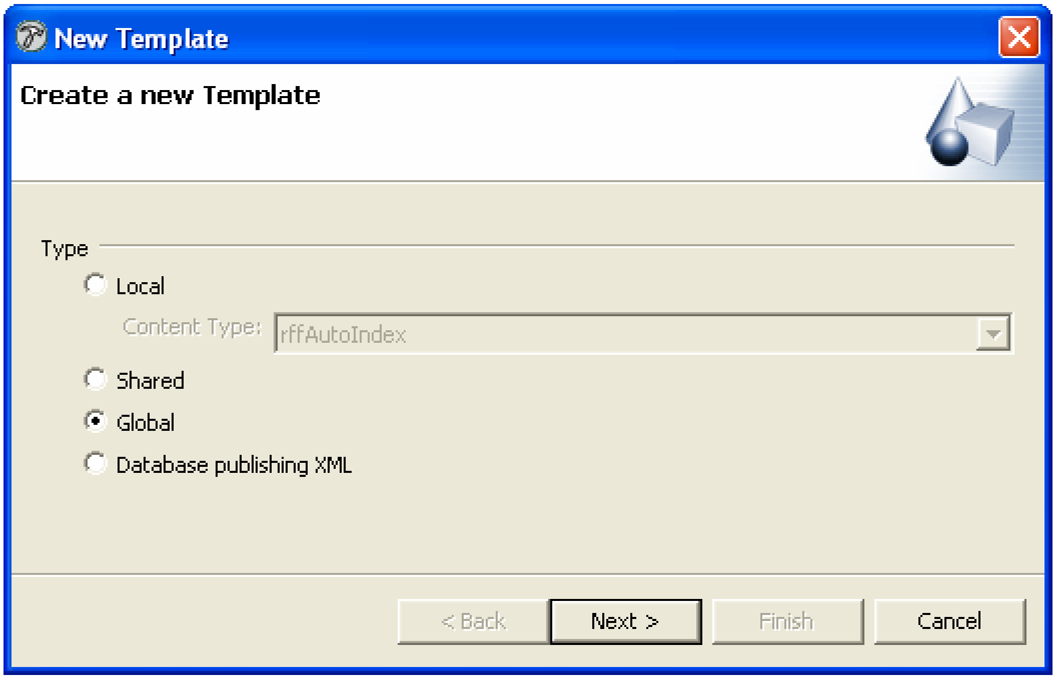
Creating the Global Template Object in the Rhythmyx Workbench

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.



New Template

Create a new Template

Type

○ Local

Content Type: rffAutoIndex

○ Shared

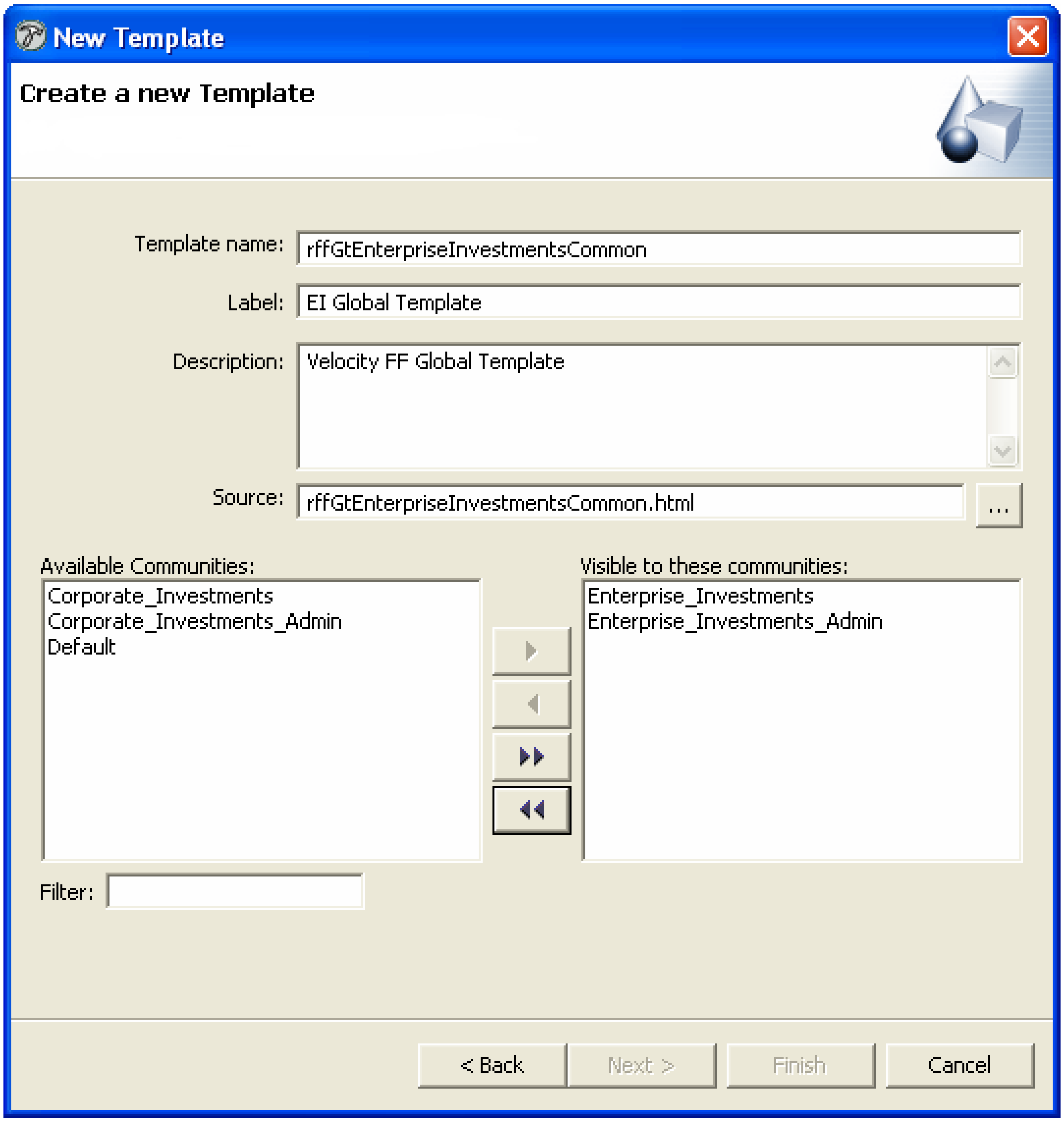
● Global

○ Database publishing XML

Figure 125: 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.



Template name: rffGtEnterpriseInvestmentsCommon

Label: EI Global Template

Description: Velocity FF Global Template

Source: rffGtEnterpriseInvestmentsCommon.html

Available Communities:
Corporate_Investments
Corporate_Investments_Admin
Default

Visible to these communities:
Enterprise_Investments
Enterprise_Investments_Admin

Filter:

Figure 126: 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.

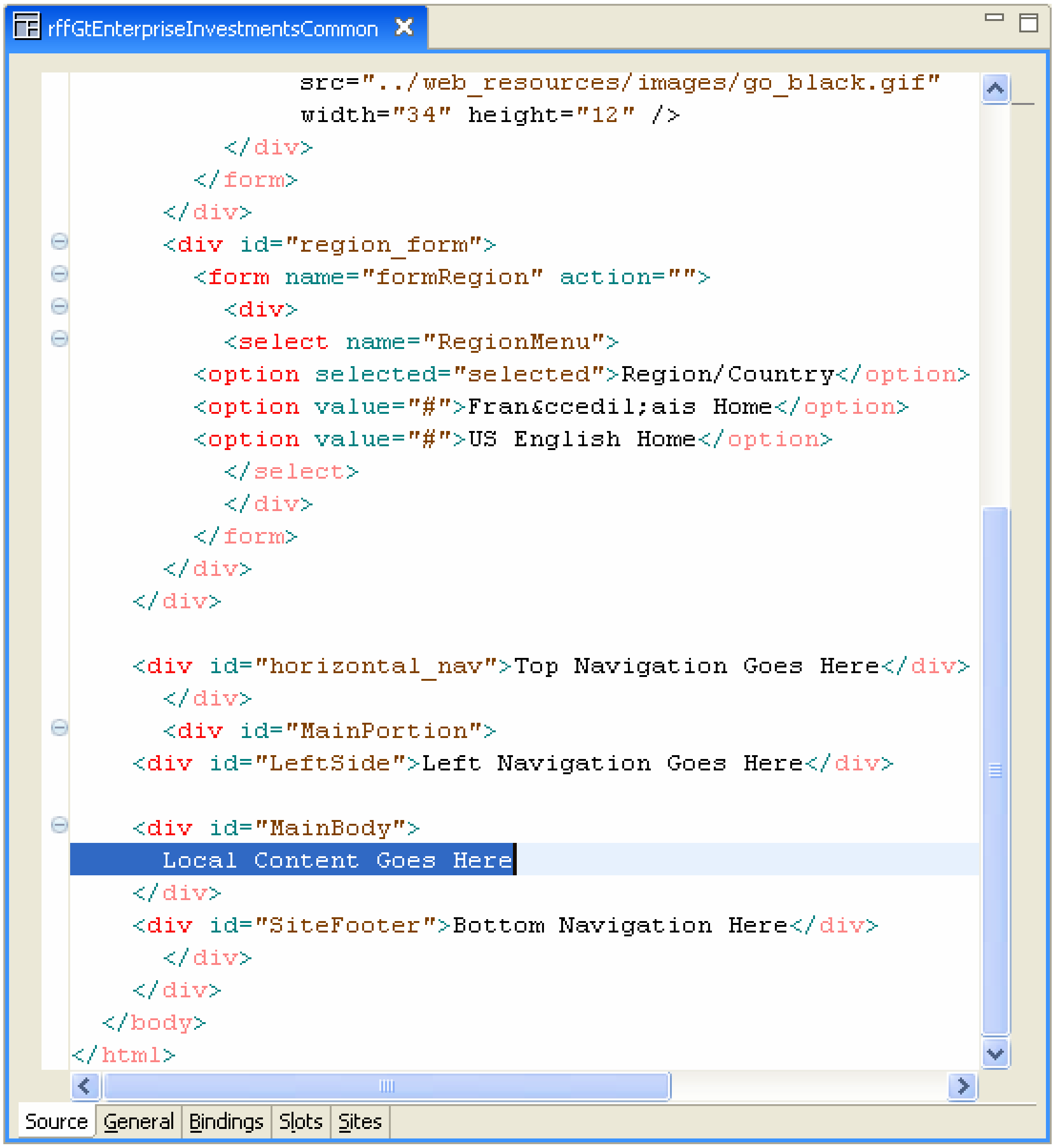
The Rhythmyx Workbench displays the Contained Slots dialog of the Template wizard.

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

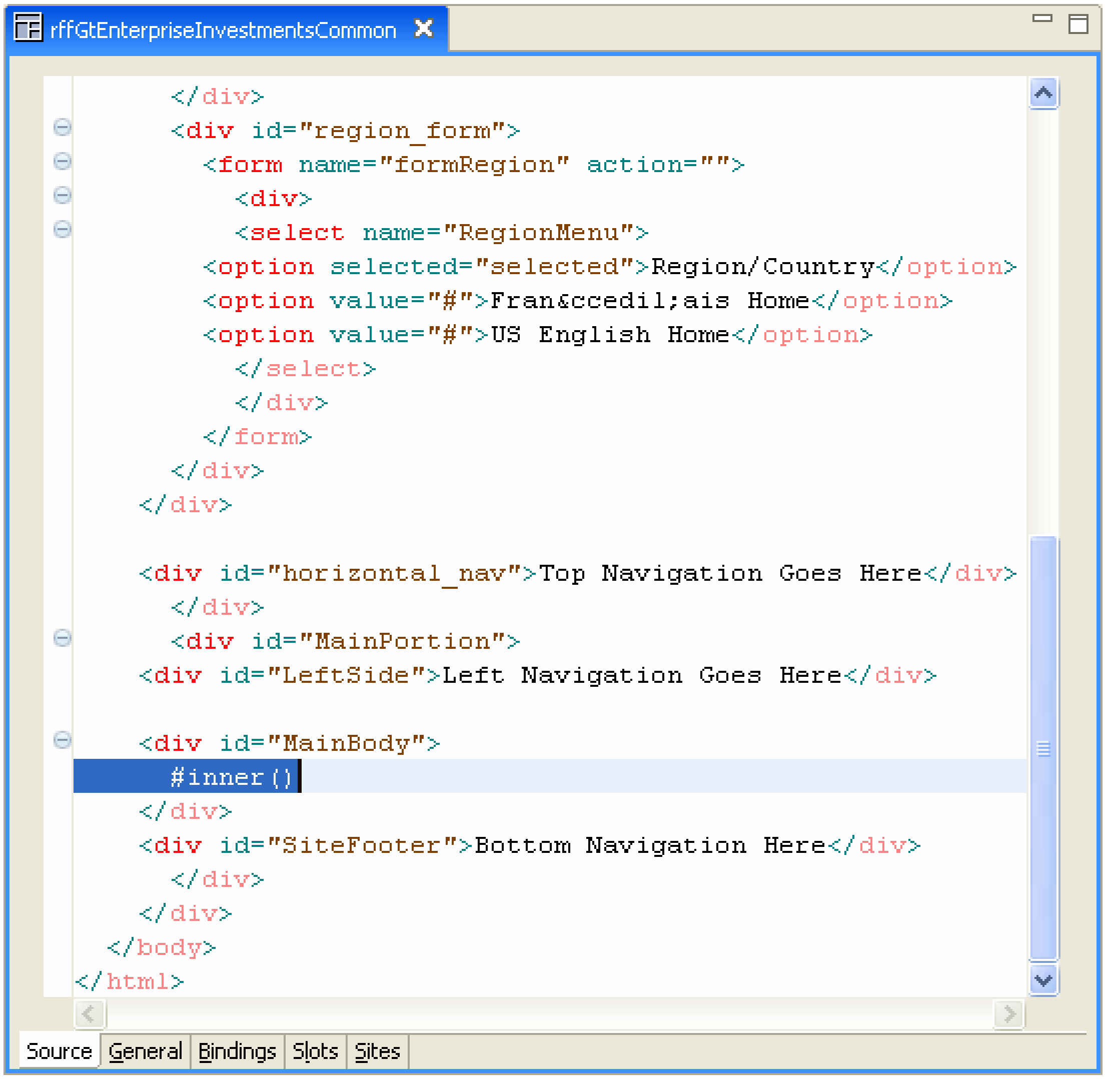
In the raw markup of the Enterprise Investments Global Template, we have inserted a note "Local Content Goes Here" to denote the location of the local content in the Global Template:



```
               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 127: rffGtEnterpriseInvestmentsCommon Template HTML with location of local content highlighted

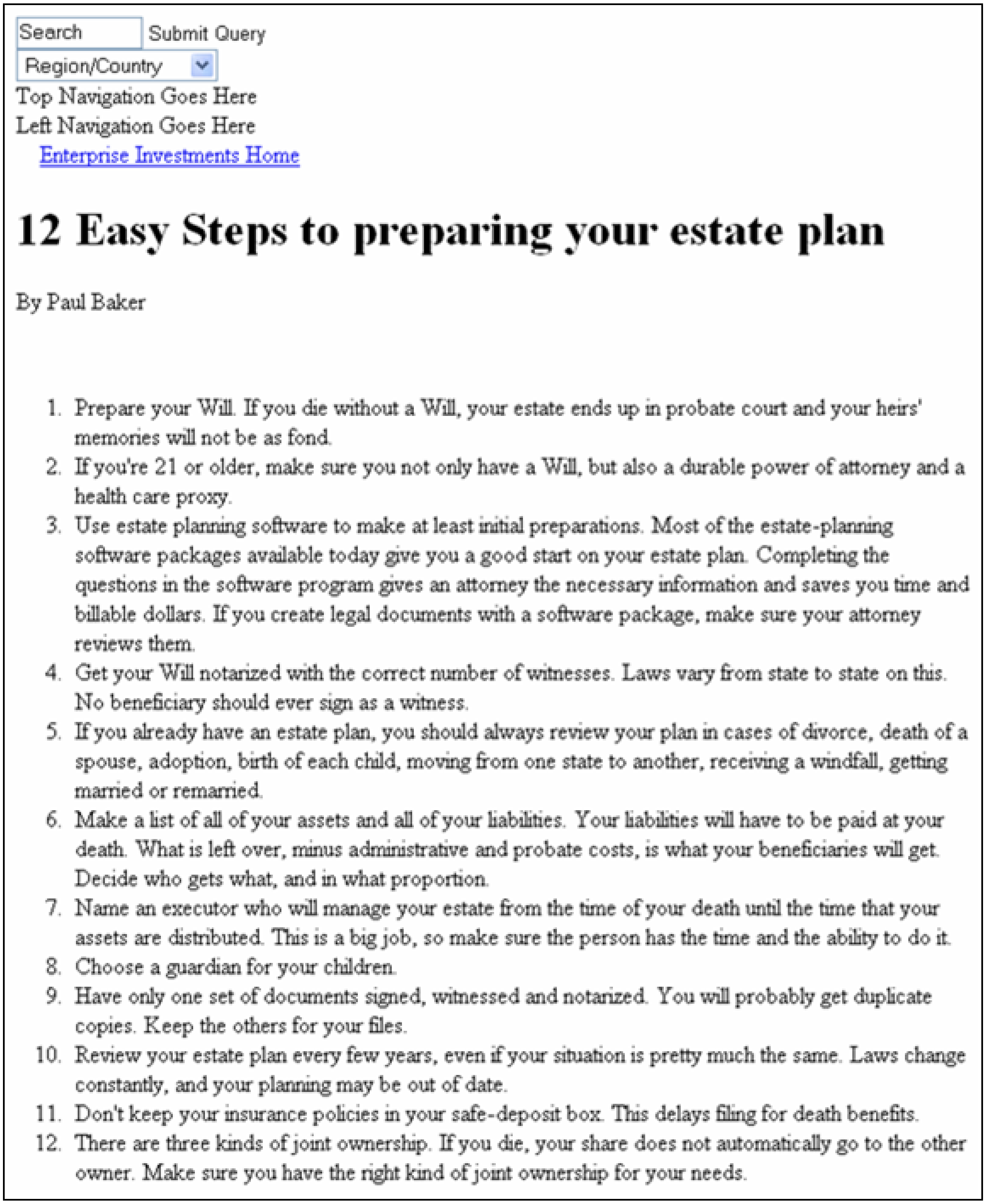
To include the content of a Local Template, use the `#inner` macro. This macro does not include any attributes.



```
        </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 128: 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 129: Preview of rffEnterpriseInvestmentsCommon Template. Locations for Managed Navigation are noted with text.

Adding Managed Navigation to the Global Template

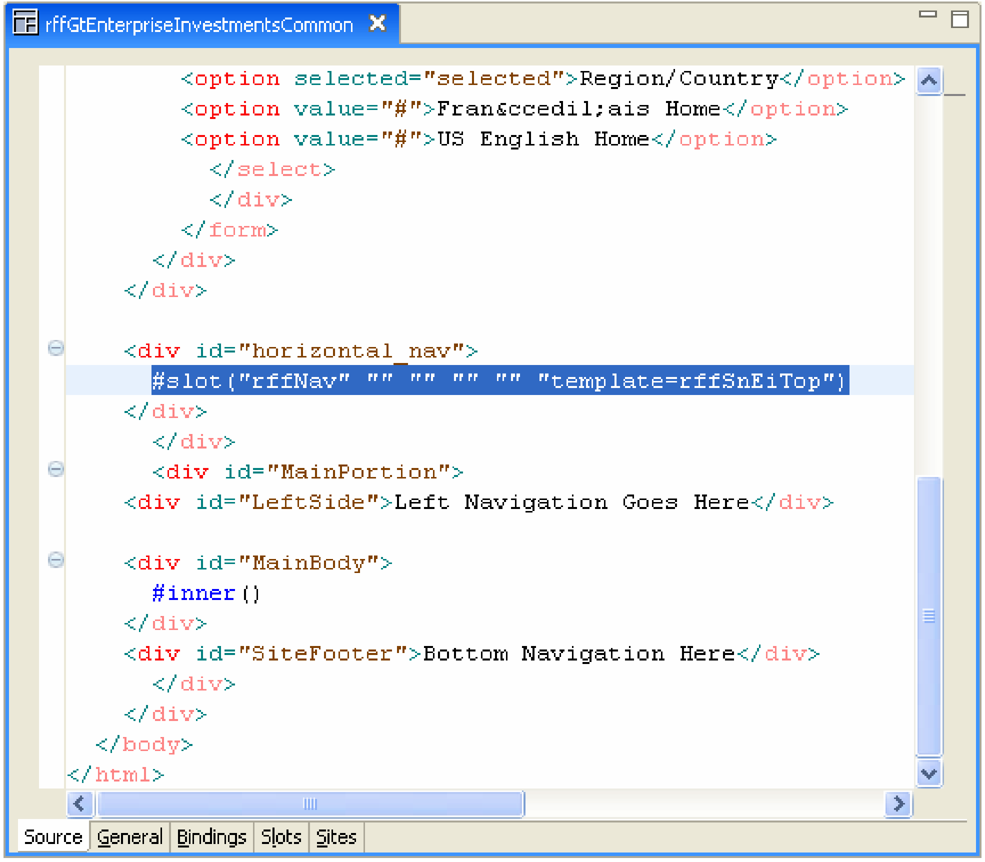
Managed Navigation is a Rhythmyx feature that allows you to create and maintain simple and effective navigation for your site automatically during publishing. The section ***Managed Navigation*** (see page 279) explains how to implement Managed Navigation in detail. For now, we only need to focus on 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:

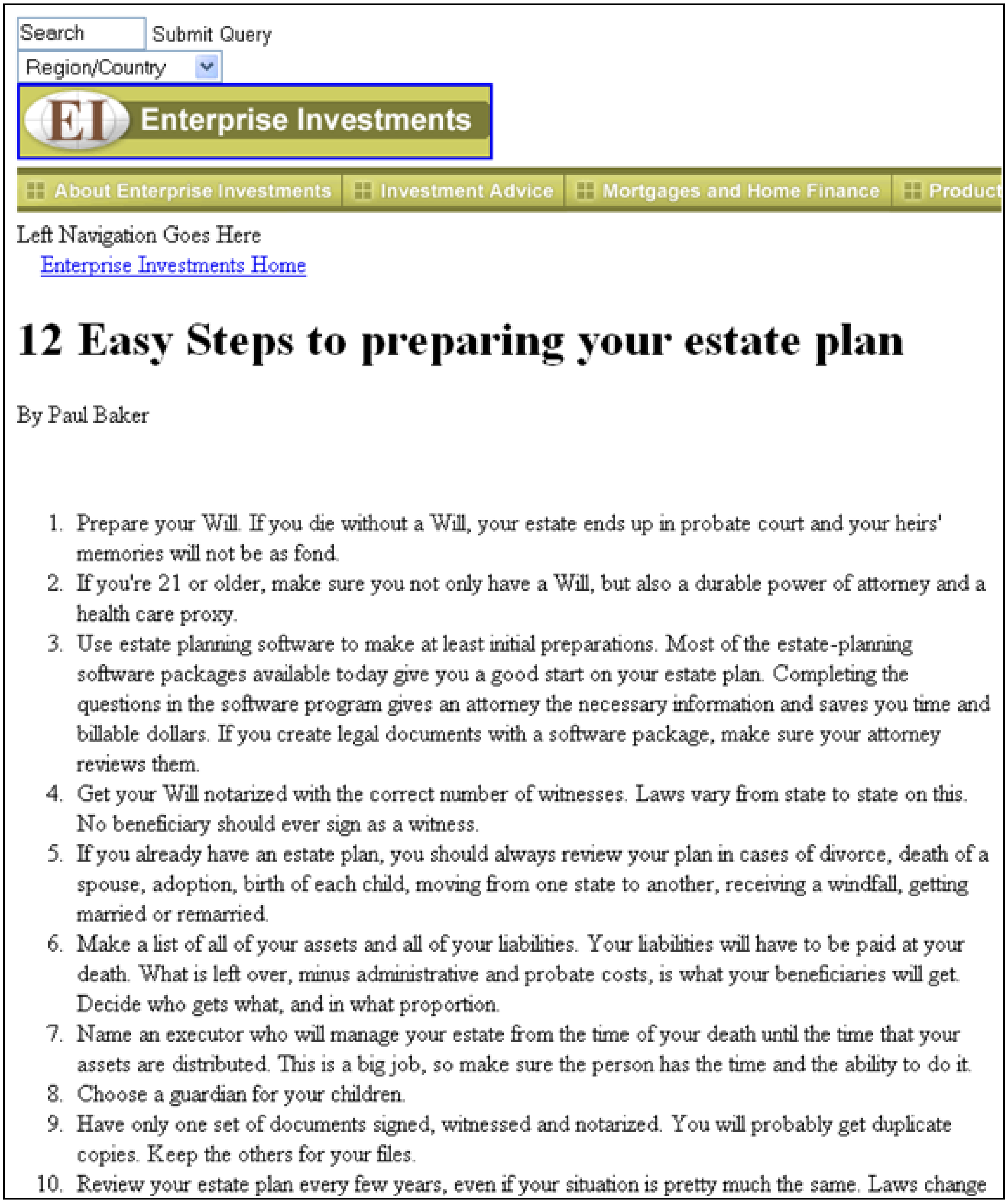
```
#slot("rffNav" "" "" "" "" "template=rffSnEiTop")
```



```
        <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 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 130: 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 131: 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("rffNav" "" "" "" "" "template=rffSnEINavLeft")
#slot("rffNav" "" "" "" "" "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

To demonstrate the process of creating and using Context Variables, we will illustrate how the standard $rxs_navbase Context Variable was created for the Preview Output Context for the Enterprise Investments Site, and how to define additional values for this Context Variable. Other Context Variables can be created easily by copying existing Context Variables.

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

To create the $rxs_navbase Context Variable:

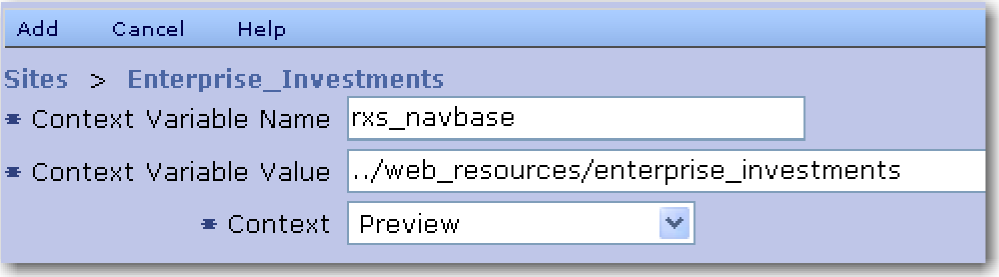
1. Log in to Content Explorer and open the Enterprise Investments Site registration.
2. In the Menu bar, choose *Action > Add Context Variable*.

   Rhythmxy returns the Add Context Variable page
3. In the Context Variable Name field, enter *rxs_navbase*.

4 In the Context Variable Value field, enter *../web_resources/enterprise_investments*. This path points to the location where static files for the Enterprise Investments site are located for previewing within Rhythmyx.

5 In the Context drop list, choose *Preview*.

6 The Context Variable definition should resemble the following screenshot:



Sites > Enterprise_Investments

* Context Variable Name: rxs_navbase

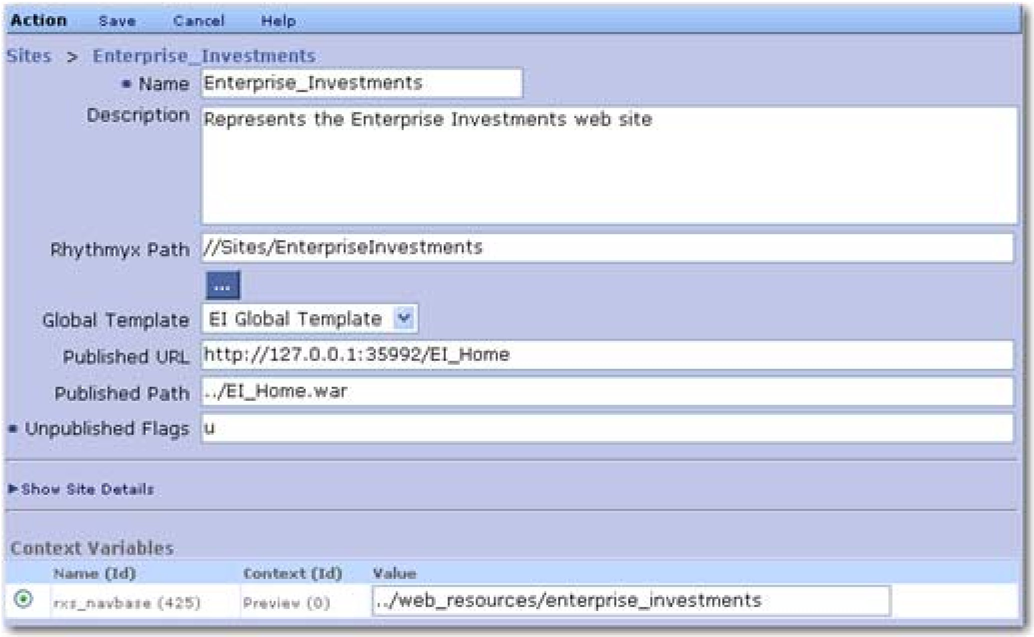
* Context Variable Value: ../web_resources/enterprise_investments

* Context: Preview

Figure 132: Defining the rxs_navbase Context Variable

7 In the Menu bar, click *Add*.

Rhythmyx adds the Variable to the Enterprise Investments Site.



Sites > Enterprise_Investments

* Name: Enterprise_Investments

Description: Represents the Enterprise Investments web site

Rhythmyx Path: //Sites/EnterpriseInvestments

Global Template: EI Global Template

Published URL: http://127.0.0.1:35992/EI_Home

Published Path: ../EI_Home.war

* Unpublished Flags: u

Context Variables

	Name (Id)	Context (Id)	Value
◉	rxs_navbase (425)	Preview (0)	../web_resources/enterprise_investments

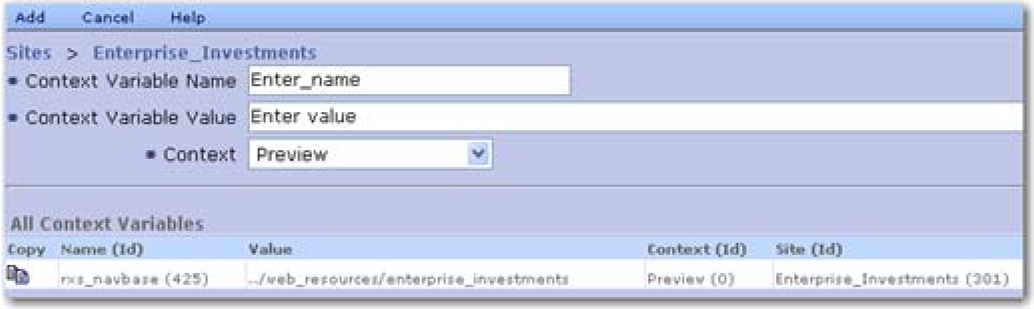
Figure 133: rxs_navbase Context Variable defined

We also need a value for this Context Variable for the Site Folder Assembly Context. We can create this additional value by copying the Context Variable and defining a new value for the Site Folder Assembly Context. To create the Site Folder Assembly value for the rxs_navbase Context Variable:

1 Open the Enterprise Investments Site registration.

2 In the Menu bar, choose *Action > Add Context Variable*.

Rhythmxy returns the Add Context Variable page. Note that the All Context Variables table includes the original value that we defined.



Sites > Enterprise_Investments

- Context Variable Name: Enter_name
- Context Variable Value: Enter value
- Context: Preview

All Context Variables

Copy	Name (Id)	Value	Context (Id)	Site (Id)
	rxs_navbase (425)	../web_resources/enterprise_investments	Preview (0)	Enterprise_Investments (301)

Figure 134: Add Context Variable page showing the rxs_navbase Context Variable with the valeu defined for the Preview Context

3 Click the copy button in the row of the rxs_navbase Context Variable.

Rhythmyx copies the values of the rxs_navbase Context variable to the fields.

4 Change the Context Variable Value to */EI_Home/resources*.

5 In the Context drop list, choose *Site_Folder_Assembly*.

6 In the Menu bar, click *Add*.

Rhythmyx saves the new value for the rxs_navbase Context Variable and adds it to the Enterprise Investments Site registration.

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.rxs_navbase\css\rxs_styles.css
```

Thus, the header references become:

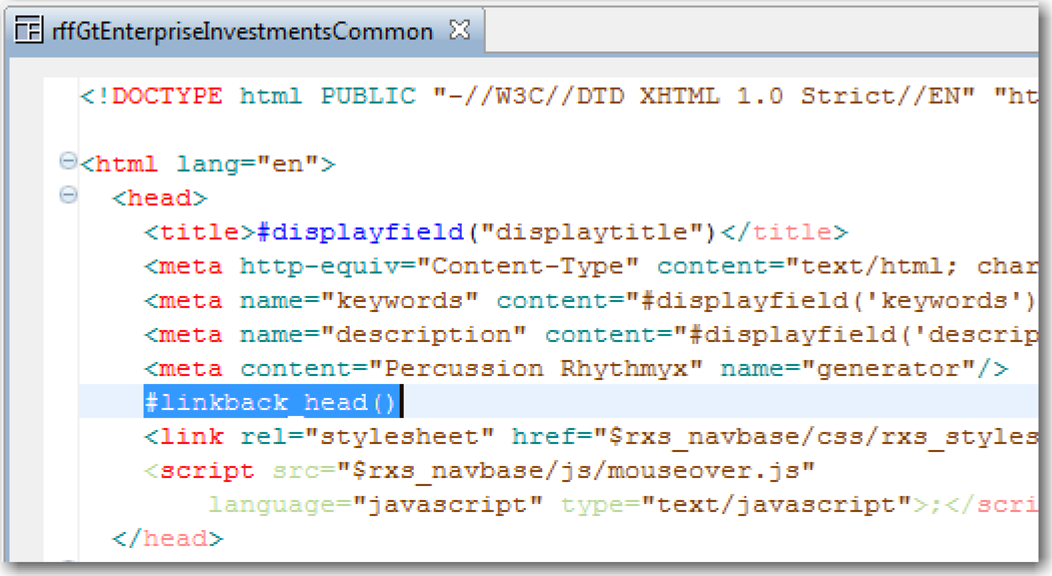
```
<link rel="stylesheet"
href="$sys.variables.rxs_navbase\css\rxs_styles.css" type="text/css" />
<script psx-src="$sys.variables.ResourcePath\js\mouseover.js"
language="javascript" type="text/javascript">;</script>
```

This produces a Preview that uses all of the correct Cascading Stylesheets and JavaScript files.

Adding Linkback

Linkback is a Percussion CM System feature that allows a user that is viewing an HTML page to access the Percussion CM System Content Item from which that page was generated.

To add linkback to a Global Template, add the macro `#linkback_head` to the header in the Global Template HTML code. The following screenshot illustrates the macro added to the rffEnterpriseInvestmentsCommon Global Template.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "ht

<html lang="en">
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta http-equiv="Content-Type" content="text/html; char
    <meta name="keywords" content="#displayfield('keywords')
    <meta name="description" content="#displayfield('descrip
    <meta content="Percussion Rhythmyx" name="generator"/>
    #linkback_head()
    <link rel="stylesheet" href="$rxs_navbase/css/rxs_styles
    <script src="$rxs_navbase/js/mouseover.js"
        language="javascript" type="text/javascript">;</scri
  </head>
```

Figure 135: rffGtEnterpriseInvestmentsCommon Global Template with linkback macro highlighted

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.

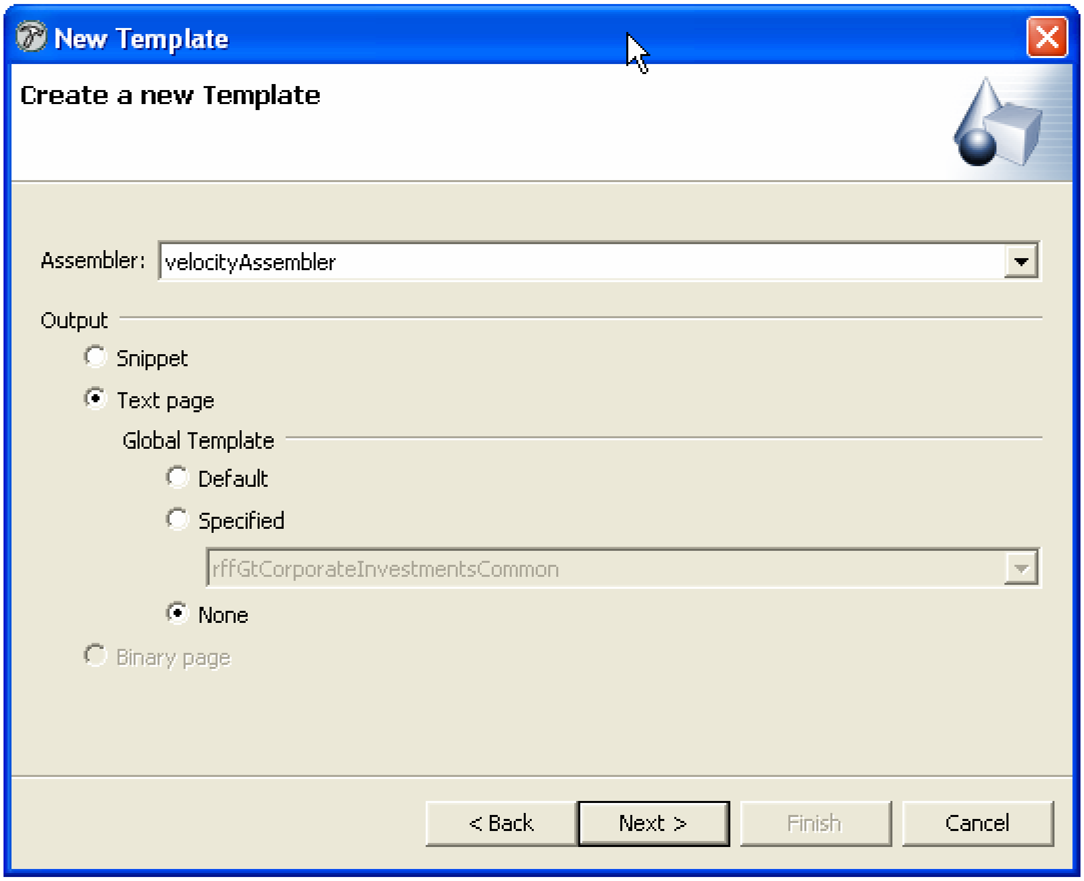
To create the rffPgEIHome PageTemplate object:

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.



New Template

Create a new Template

Assembler: velocityAssembler

Output

○ Snippet

● Text page

Global Template

○ Default

○ Specified

rffGtCorporateInvestmentsCommon

● None

○ Binary page

Figure 136: 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.

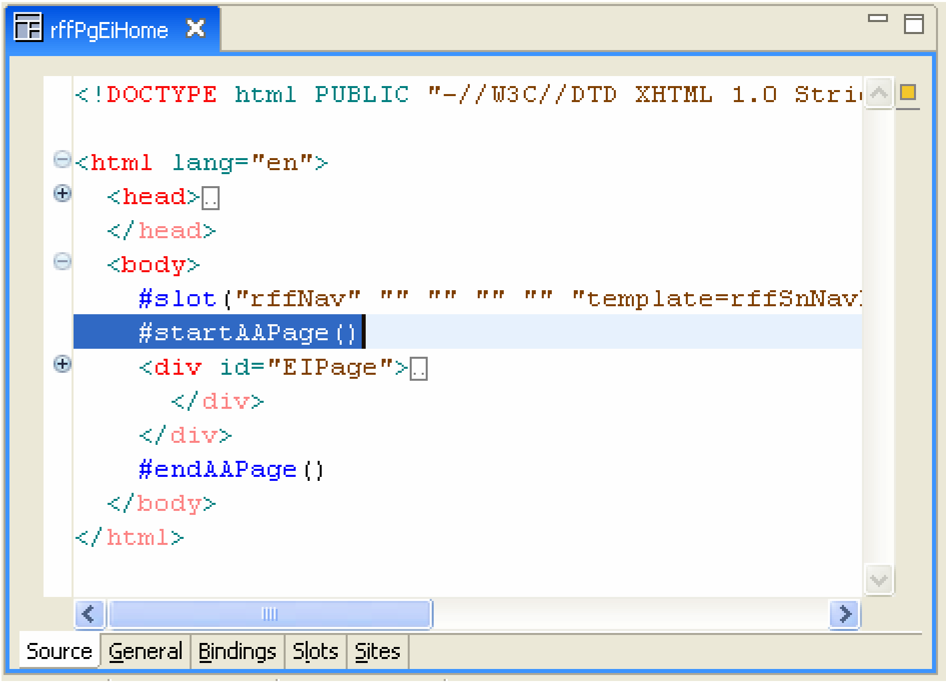
Rhythmyx creates the Template and displays the Template editor for the rffPgEIHome Template.

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>
```

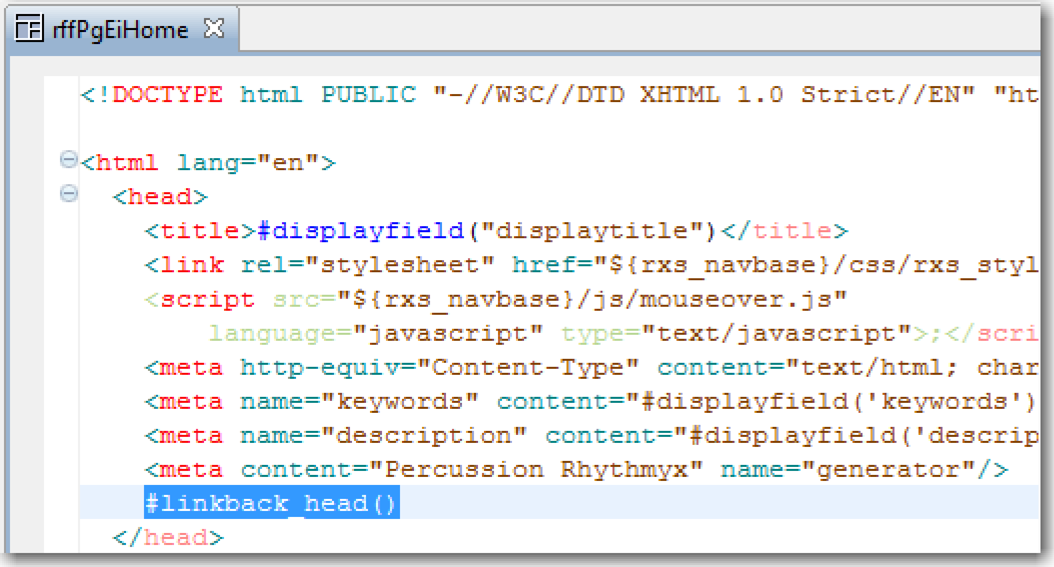
Figure 137: Page Template showing the #startAAPage and #endAAPage macros used when the Template does not have a Global Template

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*.

Adding Linkback to a Page Template

If you want to include linkback functionality in a Page Template that does not use a Global Template, you must add the linkback macro (#linkback_head) to the header section of the Template, as illustrated in the following screenshot of the rffEIHome Template:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "ht

<html lang="en">
  <head>
    <title>#displayfield("displaytitle")</title>
    <link rel="stylesheet" href="${rxs_navbase}/css/rxs_styl
    <script src="${rxs_navbase}/js/mouseover.js"
        language="javascript" type="text/javascript">;</scri
    <meta http-equiv="Content-Type" content="text/html; char
    <meta name="keywords" content="#displayfield('keywords')
    <meta name="description" content="#displayfield('descrip
    <meta content="Percussion Rhythmyx" name="generator"/>
    #linkback_head()
  </head>
```

Figure 138: rffPgEiHome Page Template with linkback macro highlighted

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.

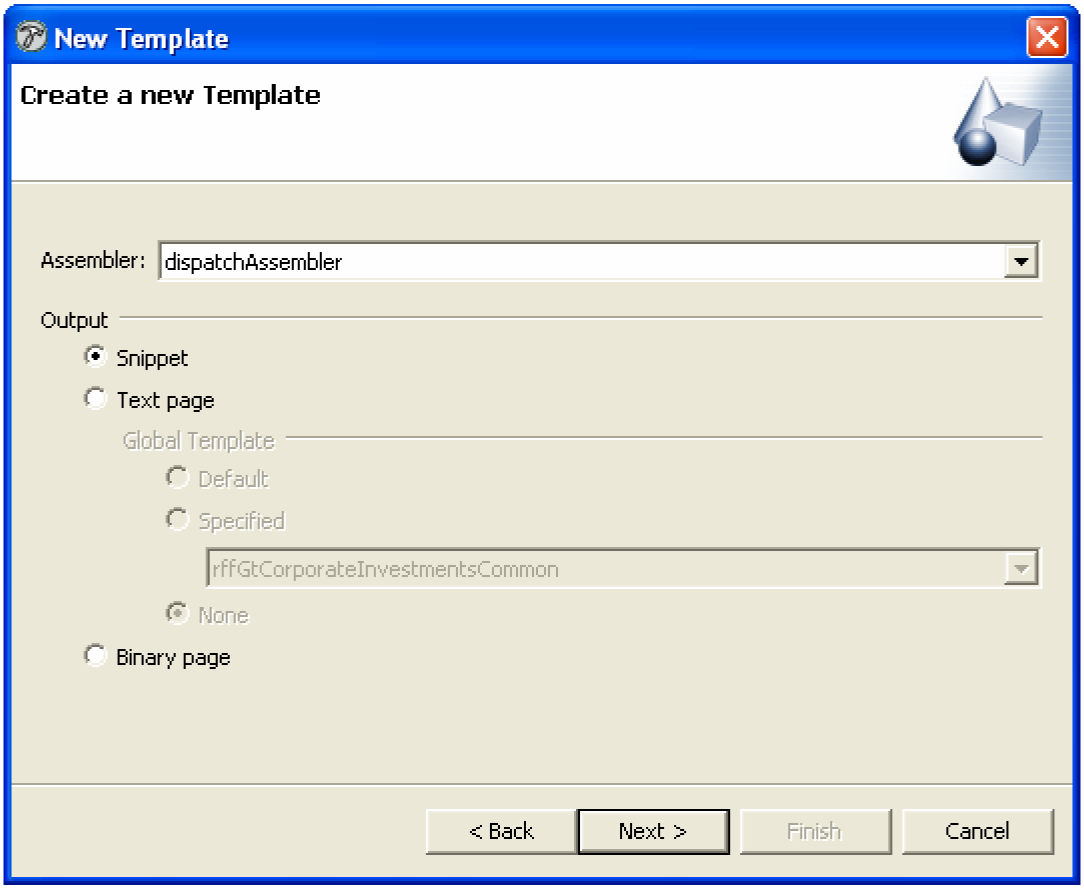
To create the rffDsEIGenericSelector Template object:

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.



New Template

Create a new Template

Assembler: dispatchAssembler

Output

● Snippet

○ Text page

Global Template

○ Default

○ Specified

rffGtCorporateInvestmentsCommon

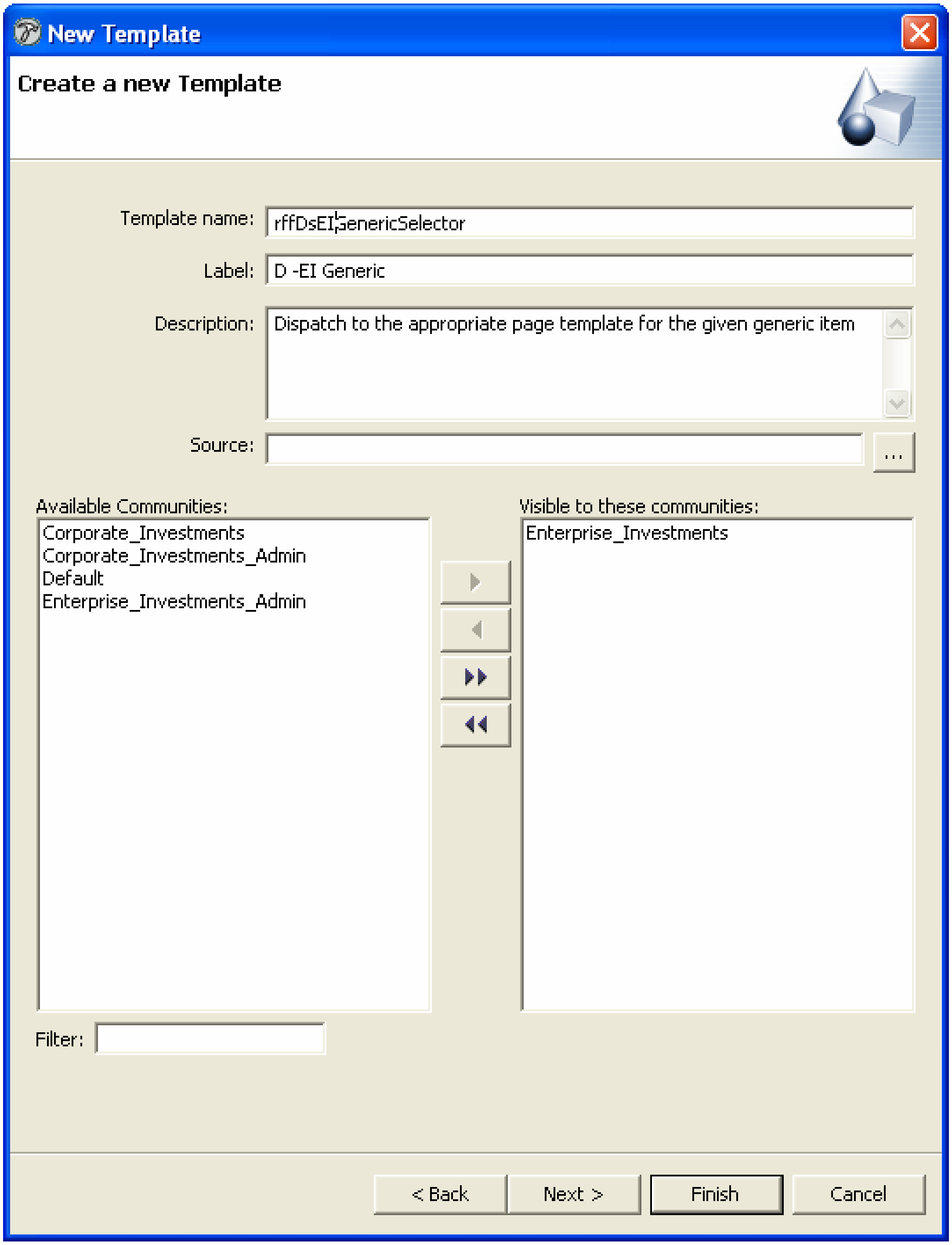
● None

○ Binary page

Figure 139: 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.



Template name: rffDsEIGenericSelector

Label: D -EI Generic

Description: Dispatch to the appropriate page template for the given generic item

Source:

Available Communities:
Corporate_Investments
Corporate_Investments_Admin
Default
Enterprise_Investments_Admin

Visible to these communities:
Enterprise_Investments

Filter:

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

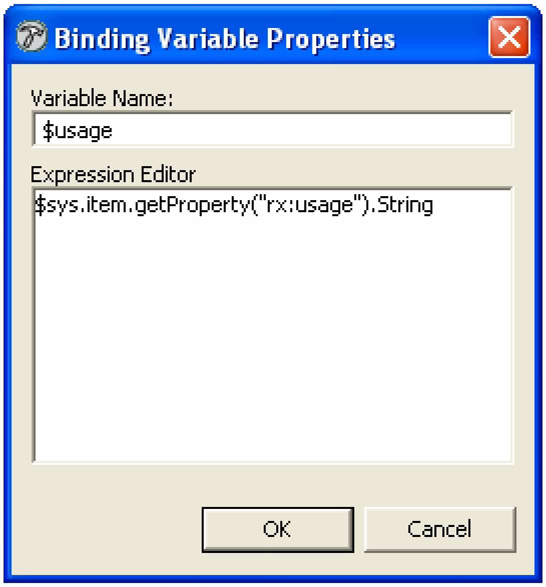
Since we want to select a Template, we will bind the variable $sys.template.

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 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 value of the usage field and assigns it to a variable; we will use $usage:



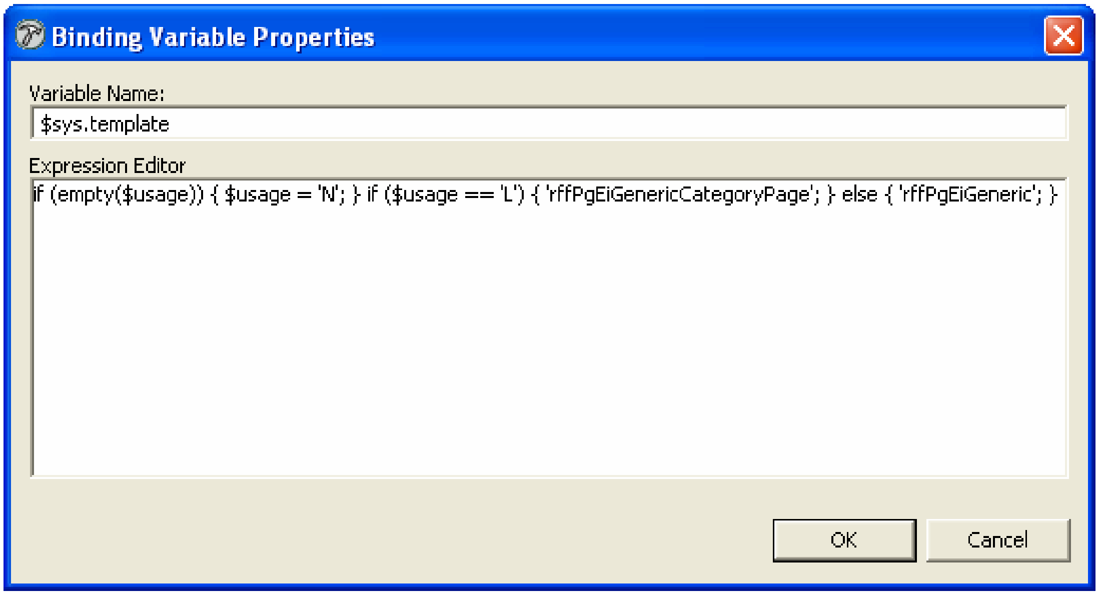
Binding Variable Properties

Variable Name:
$usage

Expression Editor
$sys.item.getProperty("rx:usage").String

Figure 140: $usage binding for Dispatch Template

The second binding tests the value of $usage to determine which Template to select.



Variable Name:

$sys.template

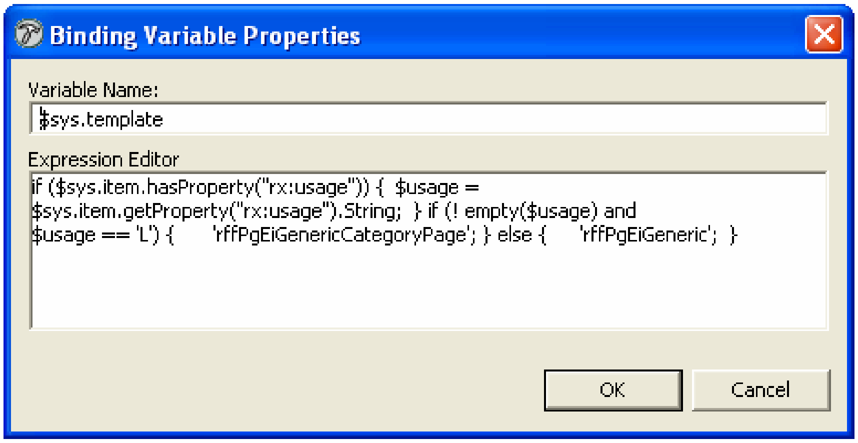
Expression Editor

```
if (empty($usage)) { $usage = 'N'; } if ($usage == 'L') { 'rffPgEiGenericCategoryPage'; } else { 'rffPgEiGeneric'; }
```

Figure 141: 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:



Variable Name:

$sys.template

Expression Editor

```
if ($sys.item.hasProperty("rx:usage")) {  $usage =
$sys.item.getProperty("rx:usage").String;  } if (! empty($usage) and
$usage == 'L') {      'rffPgEiGenericCategoryPage'; } else {     'rffPgEiGeneric';  }
```

Figure 142: 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.

Automated Slots differ from Standard Slots in two ways:

- 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.
- During incremental publishing, all Content Items assembled using a Template that includes an Automated Slot are republished. This processing ensures that changes to related Content Items in the Slot are current on the published page.

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**
rffSnPressReleases2005	Lists all Press Release Content Items created during 2005	Active Assembly	sys_AutoSlotContentFinder

The Allowed content for the Slot is defined as:

Content Type	**Template**
Press Release	rffSnDateAndTitleLink
Press Release	rffSnTitleLinkBullet

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, which does not include a date function or an IN operator (for additional details, see ***Writing Automated Slot Queries*** on page 196). We can circumvent this problem by specifying that the Content Start Date falls before January 1, 20006 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:

```
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:

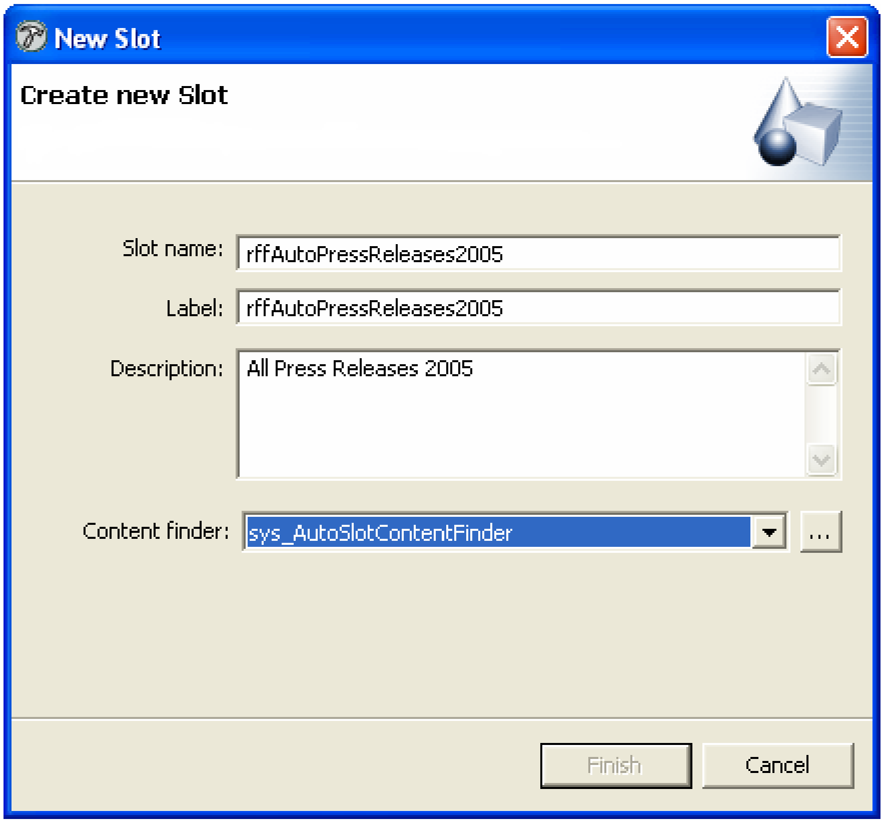
1 In the Rhythmyx Workbench, from the Menu bar, choose *File > New > Slot*.

The Rhythmyx Workbench displays the New Slot wizard.

2 In the **Slot name** field, enter *ffAutoPressReleases2005*. 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*.



Slot name: rffAutoPressReleases2005

Label: rffAutoPressReleases2005

Description: All Press Releases 2005

Content finder: sys_AutoSlotContentFinder

Figure 143: Creating the rffAutoPressReleases2005 Slot

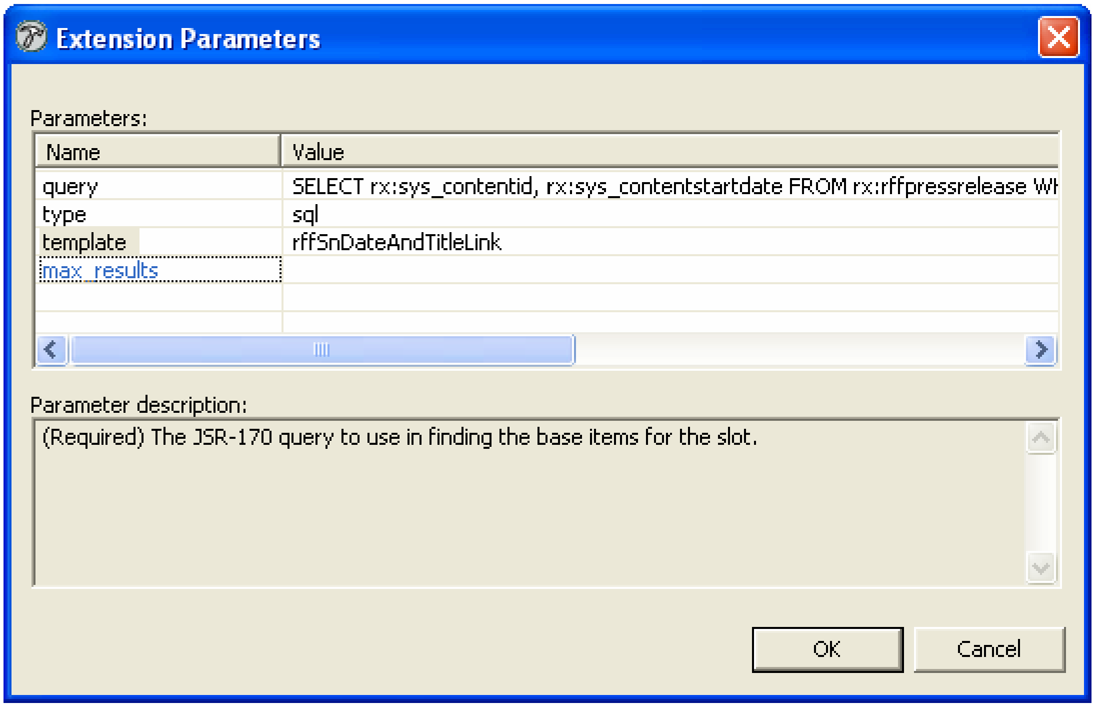
This Content Finder defines the list of Content Items for the Slot automatically. The criteria for selecting the Content Items are defined in the parameters of the sys_AutoSlotContentFinder extension. To specify the criteria for selection Content Items:

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
max_results	(Leave null)

c) When you finish entering values for the parameters, click the [**OK**] button to save your edits.



Parameters:

Name	Value
query	SELECT rx:sys_contentid, rx:sys_contentstartdate FROM rx:rffpressrelease W
type	sql
template	rffSnDateAndTitleLink
max_results	

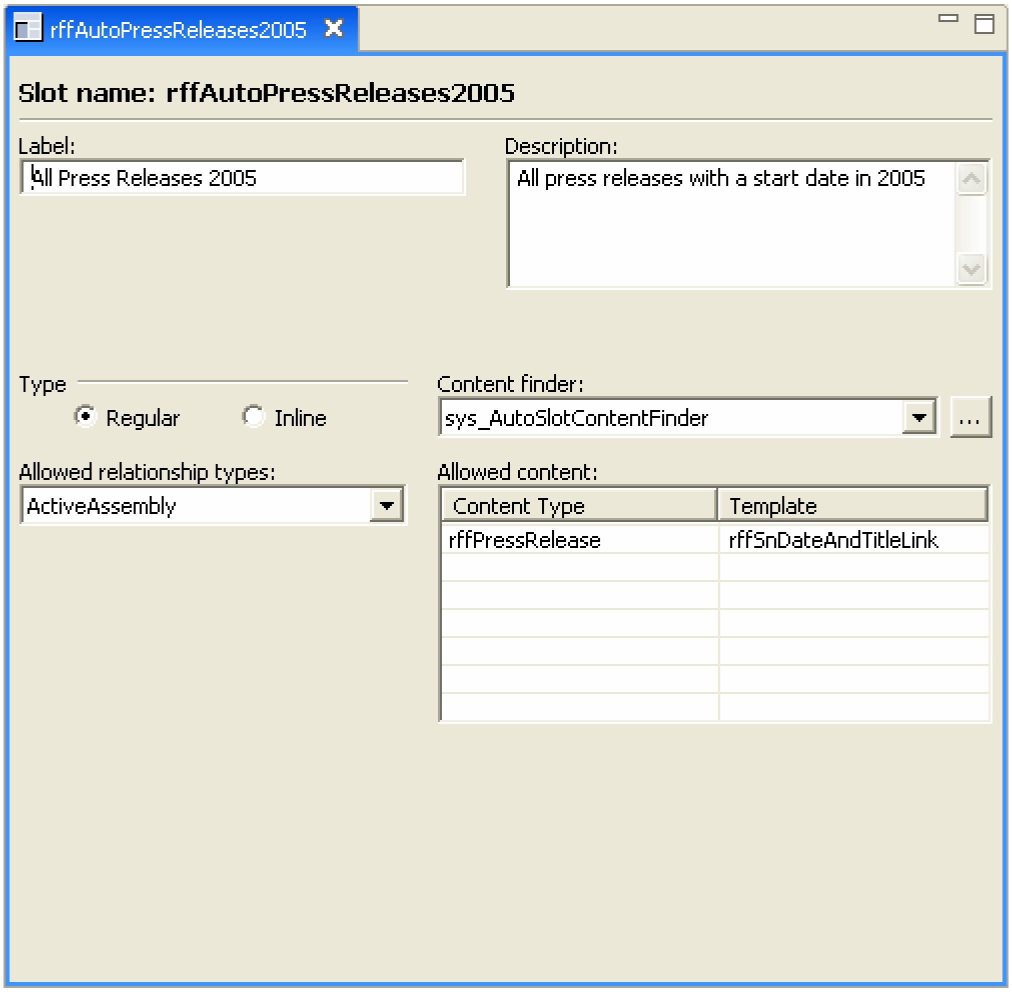
Parameter description:

(Required) The JSR-170 query to use in finding the base items for the slot.

Figure 144: rffAutoPressReleases2005 Query

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

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



rffAutoPressReleases2005

Slot name: rffAutoPressReleases2005

Label: All Press Releases 2005

Description: All press releases with a start date in 2005

Type ● Regular ○ Inline

Content finder: sys_AutoSlotContentFinder

Allowed relationship types: ActiveAssembly

Allowed content:

Content Type	Template
rffPressRelease	rffSnDateAndTitleLink

Figure 145: 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 is 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_folderid 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_folderid`. 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 or 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 `max_result` 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 `max_results = 5`.

You can use the ***Query Debugger*** (see below) to test your query once you write it.

Testing JSR-170 Queries

Rhythmyx provides a Query Debugger page where you can test JSR-170 queries. To access the query debugger, start a browser and enter the following URL:

```
http://<RhythmyxServerName:RhythmyxPort/Rhythmyx/test/search.jsp
```

Where

- RhythmyxServerName is the name or IP address of the machine where you installed Rhythmyx; and
- RhythmyxPort is the port the Rhythmyx server listens on.

You can also access the query debegger by logging in to the Rhythmyx Application Server Home Page (start a browser and enter `http://<RhythmyxServerName:RhythmyxPort` in the Address field), clicking on the Testing and Debugging tools for implementers link to go to the Debugging and Testing Page, then clicking on the Test JSR-170 searches link.

The query box displays a default query. Modify it to match the query you want to use and click the [**execute**] button to execute the query. If your query includes any variable parameters derived from JEXL functions, use the Parameters table to specify sample values for each. Because sample values and values calculated when processing Templates may not match exactly, the results of the query debugger may differ slightly from the results generated when previewing a Template.

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=$sys.item.getProperty(“fund_type”).string`
- `$fundselector.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 Status variable to *open*:
  `$fundselector.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:

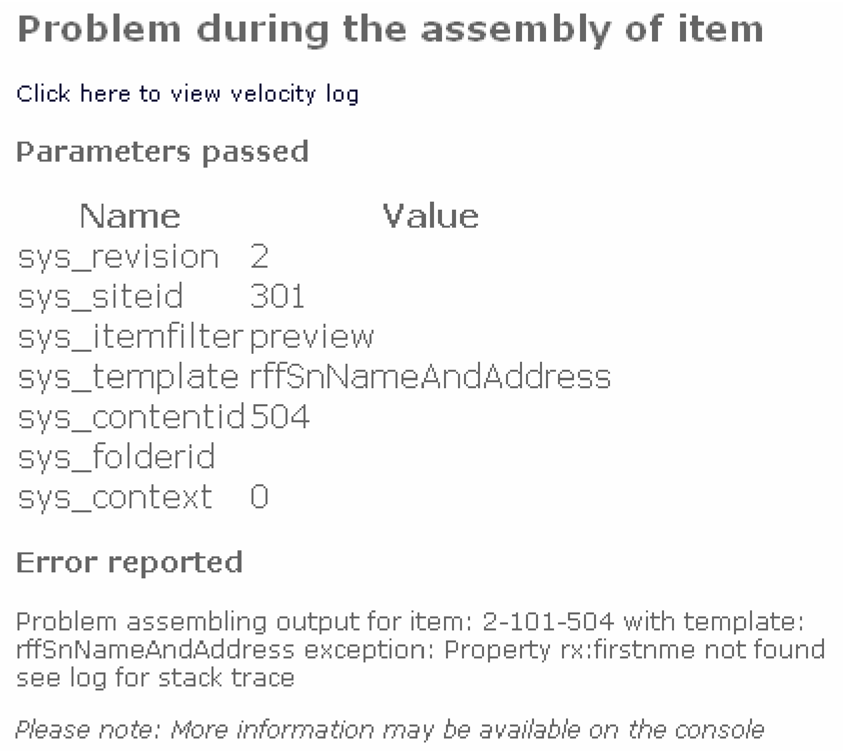
```
select rx:sys_contentid,rx:sys_folderid from rx:funds where
rx:fund_type=:ftype and rx:fund_status=:fstatus
```

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

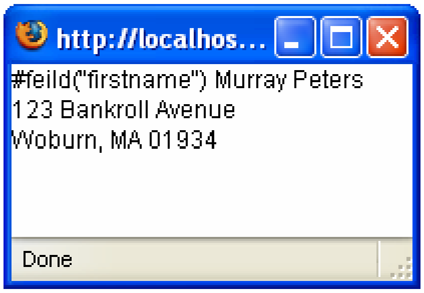
Figure 146: Error page showing "property not found"

This error indicates that the Content Item field was incorrectly 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 to 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:



#feild("firstname") Murray Peters
123 Bankroll Avenue
Woburn, MA 01934

Figure 147: 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.

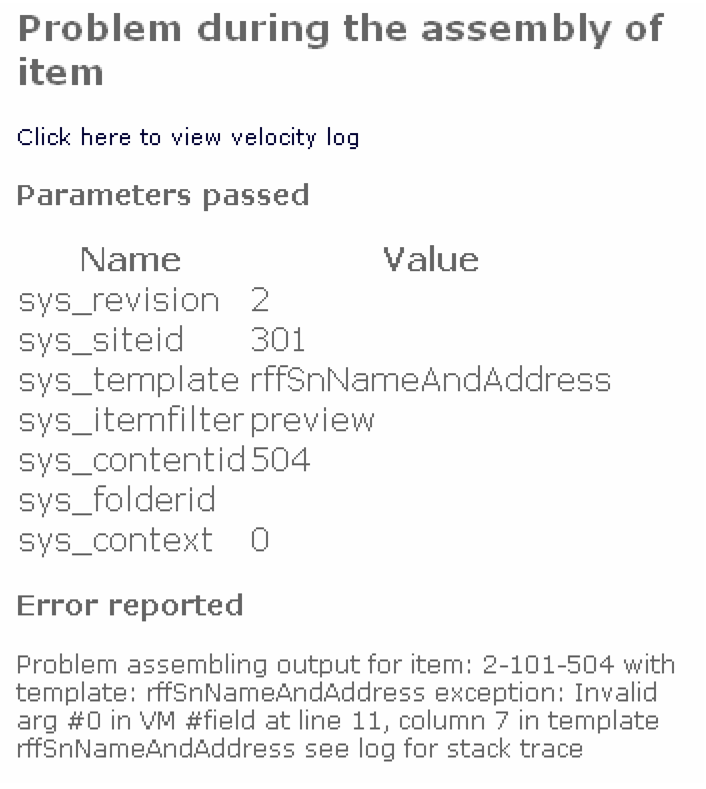
You can usually determine the error in specifying the macro by looking at the output. To address this error, specify the macro correctly:

- 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_resources/vm/rx_assembly.vm`. (NOTE: Custom macros should only be defined in the file `<Rhythmyxroot>/rx_resources/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

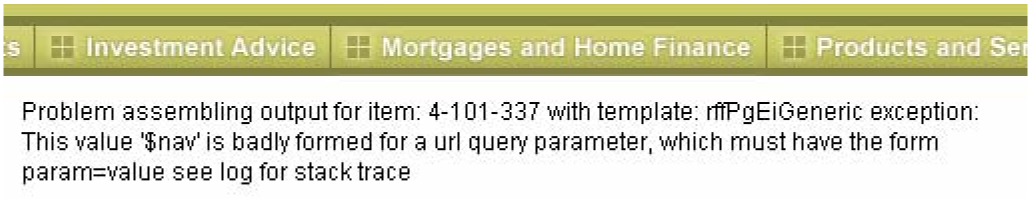
Figure 148: Error page showing "invalid argument" error

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.



Problem assembling output for item: 4-101-337 with template: rffPgEiGeneric exception: This value '$nav' is badly formed for a url query parameter, which must have the form param=value see log for stack trace

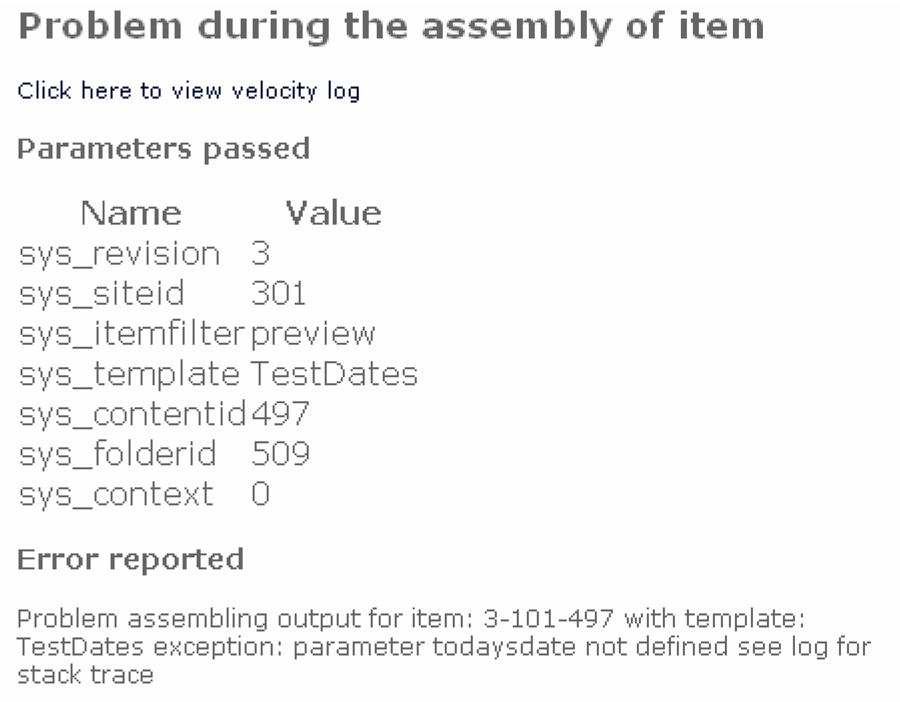
Figure 149: Partially assembled page showing incorrectly formatted object Template macro

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 150: Error page showing "parameter not defined error"

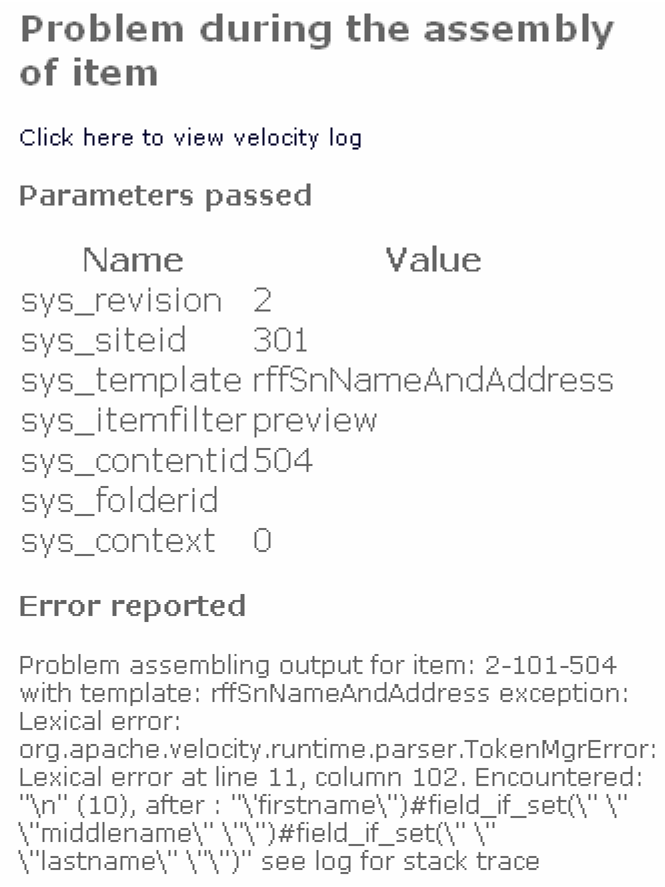
This error typically occurs when you have defined a compound variable (such as $circle.diameter and $circle.radius) and have specified the "root" variable (in this example, $circle) with quotation marks (for example, #slot ("template" "" "" "" "" "$circle"). The leaf variable you were using would be 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

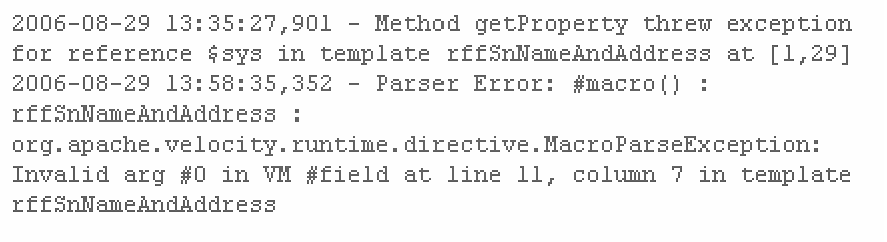
Name	Value
sys_revision	2
sys_siteid	301
sys_template	rffSnNameAndAddress
sys_itemfilter	preview
sys_contentid	504
sys_folderid	
sys_context	0

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(\" \"
\"middlename\" \"\")#field_if_set(\" \"
\"lastname\" \"\")" see log for stack trace
```

Figure 151: Error page showing "lexical error"

The log returns a result similar to the following:



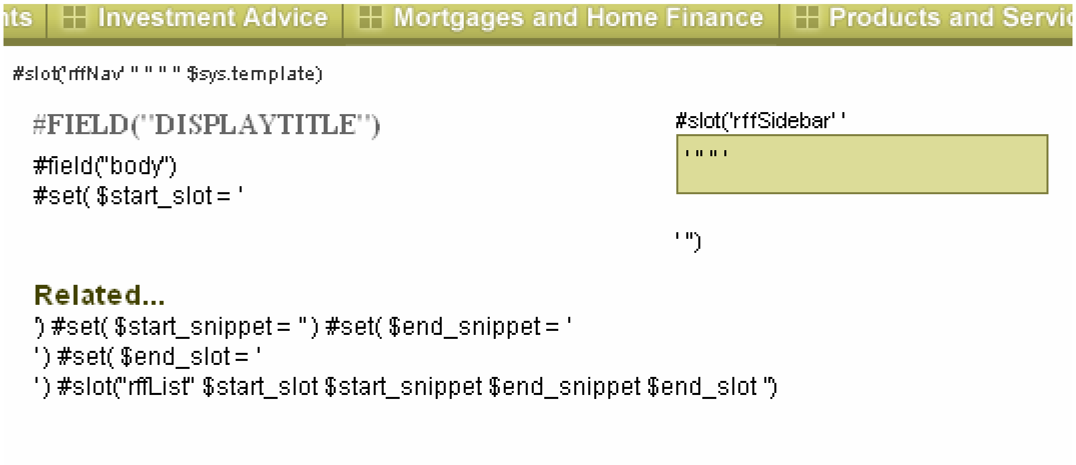
```
2006-08-29 13:35:27,901 - Method getProperty threw exception
for reference $sys in template rffSnNameAndAddress at [1,29]
2006-08-29 13:58:35,352 - Parser Error: #macro() :
rffSnNameAndAddress :
org.apache.velocity.runtime.directive.MacroParseException:
Invalid arg #0 in VM #field at line 11, column 7 in template
rffSnNameAndAddress
```

Figure 152: 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:



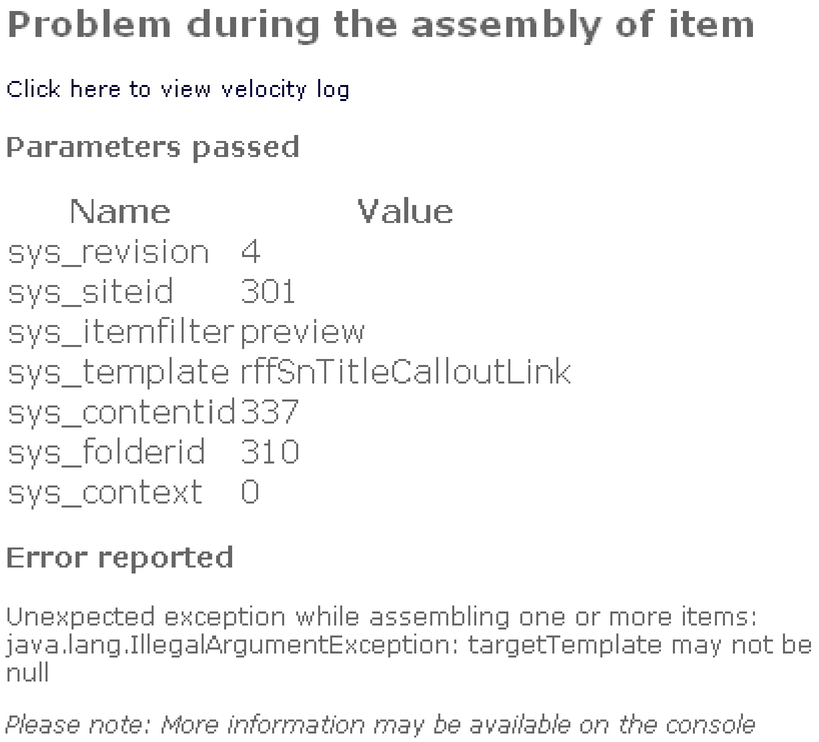
```
#slot('rffNav' " " " " $sys.template)

#FIELD("DISPLAYTITLE")                          #slot('rffSidebar' '
#field("body")                                  ' " " '
#set( $start_slot = '
                                                ' ")

') #set( $start_snippet = '') #set( $end_snippet = '
') #set( $end_slot = '
') #slot("rffList" $start_slot $start_snippet $end_snippet $end_slot ")
```

Figure 153: 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 many 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 154: 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

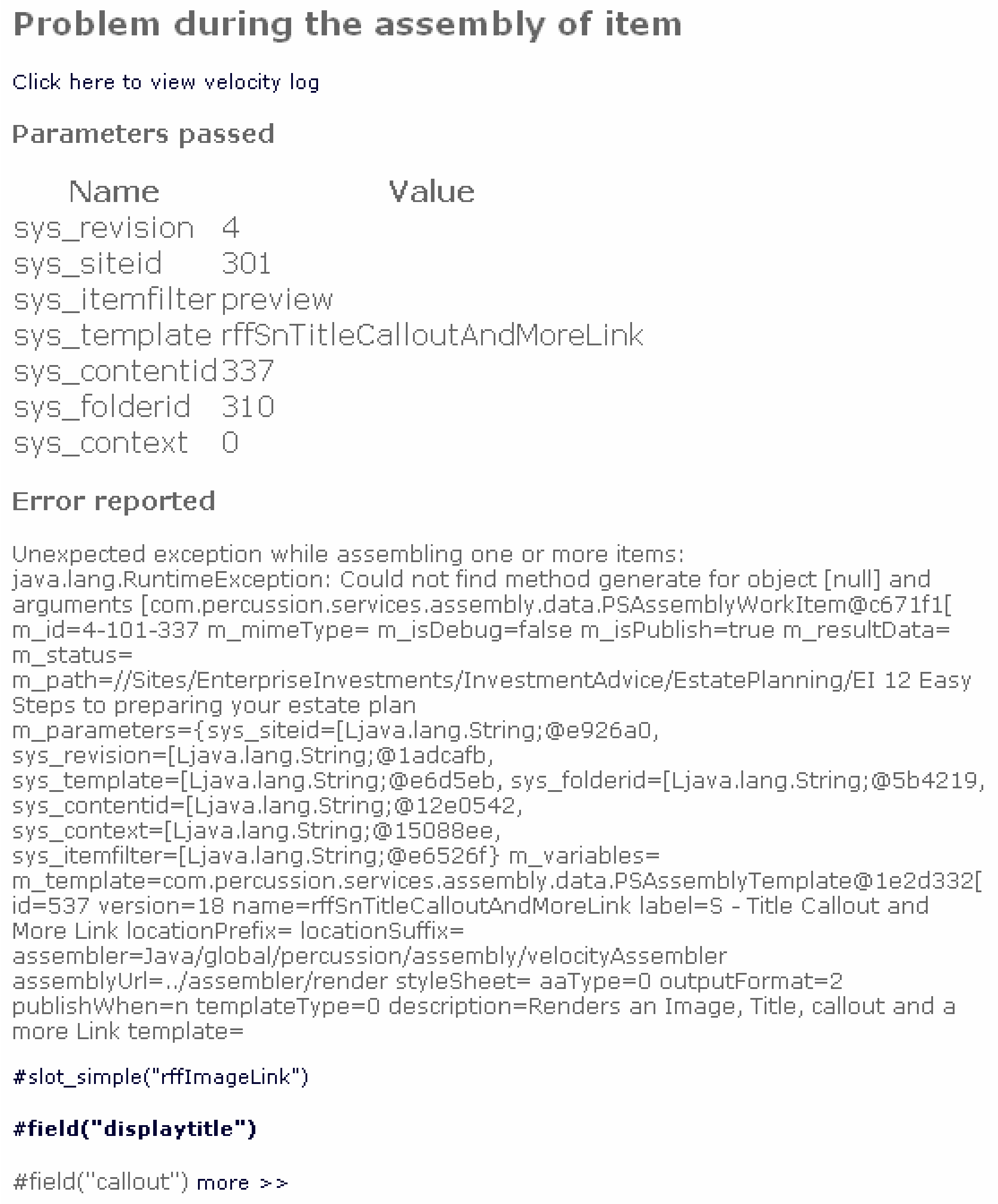
When previewing a binary Content Item, such as an image file or a .pdf file, an error is reported:

- 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

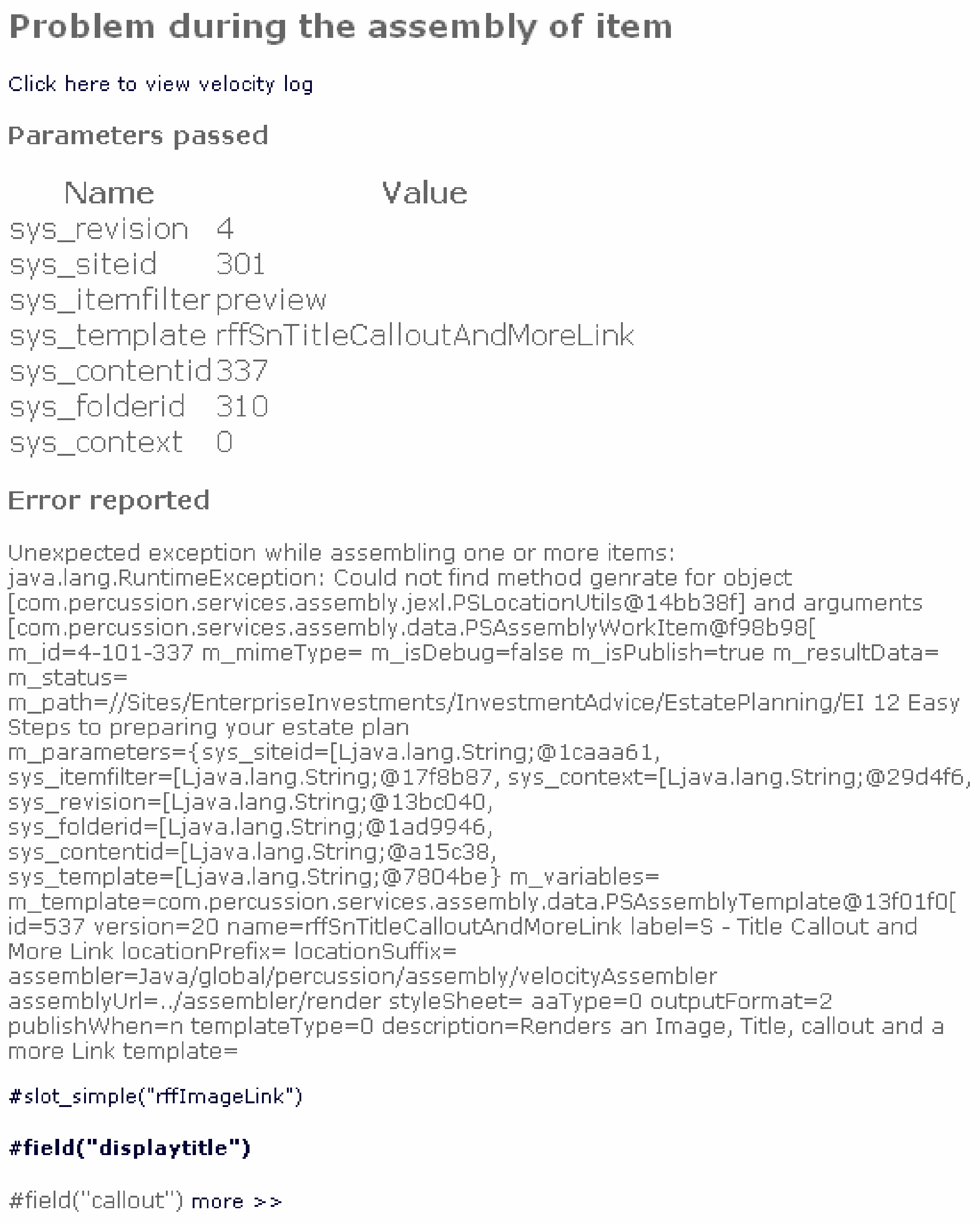
```
Unexpected exception while assembling one or more items:
java.lang.RuntimeException: Could not find method generate for object [null] and
arguments [com.percussion.services.assembly.data.PSAssemblyWorkItem@c671f1[
m_id=4-101-337 m_mimeType= m_isDebug=false m_isPublish=true m_resultData=
m_status=
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 155: Error page showing "could not find 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 genrate 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;@1caaa61,
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 156: Error page showing "Java.lang.RuntimeException: Could not find method <name> for object <bindingfunction>"

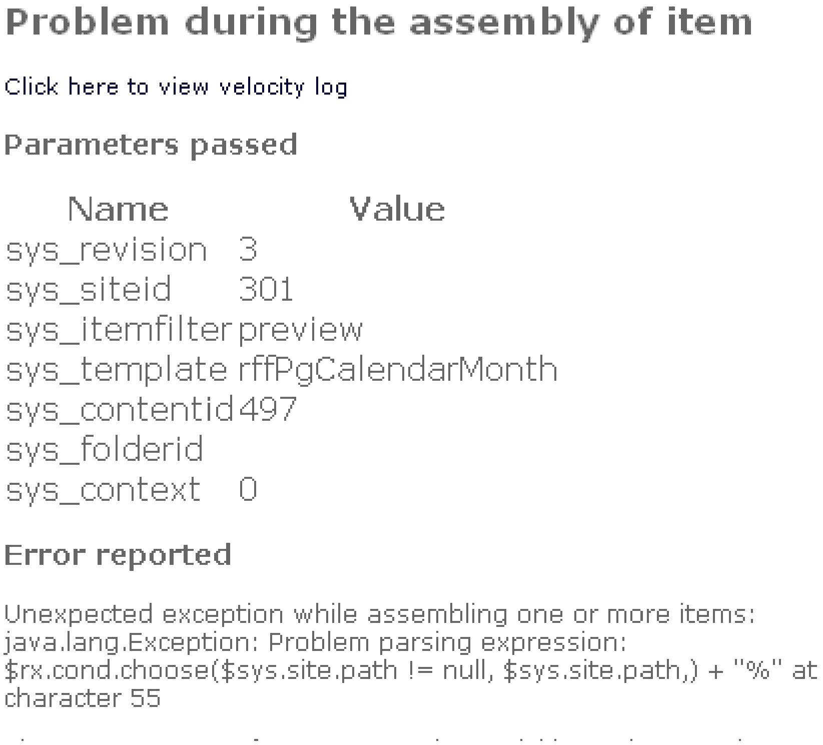
This message may occur for two reasons:

- 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

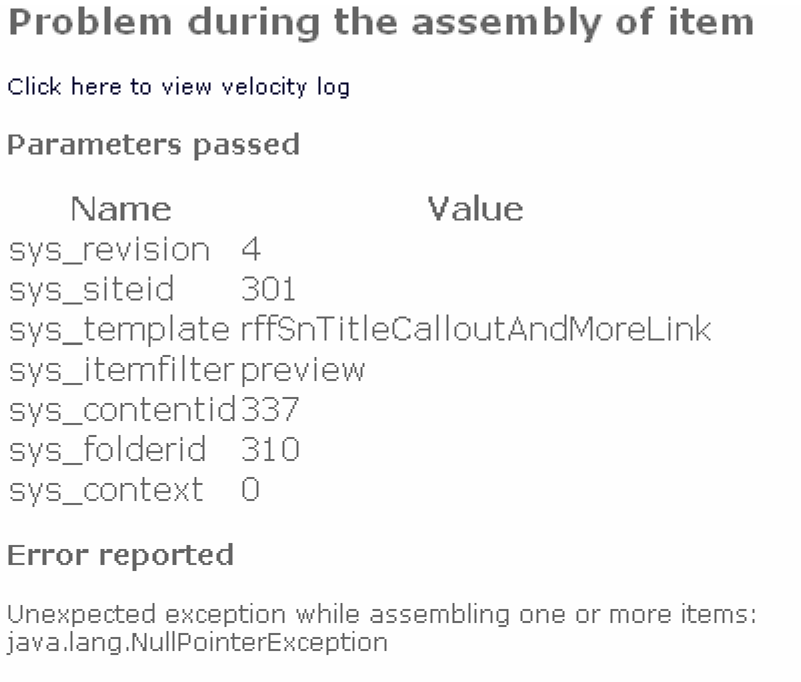
```
Unexpected exception while assembling one or more items:
java.lang.Exception: Problem parsing expression:
$rx.cond.choose($sys.site.path != null, $sys.site.path,) + "%" at
character 55
```

Figure 157: 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 158: Error page showing "null pointer exception"

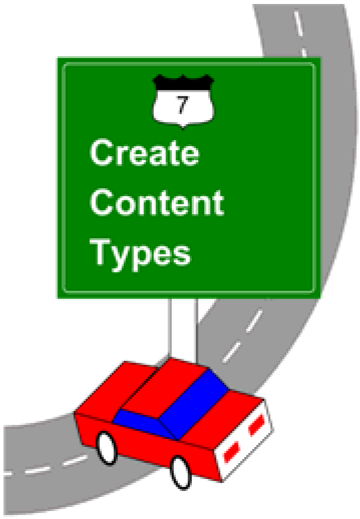
Null pointer exceptions occur whenever a null value is passed to the Assembly engine. Null values could occur for a variety of reasons. Common causes of null pointer exceptions include:

- 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.

CHAPTER 7

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*** (see page 81). 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 217)
- ***Image Content Type*** (see page 219)
- ***Event Content Type*** (see page 220)

Generic Content Type

The Generic Content Type specification 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.

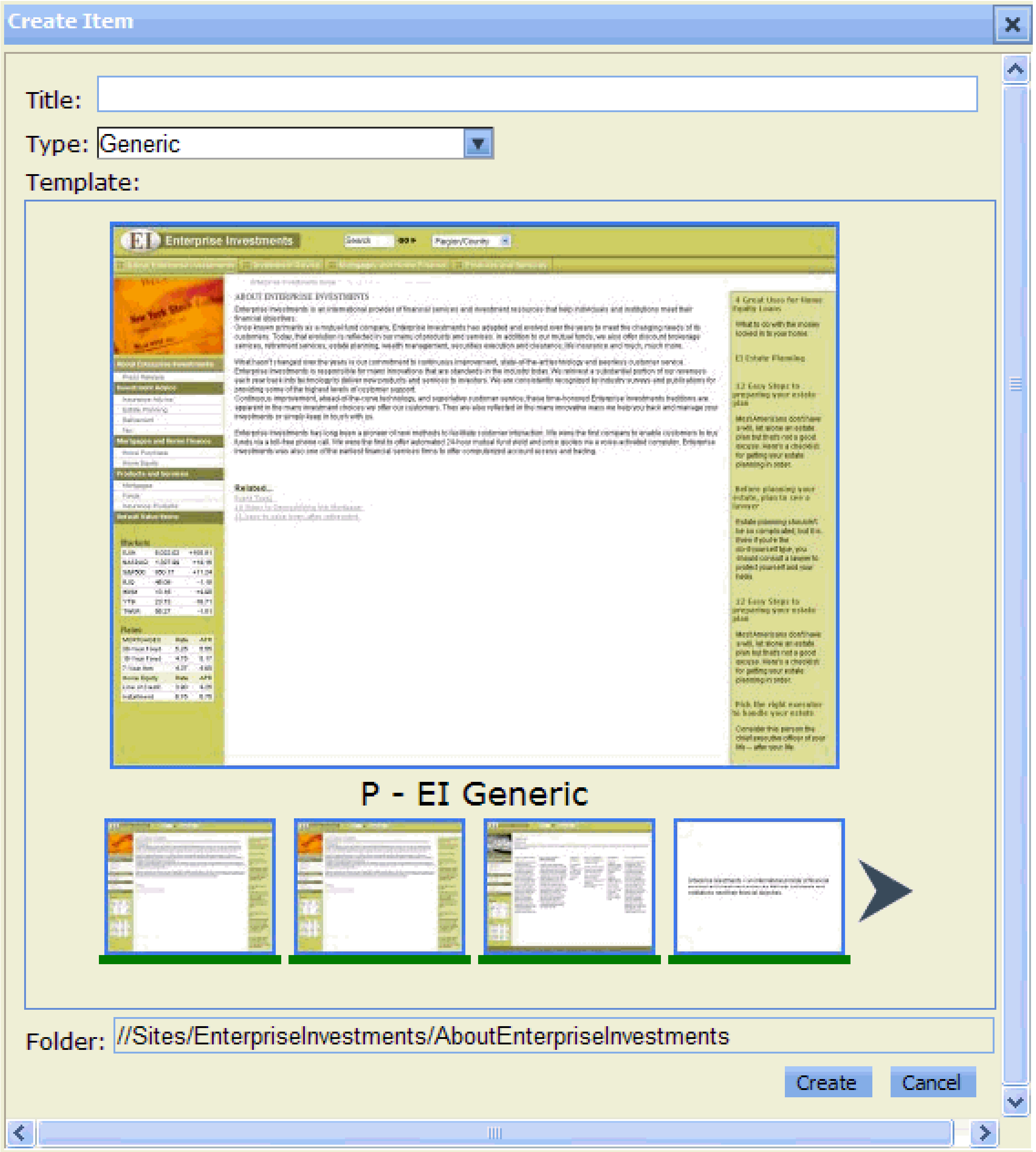
Notice that the system fields sys_title, sys_communityid, sys_lang, sys_currentview, sys_workflowid, and sys_hibernateVersion are listed. By default, they are included in every Content Type because Rhythmyx uses them for internal processing of Content Items.

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.

A local field named Usage, which lets the content contributor specify whether or not the item holds information about a product category, is included. This field enables a ***Dispatch Template*** (see page 187) to determine the correct Page Template to be applied to an item.

Below the table, the Default Values, Allowed Workflows, Default Workflow, and Communities that can view the Content Type are listed.

Notice that most required fields have a default value. This is recommended to simplify the process of creating a new item in the Active Assembly interface. When a user creates a new item in Active Assembly, the following dialog opens:



Create Item

Title:

Type: Generic

Template:

P - EI Generic

Folder: //Sites/EnterpriseInvestments/AboutEnterpriseInvestments

Figure 159: Create Item dialog

The only fields available are the Title and the Content Type. Rhythmyx places the value entered into the Title field into the sys_title field and the displaytitle field (if the displaytitle field exists). With some exceptions in binary Content Type fields, if you mark fields other than sys_title and displaytitle as required, but do not give them a default value, the user is required to open the content editor and fill in the field before creating the item. If you enter default values during implementation, the user can avoid this extra step. (Notice that the sys_workflowid field appears to have no default value, but when the content item is created, Rhythmyx gives it the value of the default Workflow for the Content Type.)

We will continue to refer to the Generic Content Type specification when we create the Generic Content Type in the section ***Basic Content Type Creation*** (on page 222).

Image Content Type

The ***Image Content Type specification*** (see page 467) 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 239). 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 Default Values, Allowed Templates, Allowed Workflow, Default Workflow, and Communities that can view the Content Type are listed.

In the preceding discussion of the ***Generic Content Type*** (see page 217), we explained that most of the required fields have default values in order to simplify the process of creating a new content item in Active Assembly. In the Image Content Type, the fields img1 and img1_ext are required but do not have default values. The system makes an exception for these fields for the ease of creating an Image content item in Active Assembly, but no further Workflow processing of them can be completed until an image is uploaded.

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 250). 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 Default Values, Allowed Templates, 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
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

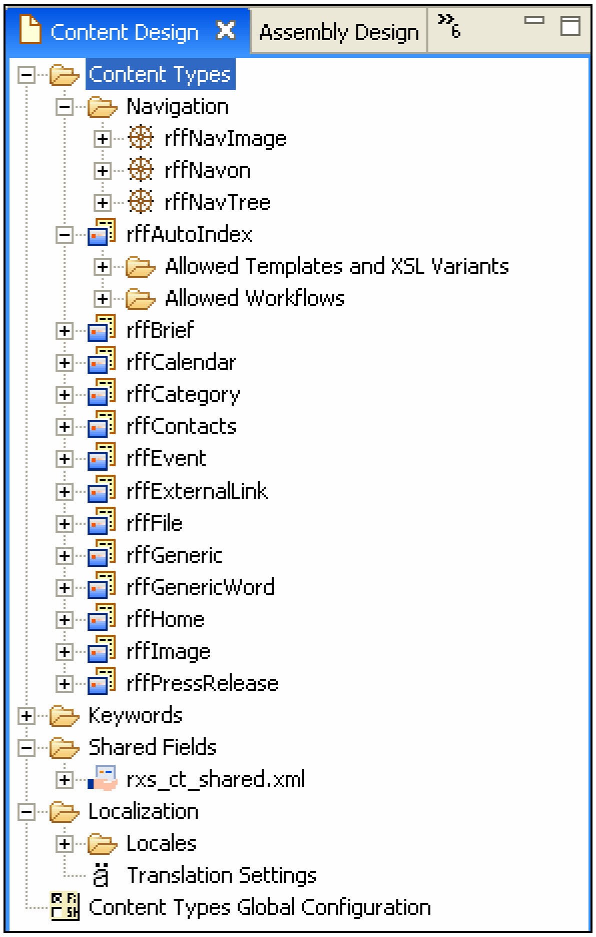
Name	Label	Control Name	Occur	Data Type	Format
event_contact	Event Contact:	sys_TextArea	optional	text	255

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 279) for more information about managed navigation.



- Content Types
  - Navigation
    - rffNavImage
    - rffNavon
    - rffNavTree
  - rffAutoIndex
    - Allowed Templates and XSL Variants
    - Allowed Workflows
  - rffBrief
  - rffCalendar
  - rffCategory
  - rffContacts
  - rffEvent
  - rffExternalLink
  - rffFile
  - rffGeneric
  - rffGenericWord
  - rffHome
  - rffImage
  - rffPressRelease
- Keywords
- Shared Fields
  - rxs_ct_shared.xml
- Localization
  - Locales
  - Translation Settings
- Content Types Global Configuration

Figure 160: 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 local field, Usage) and includes no special features. Its main fields are 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 223).

2 ***Including fields*** (see page 227).

3 ***Specifying an Icon for the Generic Content Type.*** (see page 229)

4 ***Making the Generic Content Type Visible to Another Community*** (on page 231).

5 ***The Generic Content Editor*** (see page 235).

6 ***Viewing Generic Content Items*** (see page 236).

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 modeled 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*.

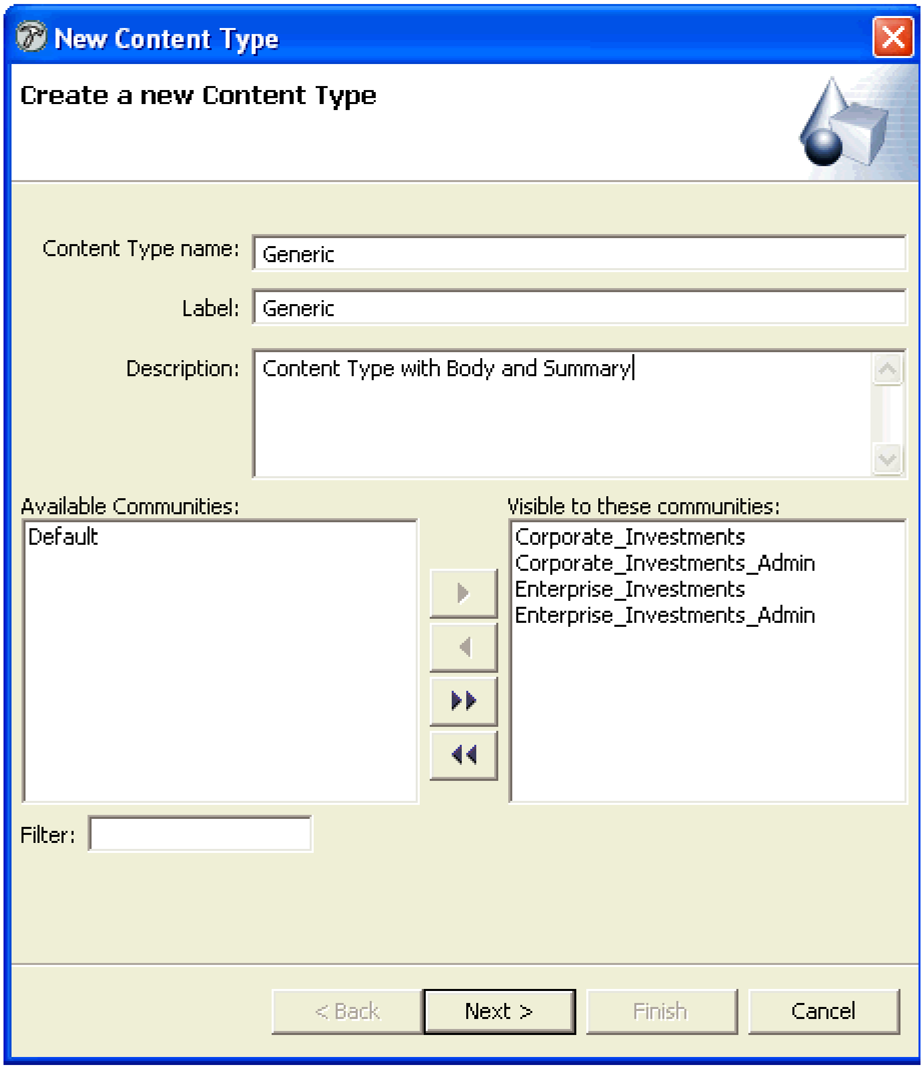
The Rhythmyx Workbench displays the New Content Type wizard.

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.



New Content Type

Create a new Content Type

Content Type name: Generic

Label: Generic

Description: Content Type with Body and Summary

Available Communities:
Default

Visible to these communities:
Corporate_Investments
Corporate_Investments_Admin
Enterprise_Investments
Enterprise_Investments_Admin

Filter:

Figure 161: New Content Type wizard, first screen

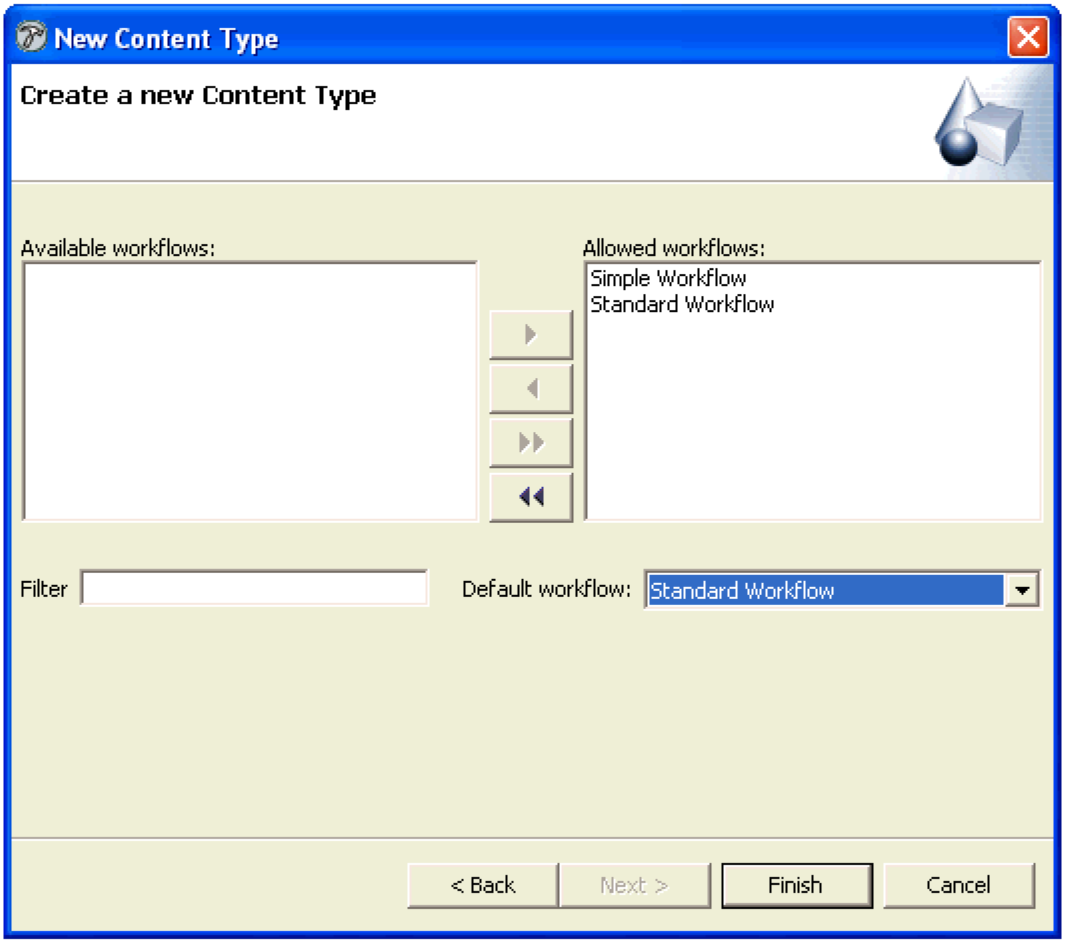
Note: The Default Community is included with Rhythmyx and is not part of the FastForward sample; therefore it is not usually given access to FastForward components.

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*.



Allowed workflows: Simple Workflow, Standard Workflow

Default workflow: Standard Workflow

Figure 162: 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.

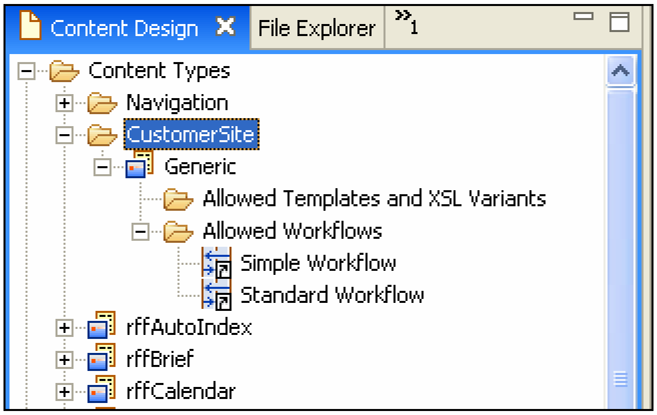
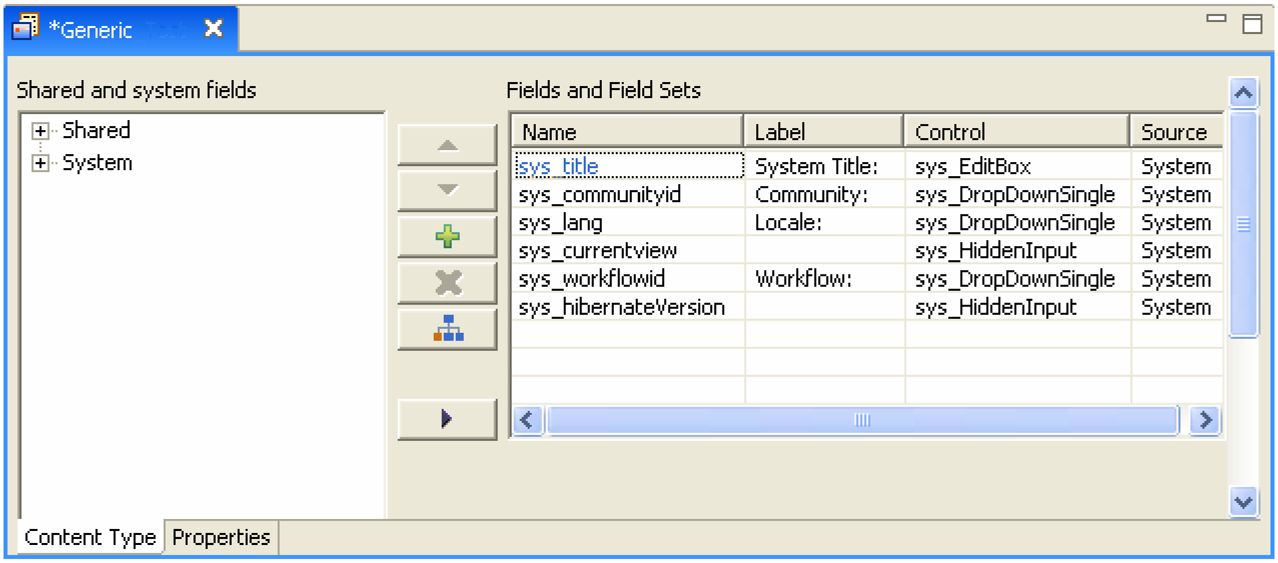


Figure 163: New Content Type in Content Design View

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)



Name	Label	Control	Source
sys_title	System Title:	sys_EditBox	System
sys_communityid	Community:	sys_DropDownSingle	System
sys_lang	Locale:	sys_DropDownSingle	System
sys_currentview		sys_HiddenInput	System
sys_workflowid	Workflow:	sys_DropDownSingle	System
sys_hibernateVersion		sys_HiddenInput	System

Figure 164: Content Type Editor

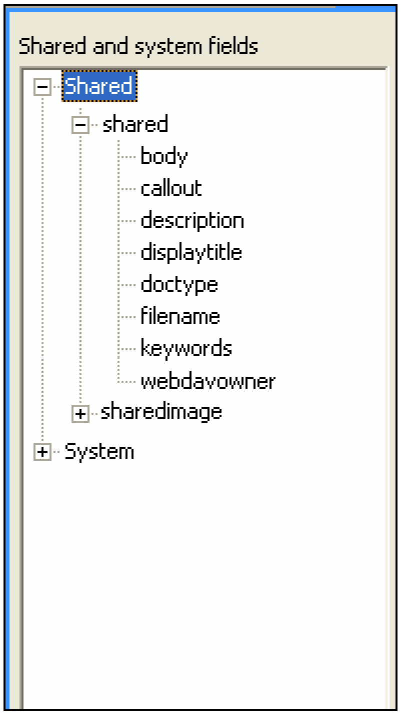
You can move the position of these fields in the table to insert other fields above or below them.

For the Generic Content Type, you simply include additional existing shared and/or system fields. One local field is required; we will also add the local field usage.

To include fields in the Generic Content Type:

1. Refer to the rffGeneric Content Type specification 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*.



Shared and system fields

- Shared
  - shared
    - body
    - callout
    - description
    - displaytitle
    - doctype
    - filename
    - keywords
    - webdavowner
  - sharedimage
- System

Figure 165: 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. 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 usage field, see ***Including a Local Field*** (see page 241).

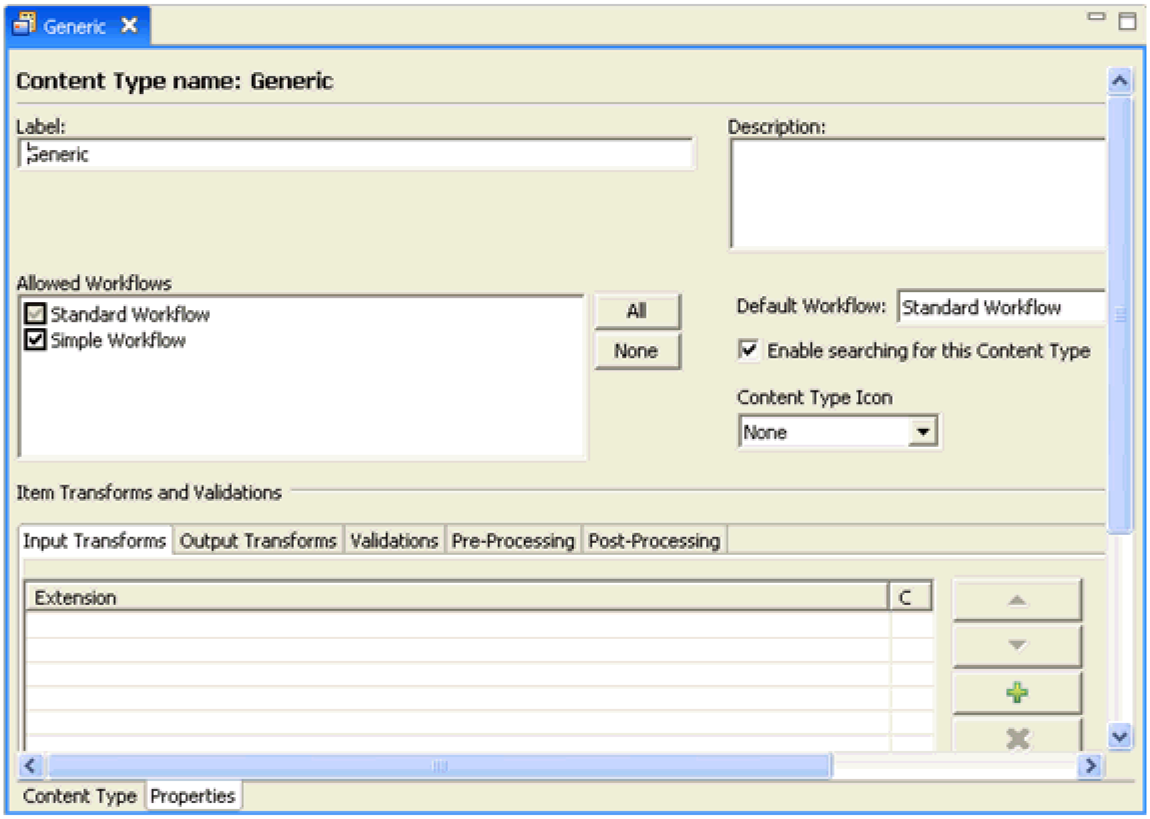
When you are done, the Content Type tab should appear as follows. Note that not all of the fields are visible in the portion of the Fields and Field Sets table shown.

8 To save the changes you have made in the editor, click the Save icon in the Menu bar.

9 Click the Properties tab at the bottom of the Content Type editor to ***specify an icon to represent the Content Type*** (see page 229).

Specifying an Icon for the Generic Content Type

The Property tab should appear as follows:



Content Type name: Generic

Label: Generic

Description:

Allowed Workflows

☑ Standard Workflow

☑ Simple Workflow

Default Workflow: Standard Workflow

☑ Enable searching for this Content Type

Content Type Icon: None

Item Transforms and Validations

Extension	C

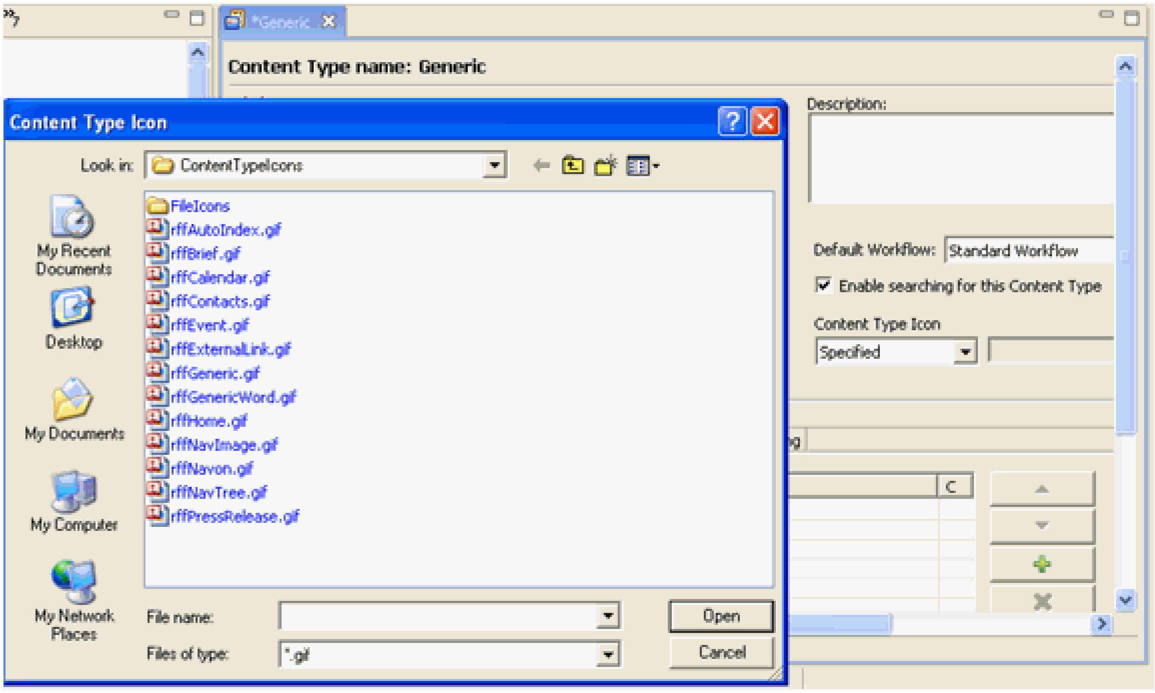
Figure 166: Properties Tab

By default, the Content Type Icon field is set to *None*, indicating that the default 🗋 icon will represent Generic content items in interfaces that use icons. Rhythmyx already includes a custom icon for the Generic content item, rffGeneric.gif, in the folder <Rhythmyx root>\rx_resources\images\ContentTypeIcons. During an actual implementation, once you select an icon file, Rhythmyx moves it to this folder if it is not already located there.

To specify the custom icon for the Content Type:

1 Change the value in the Content Type Icon drop list from *None* to *Specified*. (A third option is *File Extension Field*; we will cover its use in the Image Content Type topic ***Entering Content Editor Properties*** (see page 243)).

The Properties tab opens a search dialog and displays a blank field next to the Content Type Icon field.



Content Type name: Generic

Description:

Default Workflow: Standard Workflow

☑ Enable searching for this Content Type

Content Type Icon: Specified

Content Type Icon

Look in: ContentTypeIcons

- FileIcons
- rffAutoIndex.gif
- rffBrief.gif
- rffCalendar.gif
- rffContacts.gif
- rffEvent.gif
- rffExternalLink.gif
- rffGeneric.gif
- rffGenericWord.gif
- rffHome.gif
- rffNavImage.gif
- rffNavon.gif
- rffNavTree.gif
- rffPressRelease.gif

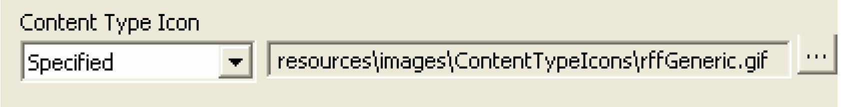
File name:

Files of type: *.gif

Figure 167: Choosing a Content Type Icon

2 If the search dialog does not open to ...\rx_resources\images\ContentTypeIcons, browse to the location. Choose rffGeneric.gif and click [**Open**].

3 The search dialog closes and the blank field now stores the filename rffGeneric.gif.



Content Type Icon

Specified | resources\images\ContentTypeIcons\rffGeneric.gif

Figure 168: Generic Icon selected

The rffGeneric.gif icon will now represent Generic content items in interfaces that use icons.

4 Save your changes.

Making the Generic Content Type Visible to Another Community

In the Workbench, look at Community Visibility view. Your Generic Content Type is listed under Communities that it is visible to.

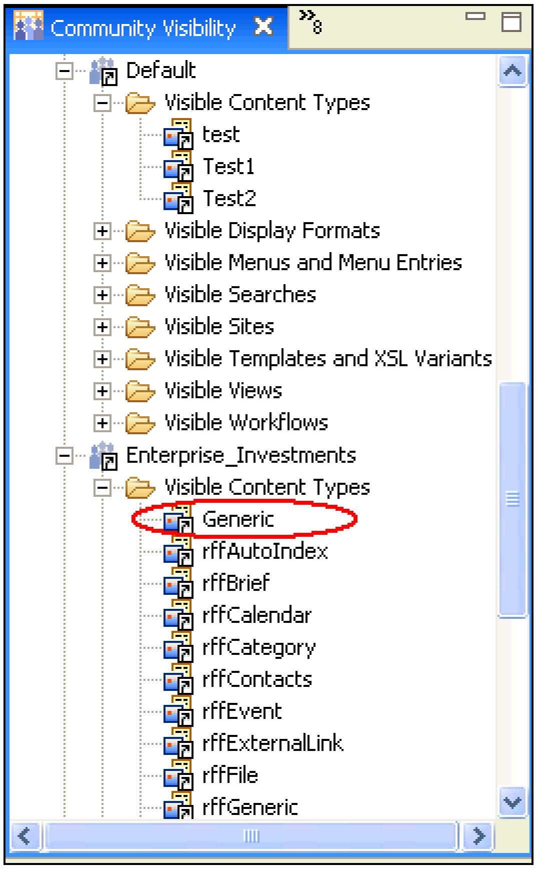


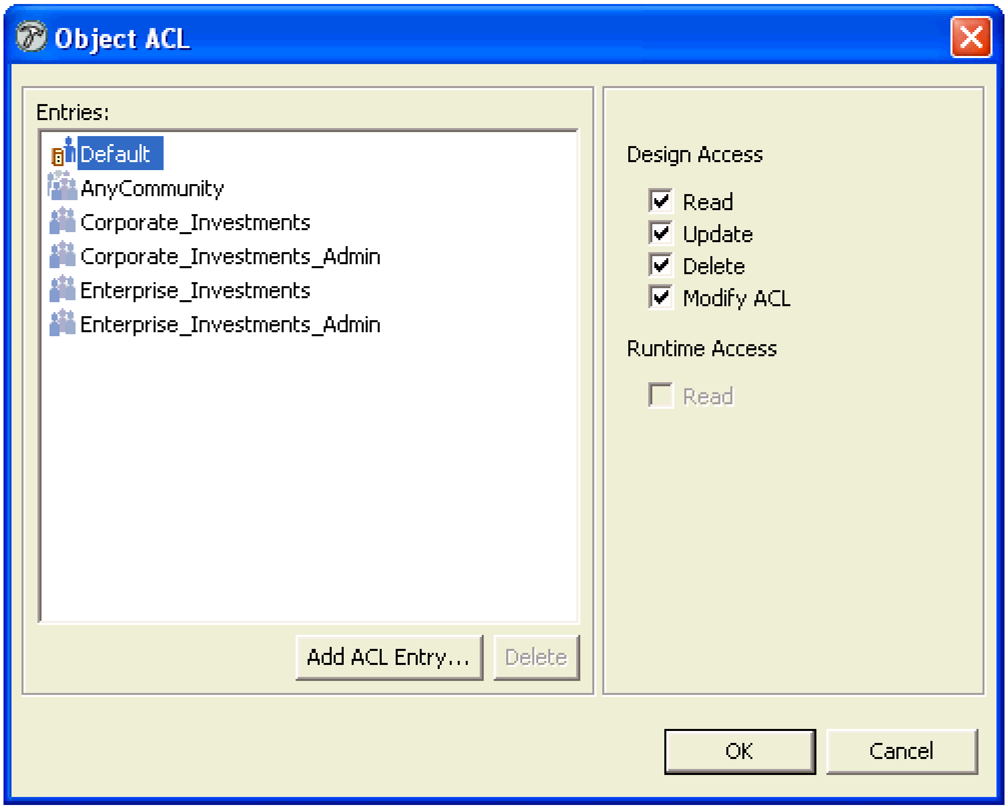
Figure 169: Community Visibility view

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.



Object ACL

Entries:

- Default
- AnyCommunity
- Corporate_Investments
- Corporate_Investments_Admin
- Enterprise_Investments
- Enterprise_Investments_Admin

Design Access

☑ Read
☑ Update
☑ Delete
☑ Modify ACL

Runtime Access

□ Read

Figure 170: 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.

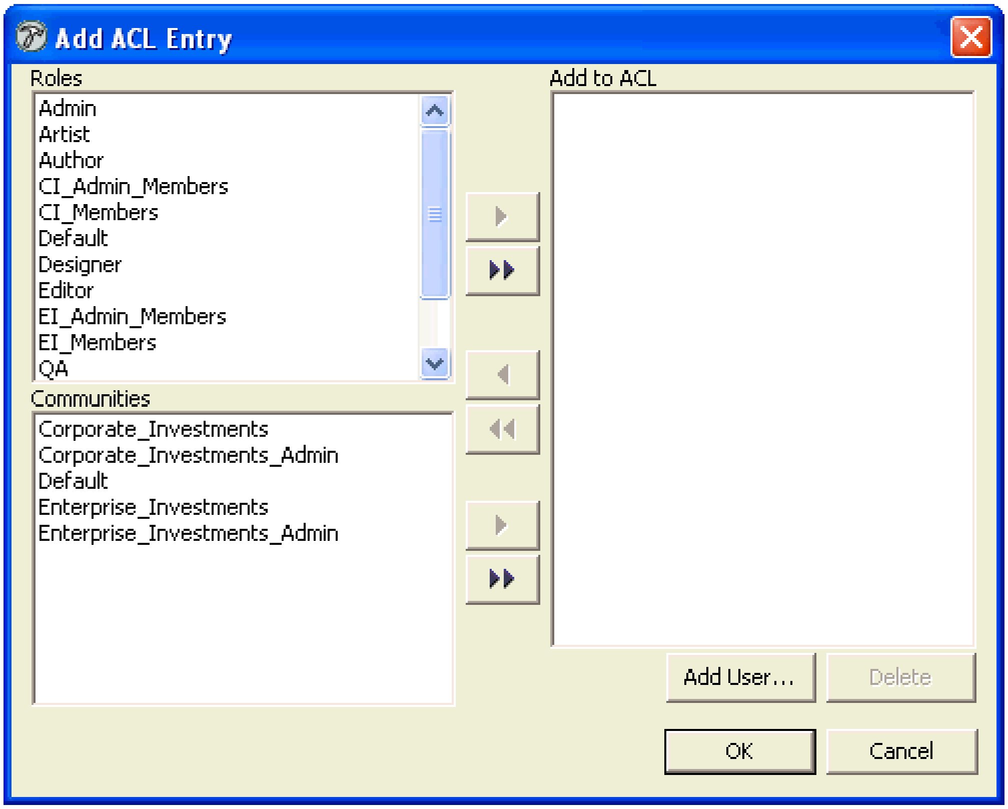


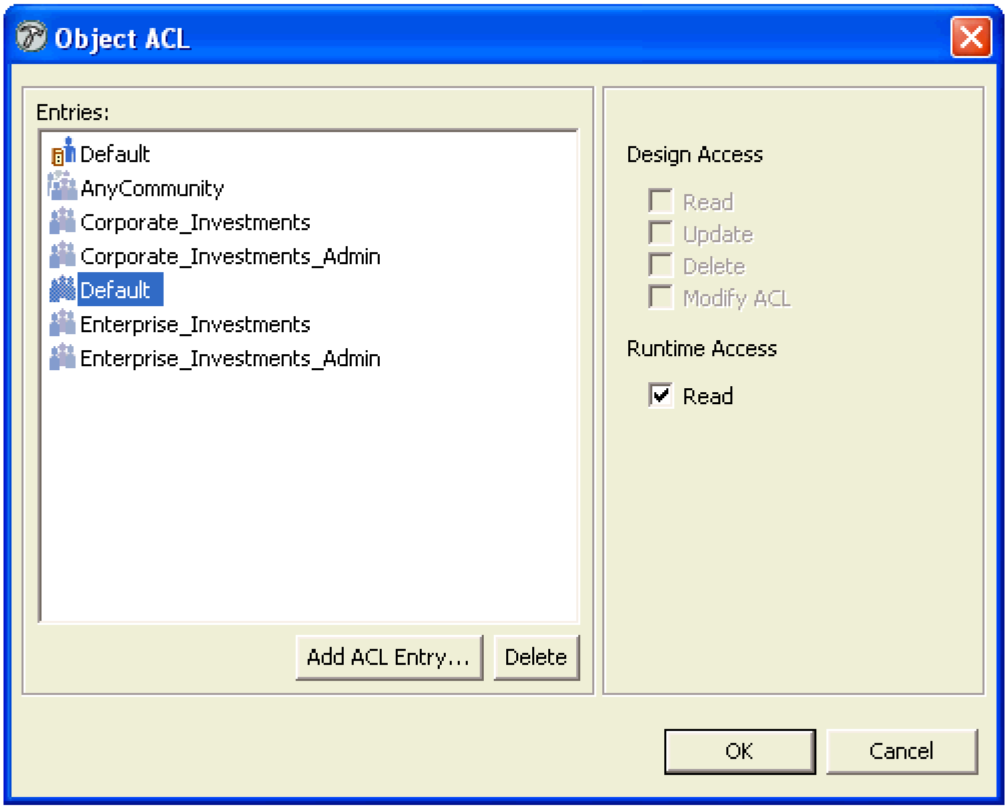
Figure 171: Add ACL Entry dialog

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.



Entries:

- Default
- AnyCommunity
- Corporate_Investments
- Corporate_Investments_Admin
- Default
- Enterprise_Investments
- Enterprise_Investments_Admin

Design Access

□ Read
□ Update
□ Delete
□ Modify ACL

Runtime Access

☑ Read

Figure 172: 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.

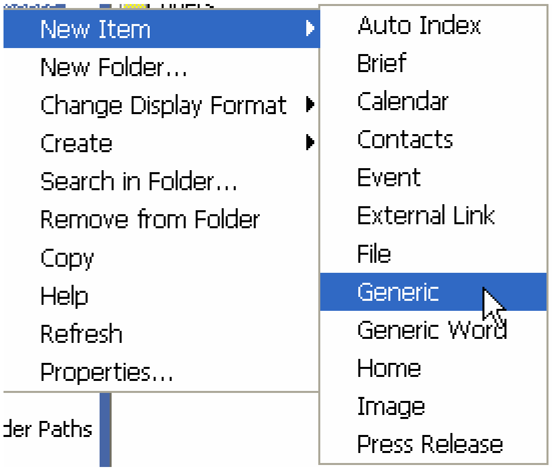
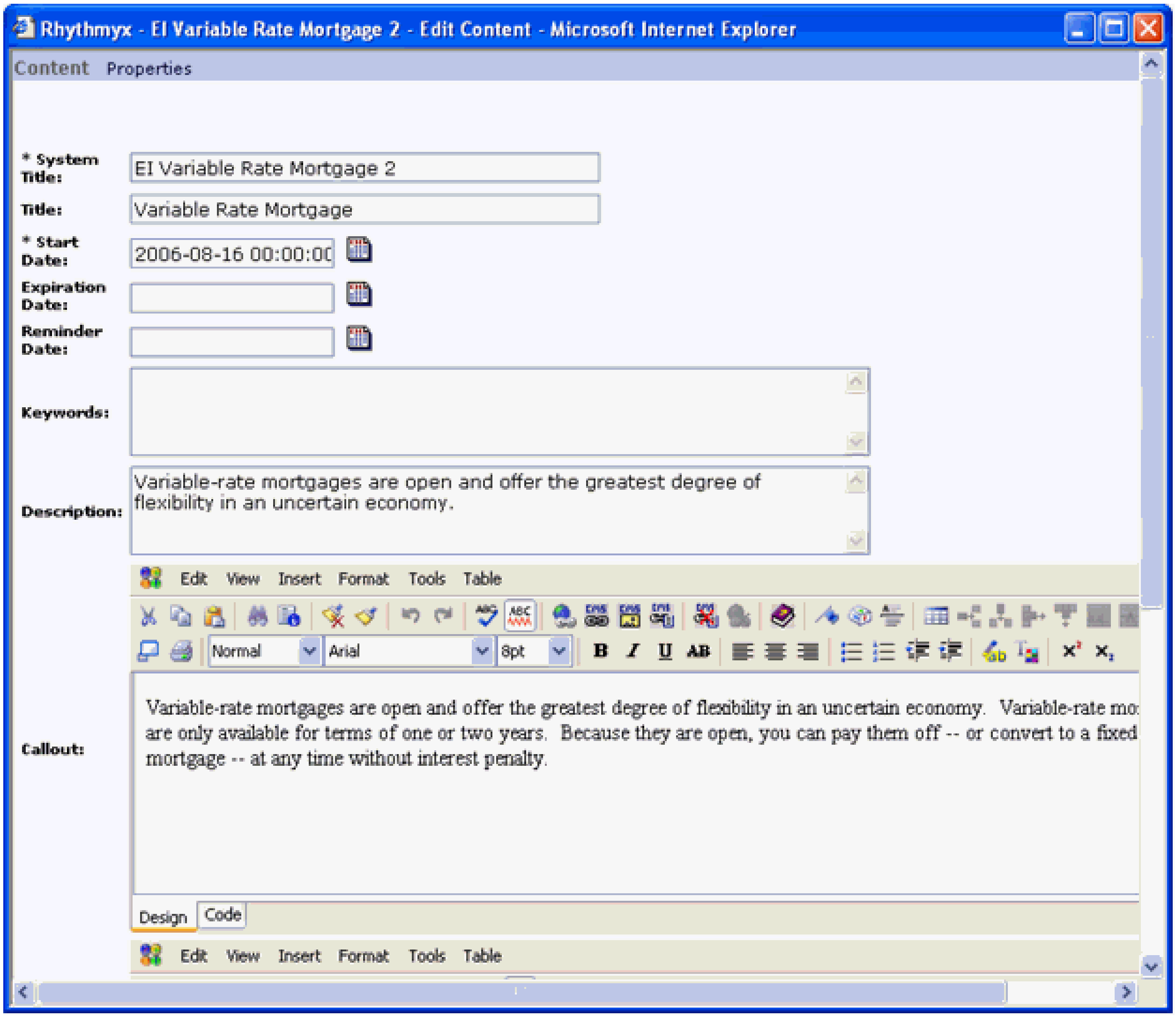


Figure 173: New Content Type in drop menu

The following graphic shows a Content Item entered in your new Content Editor.



Rhythmyx - EI Variable Rate Mortgage 2 - Edit Content - Microsoft Internet Explorer

* System Title: EI Variable Rate Mortgage 2

Title: Variable Rate Mortgage

* Start Date: 2006-08-16 00:00:0C

Expiration Date:

Reminder Date:

Keywords:

Description: Variable-rate mortgages are open and offer the greatest degree of flexibility in an uncertain economy.

Callout:
Variable-rate mortgages are open and offer the greatest degree of flexibility in an uncertain economy. Variable-rate mo:
are only available for terms of one or two years. Because they are open, you can pay them off -- or convert to a fixed
mortgage -- at any time without interest penalty.

Figure 174: Generic Content Editor

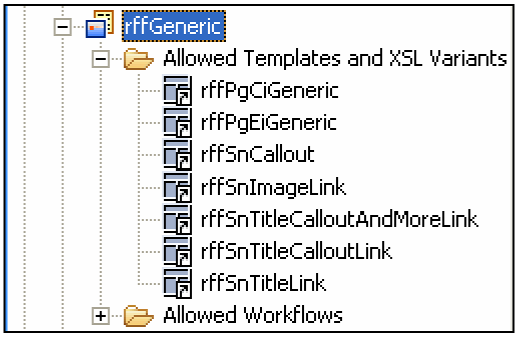
If you refer back to the table in the Generic Content Type specification, you can see that the fields appear in the order in which you entered them. The Content Editor displays the Labels and controls specified for each field. Note that the required fields have an asterisk next to them.

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 113).

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.



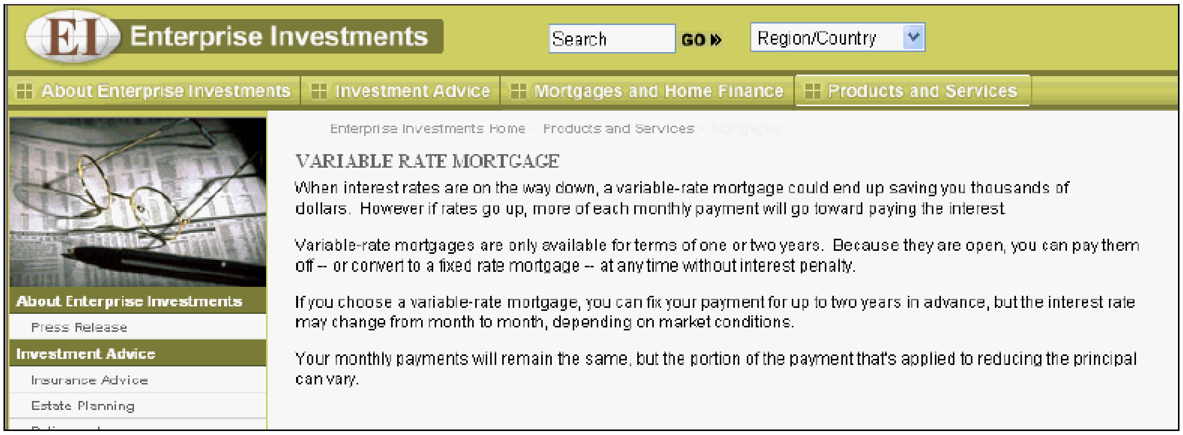
rffGeneric
Allowed Templates and XSL Variants
rffPgCiGeneric
rffPgEiGeneric
rffSnCallout
rffSnImageLink
rffSnTitleCalloutAndMoreLink
rffSnTitleCalloutLink
rffSnTitleLink
Allowed Workflows

Figure 175: 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 125).

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*).


EI
Enterprise Investments
Search
GO
Region/Country
About Enterprise Investments
Investment Advice
Mortgages and Home Finance
Products and Services
Enterprise Investments Home > Products and Services > Mortgages
About Enterprise Investments
Press Release
Investment Advice
Insurance Advice
Estate Planning

VARIABLE RATE MORTGAGE

When interest rates are on the way down, a variable-rate mortgage could end up saving you thousands of dollars. However if rates go up, more of each monthly payment will go toward paying the interest

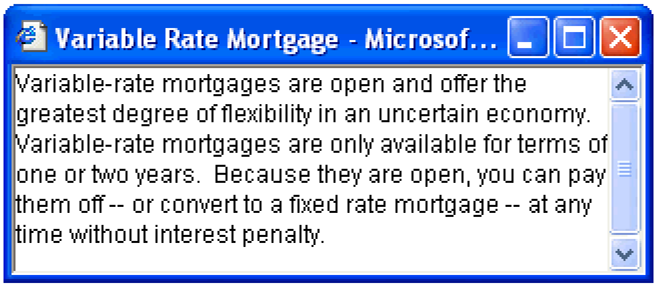
Variable-rate mortgages are only available for terms of one or two years. Because they are open, you can pay them off -- or convert to a fixed rate mortgage -- at any time without interest penalty.

If you choose a variable-rate mortgage, you can fix your payment for up to two years in advance, but the interest rate may change from month to month, depending on market conditions.

Your monthly payments will remain the same, but the portion of the payment that's applied to reducing the principal can vary.

Figure 176: Generic Content Item formatted by page template

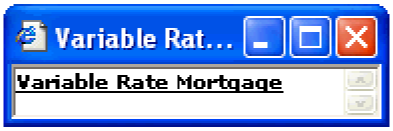
The rffSnCallout template displays the Generic Content Type in the following snippet format. This template only shows the callout (summary) field. To display this snippet on your Web site, you would have to include it on a Page template.



Variable-rate mortgages are open and offer the greatest degree of flexibility in an uncertain economy. Variable-rate mortgages are only available for terms of one or two years. Because they are open, you can pay them off -- or convert to a fixed rate mortgage -- at any time without interest penalty.

Figure 177: 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.



Variable Rate Mortgage

Figure 178: 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 241).

In the chapter ***Creating Shared Fields*** (see page 81), the topic ***sharedimage Field Set*** (see page 84) 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, but in our example, we only use sys_imageInfoExtractor since it encompasses the functionality of sys_FileInfo. For a full list of the fields sys_imageInfoExtractor looks for and the content that it inserts into them, see the topic *sys_imageInfoExtractor* in the *Rhythmyx Technical Reference*. Note that the sharedimage Field Set does not include all of the fields that *sys_imageInfoExtractor* looks for, but only the ones that implementers most commonly use in Content Types. You may include any fields with *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 272) 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 240).
2. ***Including a local field*** (see page 241).
3. ***Entering Properties.*** (see page 243)
4. ***The Image Content Editor*** (see page 246).
5. ***Image Content Items*** (see page 248).
6. ***WebImageFX*** (on page 249).

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.

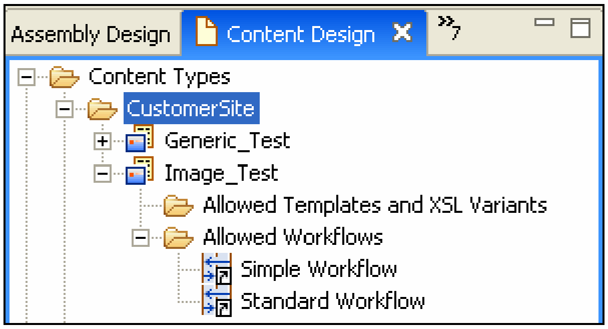


Figure 179: Image_Test Content Type

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 227) 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 227).
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 227) 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 87) and ***Implementing the "sharedimage" Field Set*** (see page 98).

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.

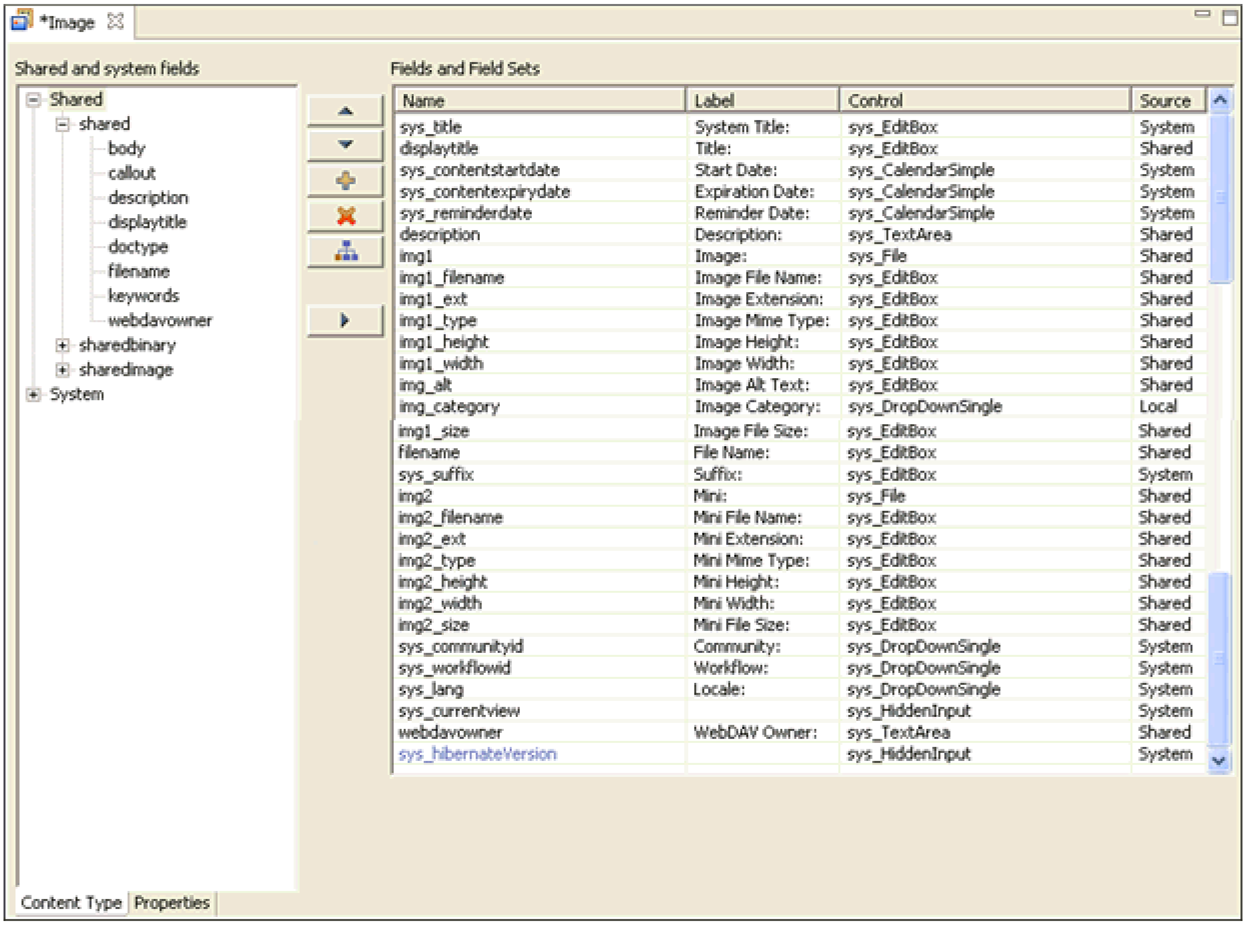
Choose sys_DropDownSingle. Populate the control with the choices from the Keyword *FF Image Category*. For information about the sys_DropDownSingle control and how to populate it with keyword choices, see the topic ***Implementing a List Control*** (see page 95) in the chapter *Creating Shared Fields*. For instructions on creating a Keyword and Keyword Choices, see the topic ***Creating and Using Keywords*** (see page 291) 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 467). 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:



Fields and Field Sets

Name	Label	Control	Source
sys_title	System Title:	sys_EditBox	System
displaytitle	Title:	sys_EditBox	Shared
sys_contentstartdate	Start Date:	sys_CalendarSimple	System
sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	System
sys_reminderdate	Reminder Date:	sys_CalendarSimple	System
description	Description:	sys_TextArea	Shared
img1	Image:	sys_File	Shared
img1_filename	Image File Name:	sys_EditBox	Shared
img1_ext	Image Extension:	sys_EditBox	Shared
img1_type	Image Mime Type:	sys_EditBox	Shared
img1_height	Image Height:	sys_EditBox	Shared
img1_width	Image Width:	sys_EditBox	Shared
img_alt	Image Alt Text:	sys_EditBox	Shared
img_category	Image Category:	sys_DropDownSingle	Local
img1_size	Image File Size:	sys_EditBox	Shared
filename	File Name:	sys_EditBox	Shared
sys_suffix	Suffix:	sys_EditBox	System
img2	Mini:	sys_File	Shared
img2_filename	Mini File Name:	sys_EditBox	Shared
img2_ext	Mini Extension:	sys_EditBox	Shared
img2_type	Mini Mime Type:	sys_EditBox	Shared
img2_height	Mini Height:	sys_EditBox	Shared
img2_width	Mini Width:	sys_EditBox	Shared
img2_size	Mini File Size:	sys_EditBox	Shared
sys_communityid	Community:	sys_DropDownSingle	System
sys_workflowid	Workflow:	sys_DropDownSingle	System
sys_lang	Locale:	sys_DropDownSingle	System
sys_currentview		sys_HiddenInput	System
webdavowner	WebDAV Owner:	sys_TextArea	Shared
sys_hibernateVersion		sys_HiddenInput	System

Figure 180: Content Type tab, Image Content Type

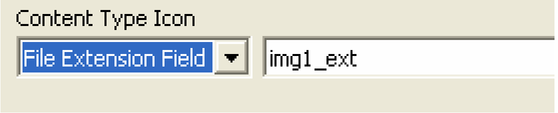
6 Now, to ***enter additional properties*** (see page 243), click the Properties tab of the Content Type editor.

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 the Content Type Icon field, set to *None* by default. When we entered the Generic Content Type, we set this field to *Specified* and indicated a location where a specified file was stored. For the image Content Type, choose *File Extension Field* in the Content Type Icon drop list. The Properties tab now displays a drop list of all text fields in the Image Content Type next to the Content Type Icon field. In the drop list, choose the field in the Content Type that holds the uploaded file's extension. In this case, choose the field img1_ext:

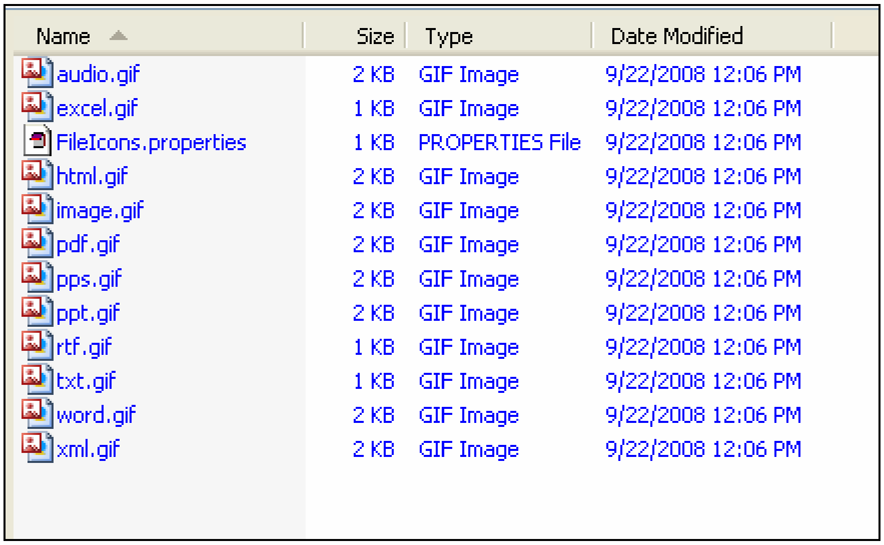


Content Type Icon

File Extension Field ▼ img1_ext

Figure 181: File Extension Field option

When you choose *File Extension Field*, Rhythmyx locates the icon filename in one of two files that map extensions to icon files. First Rhythmyx looks at the file <Rhythmyx root>\rx_resources\images\ContentTypeIcons\FileIcons\FileIcons.properties, and if it does not find a mapping for the extension, it looks in the file <Rhythmyx root>\sys_resources\images\ContentTypeIcons\FileIcons\FileIcons.properties. The FileIcons.properties file in the sys_resources folder holds Rhythmyx-provided extension/icon mappings. The FileIcons.properties file in the rx_resources folder holds user-provided extension/icon mappings. These mappings are for additional extensions or override the mappings for extensions in the sys_resources file. The rx_resources FileIcons.properties file may also contain Content Type/icon mappings for Content Types that have no matching extension/icon mapping. If neither of these types of mappings is found in one of the files, the system uses the default 🗋icon to represent the content item. Note that the files that store the icons must be present in the same folder as the FileIcons.properties file that maps them. For example, the <Rhythmyx root>\rx_resources\images\ContentTypeIcons\FileIcons folder holds the following contents:



Name	Size	Type	Date Modified
audio.gif	2 KB	GIF Image	9/22/2008 12:06 PM
excel.gif	1 KB	GIF Image	9/22/2008 12:06 PM
FileIcons.properties	1 KB	PROPERTIES File	9/22/2008 12:06 PM
html.gif	2 KB	GIF Image	9/22/2008 12:06 PM
image.gif	2 KB	GIF Image	9/22/2008 12:06 PM
pdf.gif	2 KB	GIF Image	9/22/2008 12:06 PM
pps.gif	2 KB	GIF Image	9/22/2008 12:06 PM
ppt.gif	2 KB	GIF Image	9/22/2008 12:06 PM
rtf.gif	1 KB	GIF Image	9/22/2008 12:06 PM
txt.gif	1 KB	GIF Image	9/22/2008 12:06 PM
word.gif	2 KB	GIF Image	9/22/2008 12:06 PM
xml.gif	2 KB	GIF Image	9/22/2008 12:06 PM

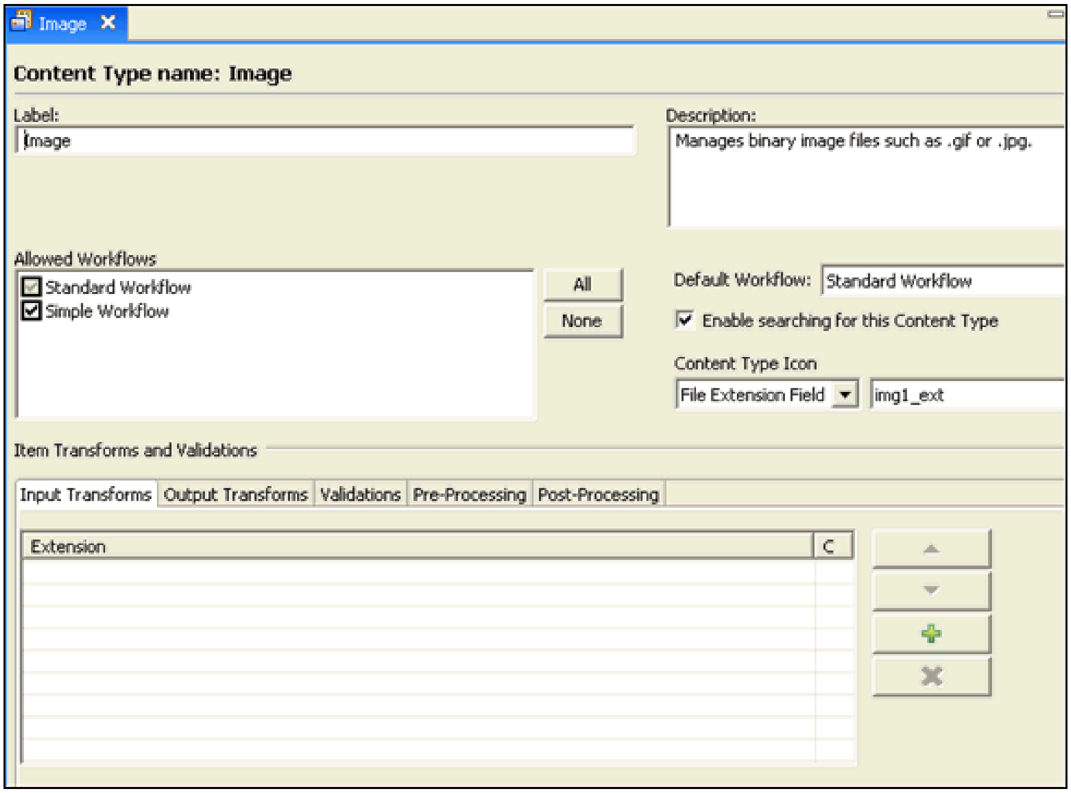
Figure 182: FileIconProperties folder

For more information about configuring icons to represent Content Types, see the section *Adding Custom Icons for Content Types* in the document *Customizing the Active Assembly and Content Explorer Interfaces*.

The Properties tab also 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 271) for definitions of item transforms, validations, and pre and post-processing extensions.
- See ***Adding Pre-processing Extensions*** (see page 272) for instructions on adding the pre-processing extensions that the Image Content Type requires.



Content Type name: Image

Label: Image

Description: Manages binary image files such as .gif or .jpg.

Allowed Workflows

☑ Standard Workflow

☑ Simple Workflow

Default Workflow: Standard Workflow

☑ Enable searching for this Content Type

Content Type Icon: File Extension Field ▾ img1_ext

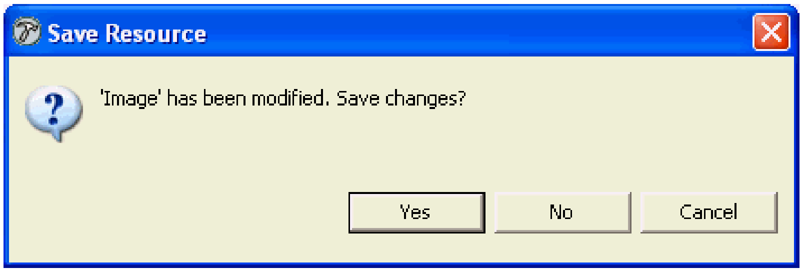
Figure 183: Image Content Type Properties Tab

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:



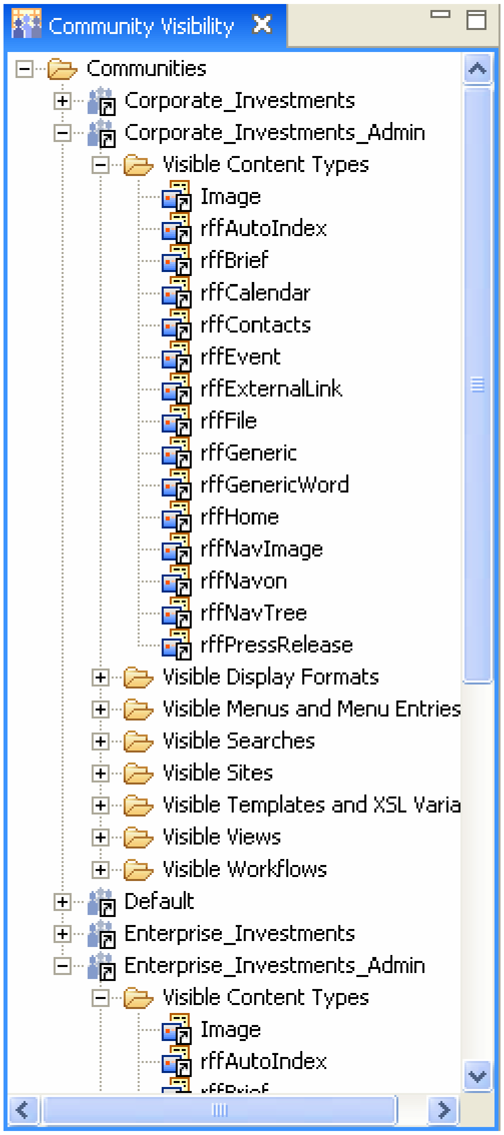
'Image' has been modified. Save changes?

Figure 184: 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*:



- Communities
  - Corporate_Investments
  - Corporate_Investments_Admin
    - Visible Content Types
      - Image
      - rffAutoIndex
      - rffBrief
      - rffCalendar
      - rffContacts
      - rffEvent
      - rffExternalLink
      - rffFile
      - rffGeneric
      - rffGenericWord
      - rffHome
      - rffNavImage
      - rffNavon
      - rffNavTree
      - rffPressRelease
    - Visible Display Formats
    - Visible Menus and Menu Entries
    - Visible Searches
    - Visible Sites
    - Visible Templates and XSL Varia
    - Visible Views
    - Visible Workflows
  - Default
  - Enterprise_Investments
  - Enterprise_Investments_Admin
    - Visible Content Types
      - Image
      - rffAutoIndex
      - rffBrief

Figure 185: Communities with Image Content Type

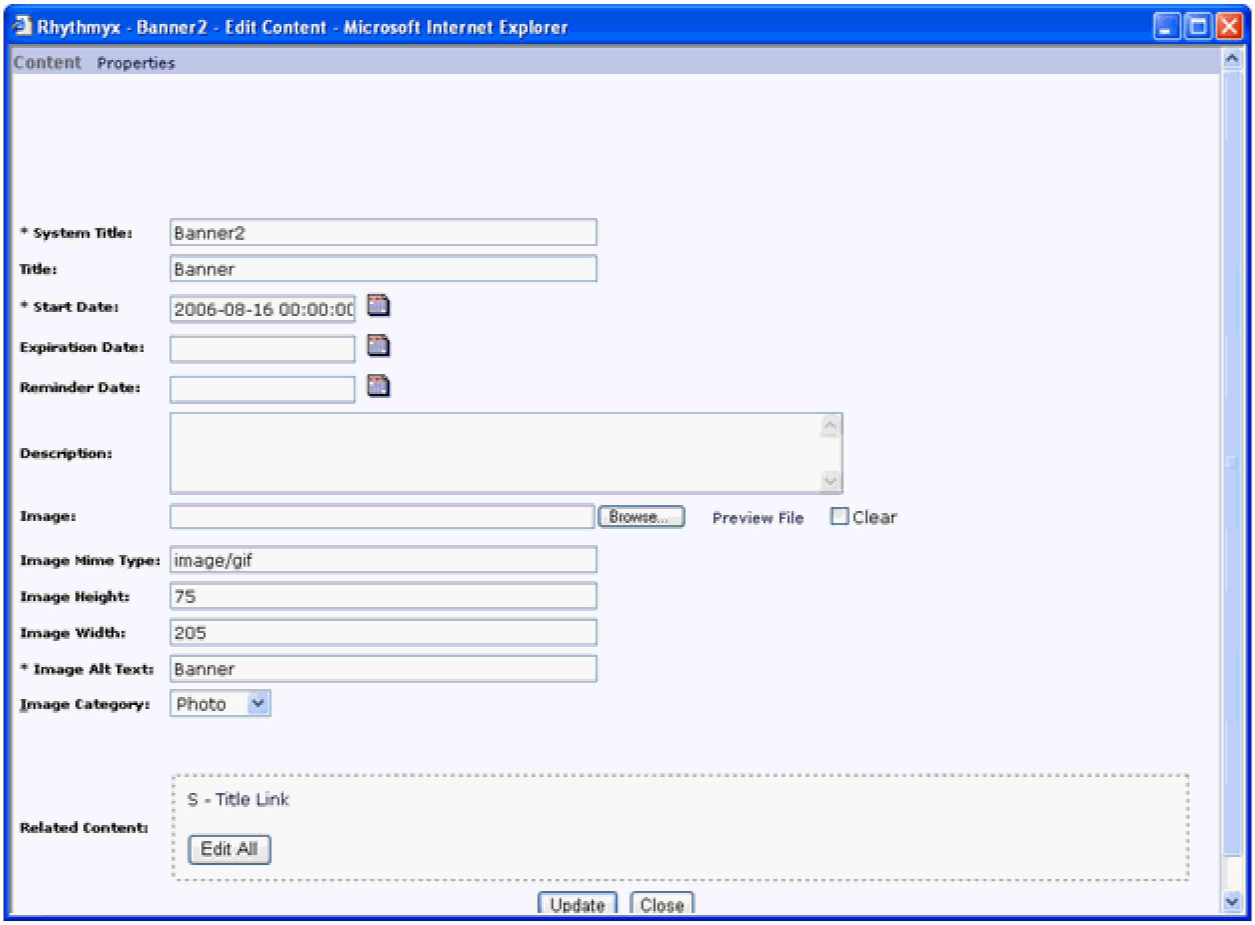
Now, an administrator in Content Explorer can open your Image Content Editor.

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 272) 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.



* System Title: Banner2
Title: Banner
* Start Date: 2006-08-16 00:00:00
Expiration Date:
Reminder Date:
Description:
Image:
Image Mime Type: image/gif
Image Height: 75
Image Width: 205
* Image Alt Text: Banner
Image Category: Photo
Related Content:

Figure 186: 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.

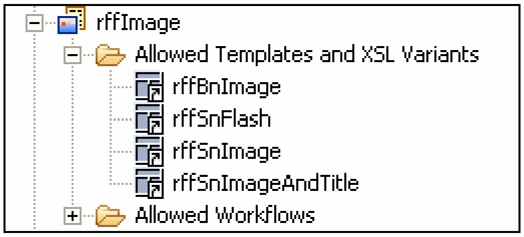
Field visibility rules (see page 104) associated with the Image Extension, Image File Size, Filename, and webDAVOwner fields as well as all of the thumbnail fields prevent them from appearing in the Content Editor. The internal sys_currentView field does not appear because it is assigned a sys_hiddenInput 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:



- rffImage
  - Allowed Templates and XSL Variants
    - rffBnImage
    - rffSnFlash
    - rffSnImage
    - rffSnImageAndTitle
  - Allowed Workflows

Figure 187: 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.

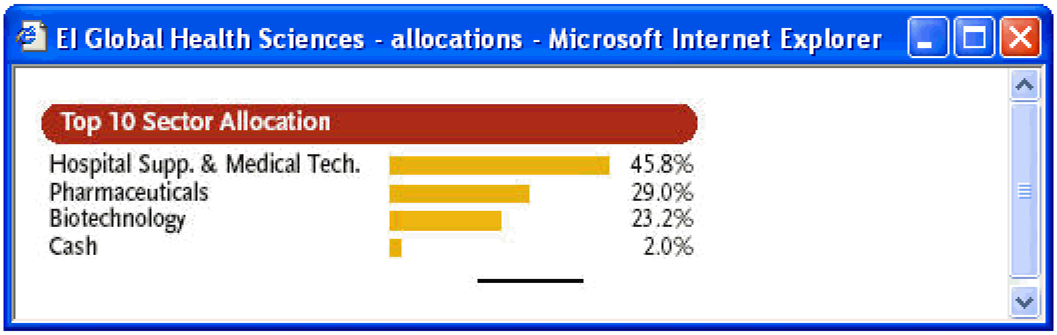

EI Global Health Sciences - allocations - Microsoft Internet Explorer
Top 10 Sector Allocation
Hospital Supp. & Medical Tech.
45.8%
Pharmaceuticals
29.0%
Biotechnology
23.2%
Cash
2.0%

Figure 188: 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:

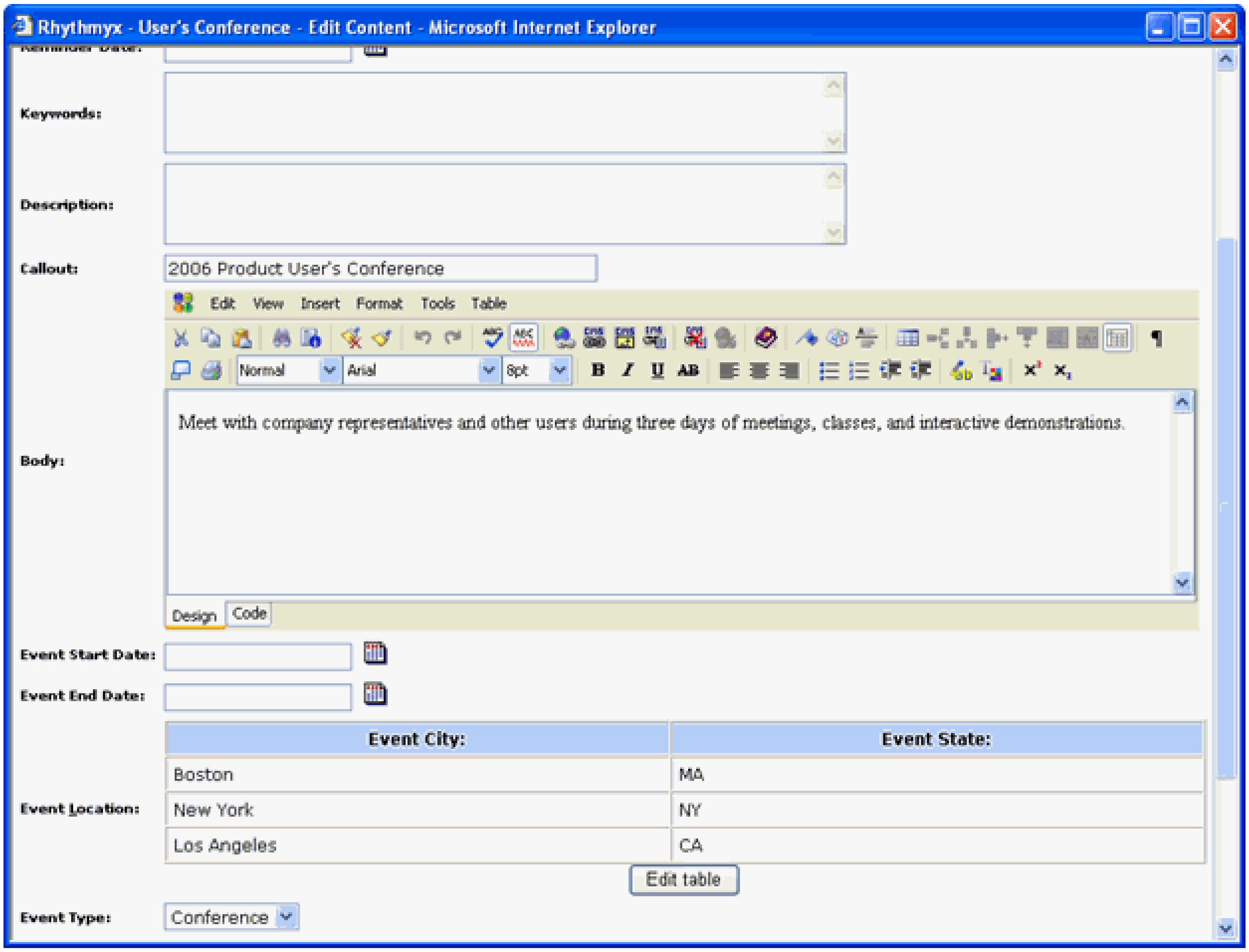
- The field containing the sys_WebImageFX control must be named *uploadfilephoto*.
- Due to the restriction on the name of the field, a Content Type cannot have more than one field that uses the sys_WebImageFX control. Only the first field that uses the control will be able to upload and edit graphics. The other controls will not be able top 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 demonstrates the process of creating a Content Type with a child field set, as well as some other features you can include in your Content Editors.

A child field set is a table with any number of sub-fields; the child field data is stored in a separate table from the table used to store the data of the parent Content Editor. The following graphic illustrates the Content Editor with the example child field set we will create in this section. The child field set is rendered as a table with entries in the Content Editor.


Rhythmyx - User's Conference - Edit Content - Microsoft Internet Explorer
Edit
View
Insert
Format
Tools
Table
Normal
Arial
8pt
Design
Code
Edit table

Keywords:

Description:

Callout: 2006 Product User's Conference

Body: Meet with company representatives and other users during three days of meetings, classes, and interactive demonstrations.

Event Start Date:

Event End Date:

Event Location:

Event City:	Event State:
Boston	MA
New York	NY
Los Angeles	CA

Event Type: Conference

Figure 189: Content Editor with child field set

To illustrate the implementation of a child field set, we will replace the *event_location* field in the FastForward Event Content Editor with a child field set detailing the location. The child field set consists of four fields: *event_city*, *event_state*, *event_address*, and *event_contact*.

The specification for the event_location child field set is:

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_DropdownSingle	required	integer	4

The data for the event_contact drop down will be retrieved from an external repository. The Northwind database available for Microsoft SQL Server provides a convenient source for this data. We will use data from the Employees table to populate this field. We must set up a connection to the Northwind database and create a Rhythmyx application to look up the data to populate the field.

To demonstrate how to override a shared field, we will also change 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 ***specification of the Event Content Type*** (see page 457) for details about the implementation of this Content Type in FastForward.

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 223) and follow the same instructions using the specifications in the ***Event Content Type*** (see page 220) topic instead. Once you have created the Content Type object, work through the other topics in this section.

Note: The data included in the following procedures is example data. Substitute the data from your own implementation plan when implementing your Content Editors.

This section includes the following steps for completing and viewing the Event Content Type:

1. ***Overriding a shared field*** (see page 251)
2. ***Including a child field set*** (see page 252)
3. ***Populating a field from an external lookup*** (see page 256).
4. ***Viewing the Event Content Editor*** (see page 267).
5. ***Viewing Event Content Items*** (see page 270).

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 227) 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. Override a shared field if you want the field to have a different property only within a specific Content Type. If you want a shared field to have a different property in all Content Types that use it, change the value directly in the shared field. If you want a system field to have a different property in all Content Types, create a shared field with the new property and use that shared field 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 227) and ***add a local field to the Image Content Type*** (see page 241). When you reach the shared/callout field, follow the steps below.

To override the properties of the shared/callout field:

1. Add 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 252).

Adding a Child Field Set

At this point, we will demonstrate how to add the event_location child field set to the Event Content Type.

Let's review the repeat the definition of the event_location field set:

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_DropDownSingle	required	int	4

Note: For simplicity, in this procedure, the event_contact field will be created with the default sys_EditBox control. We will change to the sys_DropDownSingle control in a later procedure.

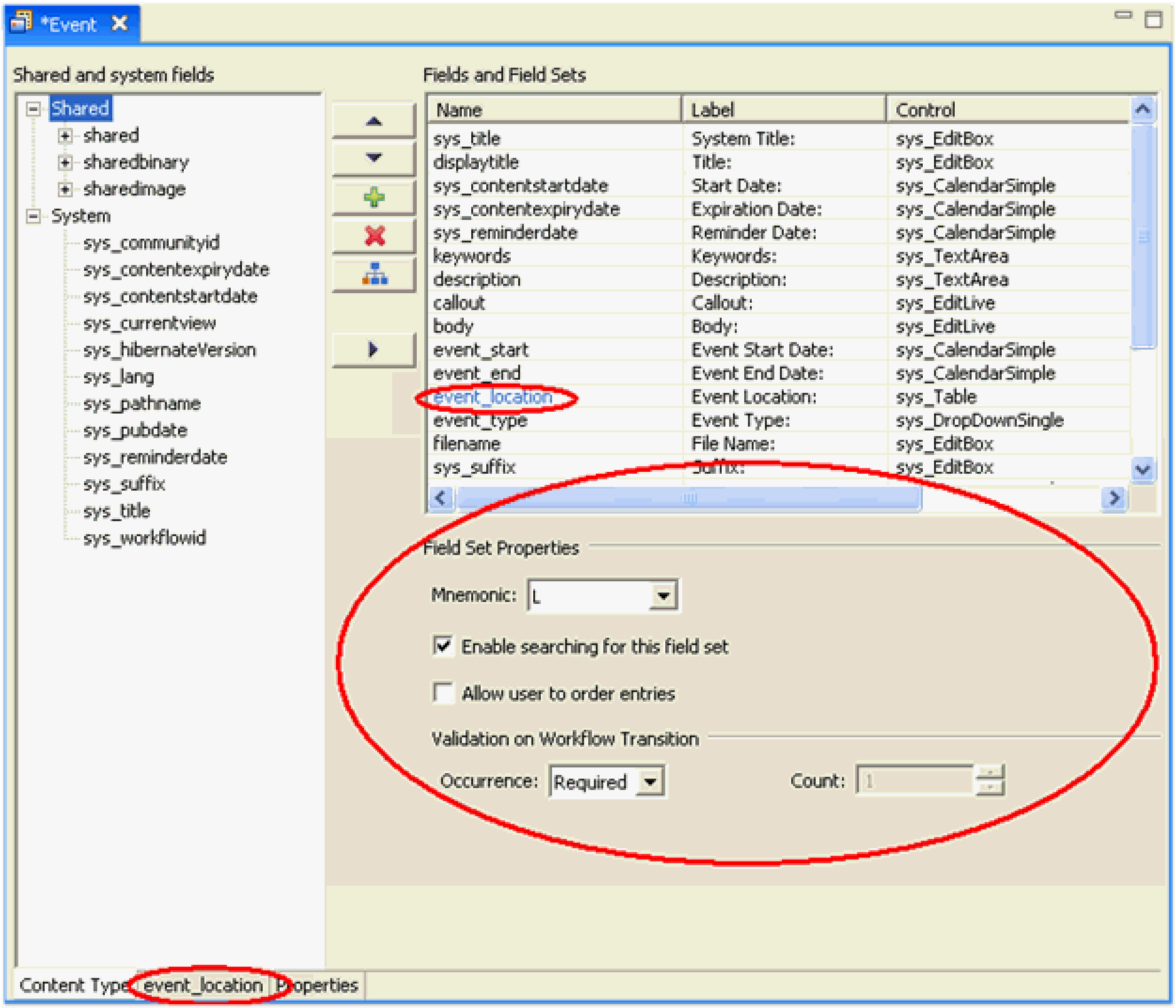
To add the event_location child field set:

1. In the rffEvent Content Editor, 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*:.



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	Event Type:	sys_DropDownSingle
filename	File Name:	sys_EditBox
sys_suffix	Suffix:	sys_EditBox

Field Set Properties

Mnemonic: L

☑ Enable searching for this field set

□ Allow user to order entries

Validation on Workflow Transition

Occurrence: Required Count: 1

Figure 190: Event Content Type with Child Field Set Added

3 Under Field Set Properties:

a) In Mnemonic, choose *L* since the field set stores location information.

b) Leave Enable searching for this field set checked, since users may search for the Content Item by its location.

c) Uncheck Allow user to order entries, since you want the location fields to be ordered consistently in all Event Content Items.

d) 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.

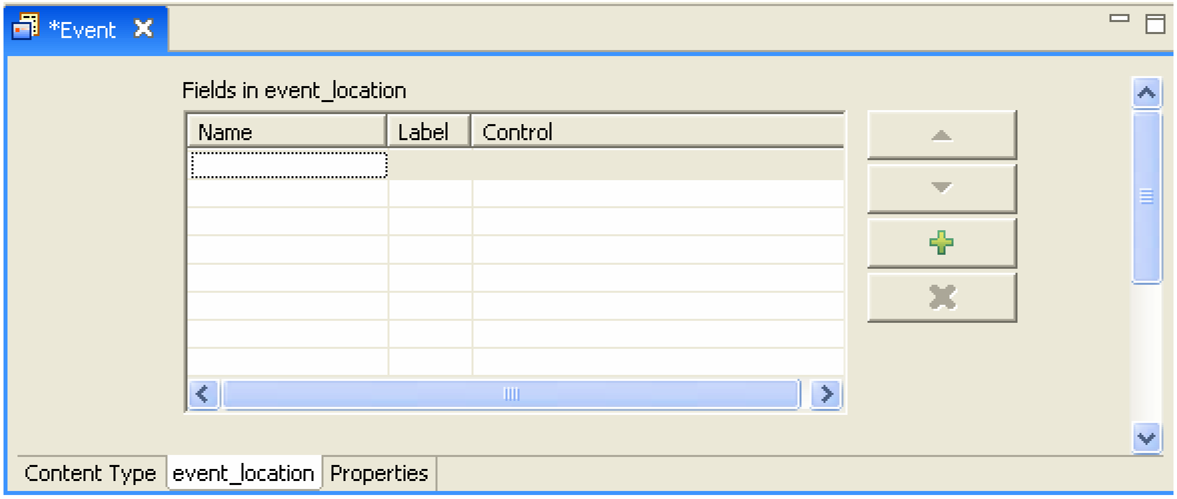


Figure 191: Child Field Set Editor

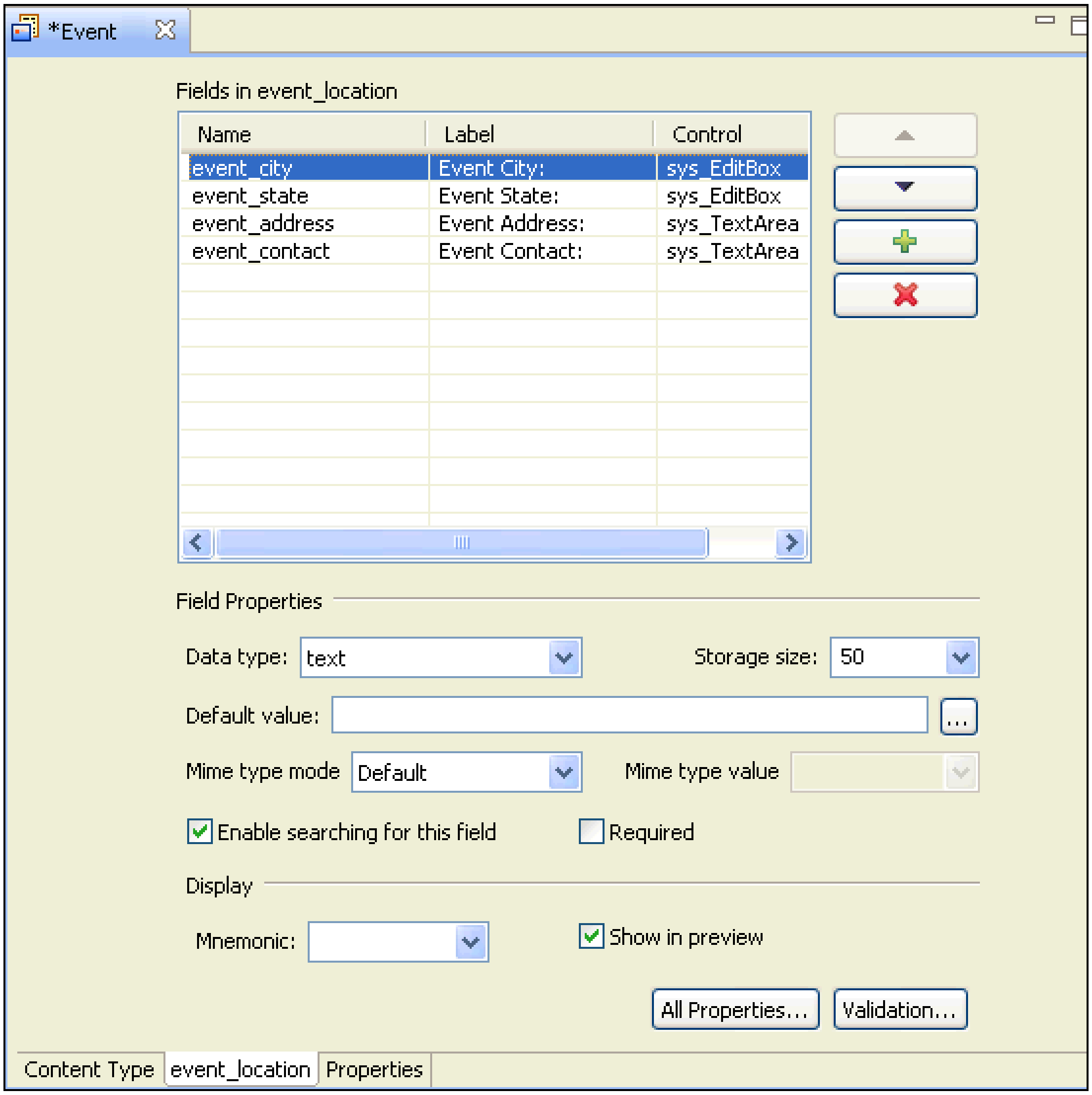
5 Enter the child field set entries and their properties (as defined in the table above) into the Fields in event_location table. For most properties, enter values exactly as you would enter values for local fields. For help, see ***Including a Local Field*** (see page 241).

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 267) for a more detailed explanation.

7 Clicking the [**Validation**] button opens the Field Visibility dialog, as with fields on the parent Content Editor. For details about adding field visibility rules, see ***Field Visibility, Validation, Read Only, and Transform Rules*** (see page 104). You can also add validation, read-only, and transform rules to child fields.

When you finish entering the field definitions, the event_location tab should resemble the following screenshot:



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

Field Properties

Data type: text

Storage size: 50

Default value:

Mime type mode Default

Mime type value

☑ Enable searching for this field

□ Required

Display

Mnemonic:

☑ Show in preview

Figure 192: Event 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 291).

10 Add the remaining fields in the table in the ***rffEvent Content Type specification*** (see page 457) to the Content Type tab.

When the Content Editor is complete, it should resemble the screenshot in the topic ***Creating a Content Type with a Child Field Set*** (see page 250).

11 Click the Properties tab. The only property that you want to change is the Content Type Icon. Change the value of this field to *Specified*. In the search dialog that opens, navigate to the icon <Rhythmyx root>/rx_resources/Images/ContentTypeIcons/FileIcons/rffEvent.gif and click [**Open**] to choose the rffEvent.gif icon to represent the Event Content Type.

12 Save and close the Content Type editor.

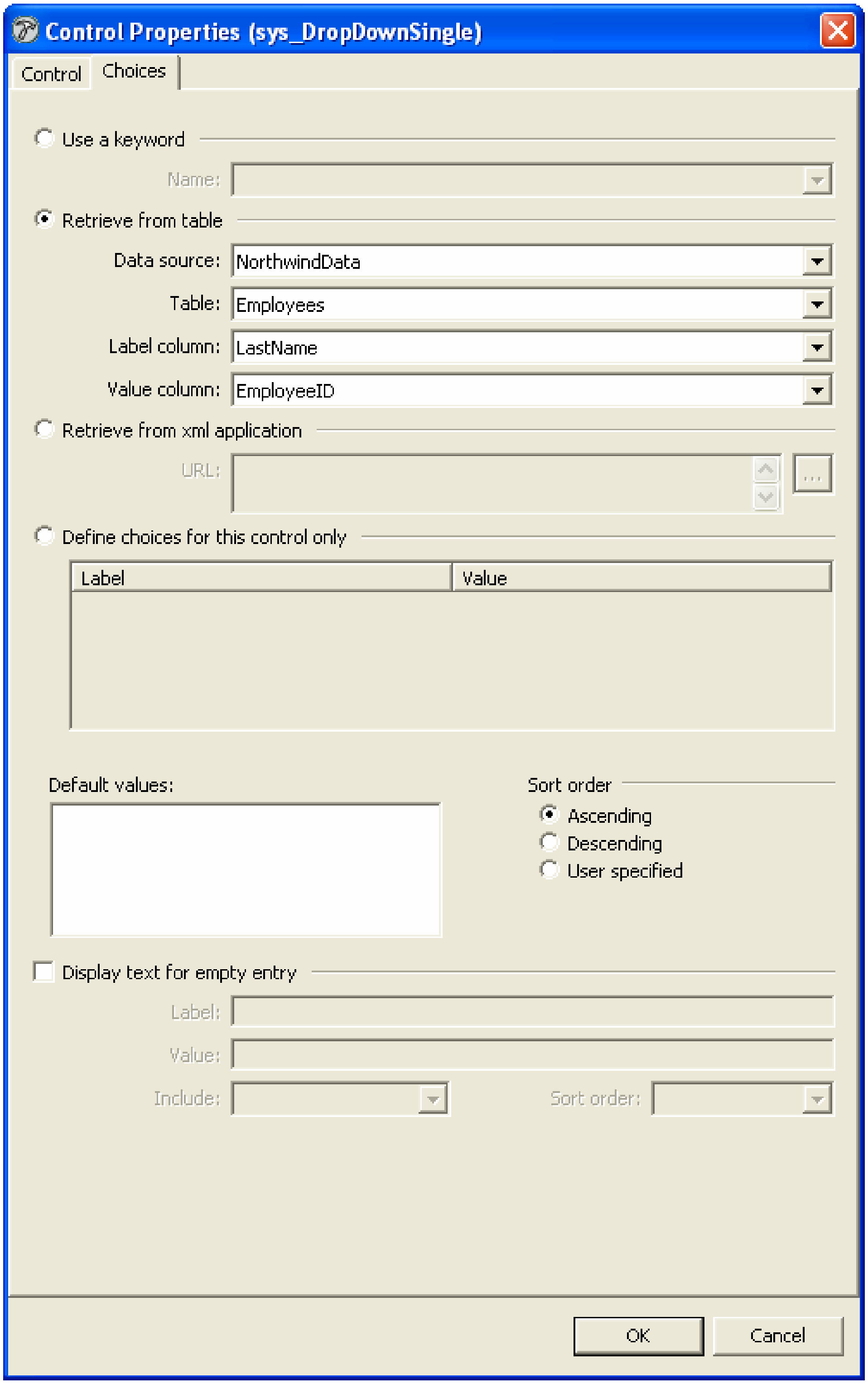
Populating a Field from an External Lookup

When implementing a list control (such as sys_DropDownSingle, sys_RadioButtons, or sys_CheckBoxGroup), you must specify the source of the choices for the list. While it is common to use a Rhythmyx Keyword to define the choices, often the choices are stored in an external repository and must be retrieved when editing and assembling a Content Item.

In our example, we will use the Employees table in the Northwind database available with Microsoft SQL Server. Among the columns in this table are:

- Last Name
- First Name
- Phone Number
- Country

One option when retrieving data from a remote repository is to select the **Retrieve from Table** option on the Choices tab of the Control Properties dialog.



Control Properties (sys_DropDownSingle)

Control | Choices

○ Use a keyword

Name:

● Retrieve from table

Data source: NorthwindData

Table: Employees

Label column: LastName

Value column: EmployeeID

○ Retrieve from xml application

URL:

○ Define choices for this control only

Label	Value

Default values:

Sort order

● Ascending

○ Descending

○ User specified

□ Display text for empty entry

Label:

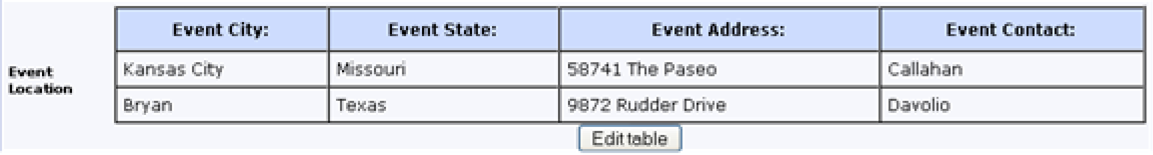
Value:

Include:

Sort order:

Figure 193: Retrieving data from a table in a remote repository

In this example, the control will display the value in the Last Name column in the Content Editor, but will store the value in the Employee ID column in the Repository.



	Event City:	Event State:	Event Address:	Event Contact:
Event Location	Kansas City	Missouri	58741 The Paseo	Callahan
	Bryan	Texas	9872 Rudder Drive	Davolio

Figure 194: Using data from a remote repository

While using the Employee ID as the key to the data in the Northwind repository works for retrieving the data, the Retrieve from Table option only allows one column value for the Label. When you want to display data in multiple columns, you need to use the Retrieve from xml application option, which uses a Rhythmyx XML application to retrieve the data from the external repository. For our example, it would be more useful to display the full name of the contact along with a phone number. That is the information that would be included in the published pages, so it would be more useful to the business users to see this information.

Creating a Connection to an External Repository

Before you can retrieve data from an external repository, you must create a connection to it. Use the Datasources tab of the Rhythmyx Server Administrator to create the connection.

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

1. Start the Rhythmyx Server Administrator and log in. For details about starting and logging in to the Rhythmyx Server Administrator, see ***Starting the Rhythmyx Server Administrator*** (on page 18).
2. Click on the Datasources tab at the top of the dialog.
3. Click on the Connection tab at the bottom of the dialog.
4. Click the [**Add**] button.

   The Rhythmyx Server Administrator displays the Connection Configuration dialog.
5. In the Name field, enter *NorthwindData*.
6. In the JNDI Datasource drop list, leave *jdbc/RhythmyxData* selected.
7. In the Database field, enter *Northwind*.
8. In the Schema/Origin field, enter *dbo*.
9. Click the [**OK**] button.

To test your connection

1. Log in to the Rhythmyx Workbench.
2. Open the Database Explorer view.

   You should see the connection to the Northwind connection listed. If the connection is not listed, the connection data is incorrect. Check the connection and correct the data.
3. Double-click on the Northwind connection expand it. Double-click on the TABLES node to expand it.

You should see the tables defined in the Northwind database. If you do not see the tables listed, the connection data is incorrect. Check the connection and correct the data.

Creating a Lookup Application

When you want to use data from multiple fields in a list control, you must use a Rhythmyx XML application to lookup and concatenate the data. In our example, we will create an application that will retrieve the following data:

- EmployeeID (Choice value, stored in the Rhythmyx Repository)
- LastName
- FirstName
- HomePhone

The values in the LastName, FirstName, and HomePhone fields will be concatenated as the Choice Label:

```
Davolio, Nancy, at (206) 555-9857
```

In a Rhythmyx XML application, data from a database table is mapped to an XML DTD. When the application is running, it can be queried to produce an XML document containing the database data structured as specified in the mapping. We will use a Rhythmyx UDF extension to retrieve and concatenate the data for the label. We also want to return only Employees from the US, so we will create criteria to ensure that only rows where the value of the Country column is USA will be returned.

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

To create the lookup application:

1. In the Rhythmyx Workbench, click on the XML Server tab.
2. Right-click the Applications Folder and from the popup menu, choose *New > Application*.

   The Rhythmyx Workbench displays the New Application dialog
3. In the Application Name field, enter *rffLookupNameAndNumber*. Optionally, enter a Description.
4. Click the [**Finish**] button.

   Rhythmyx opens the rffLookupNameAndNumber application in the Workbench.
5. Create the lookup resource.

   a) Change to the File Explorer tab. Expand the drive where you installed Rhythmyx, then expand the DTD Folder.

   b) Find the sys_Lookup.dtd. Click and drag it to the application window.

   c) From the popup menu, choose *Query*.

   Rhythmyx creates the new query resource.
6. Change the resource name.

   a) Right-click on the resource and from the popup menu, choose *Request Properties*.

   b) In the Request Name field change the default value, *sys_Lookup*, to *EmployeeLookupNameAndNumber*.

c) Click the [**OK**] button to save your change.

7 Double-click the EmployeeLookupNameAndNumber resource to display the Data Pipe.

A Data Pipe consists of a Back End Data Tank (not included when the resource is first created), a Selector, a Mapper, and the Front End Data Tank (which contains the DTD).

8 Add the Employees table and Back End Data Tank to the resource.

a) In the Rhythmyx Workbench, change to the Database Explorer tab.

b) Expand the Northwind Data node.

c) Expand the TABLES node.

d) Click on the Employees table and drag it to the left end of the Data Pipe.

The table should attach to the Data Pipe, adding the Back End Data Tank.

9 Define the Selection criteria

The Selector is used to define the query that selects data from the tables in the Back End Data Tank. (Although we added only one table in this exercise, multiple tables can be added, and joins can be defined between the tables in the Back End Data Tank.) We will define a query that selects rows from the Employee table where Country = USA.

a) Double-click on the Selector .

b) The Rhythmyx Workbench displays the Selector Properties dialog.

Two options are available to define the selector query. Our query is simple enough to use the WHERE table, which is most often the case. If you need to use a complex SQL statement to select the data (for instances, if you need to use inner or outer joins between tables) you can enter the SQL manually. The WHERE table is the default option.

c) In the first empty row of the WHERE table, click in the Variable column, then click on the arrow button. From the popup menu, choose *Single Value*.

The Rhythmyx Workbench displays the Value Selector dialog.

d) In the Type drop list, select *Backend Column*.

The Value Selector displays a list of columns from the Back End Data Tank.

e) Select the *Employees.EmployeeId* column. Selecting the column adds it to the Value field.

f) Click the [**OK**] button to save your choice.

g) Click in the Op column of the same row and from the drop list, choose the equals sign ("=").

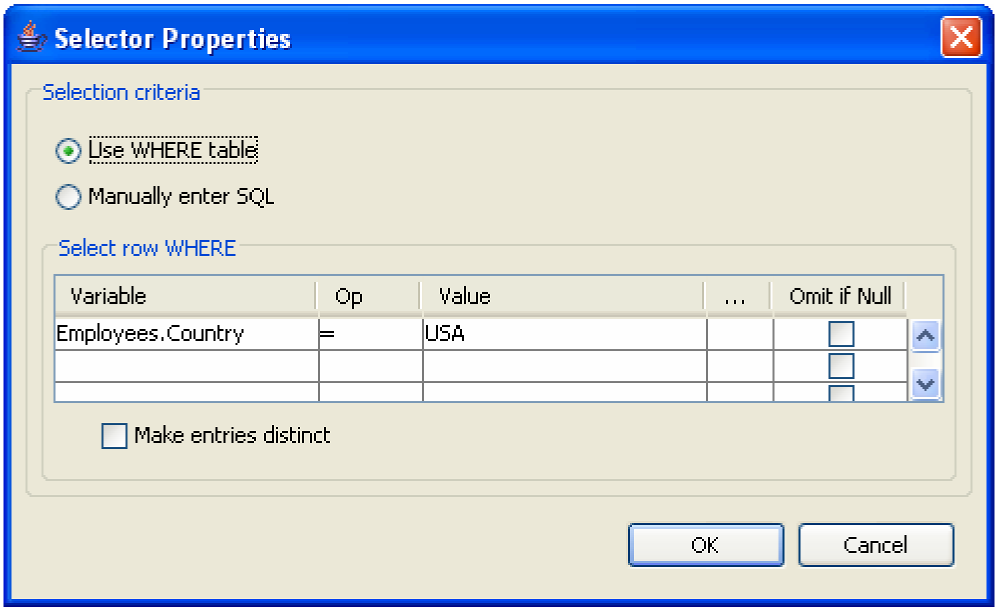
h) Repeat Step "c" in the Value column.

i) In the Type drop list of the Value Selector dialog, choose Literal

j) In the Value field of the Value Selector dialog, enter *USA*.

k) Save your choice.

l) In the Selector Properties dialog, click the [**OK**] button to save the selection criteria.



Selector Properties

Selection criteria

● Use WHERE table

○ Manually enter SQL

Select row WHERE

Variable	Op	Value	...	Omit if Null
Employees.Country	=	USA		□
				□
				□

□ Make entries distinct

Figure 195: Selector for lookup application

The selection defined here is equivalent to the following SQL statement:

```
select * from Employees where Employees.Country = 'USA'
```

10 Click on the Mapper .

The Rhythmyx Workbench displays the Mapper Properties dialog. This dialog is used to map the data returned by the Selector to the elements and attributes of the DTD.

11 Map the EmployeeID column from the Back End Data Tank to the Value element of the sys_Lookup DTD.

a) Click on the *EmployeeID* column in on the left side of the mapper and drag it to the first empty row of the Backend column.

b) Click on the *Value* node on the right side of the mapper and drag it to the XML column of the same row.

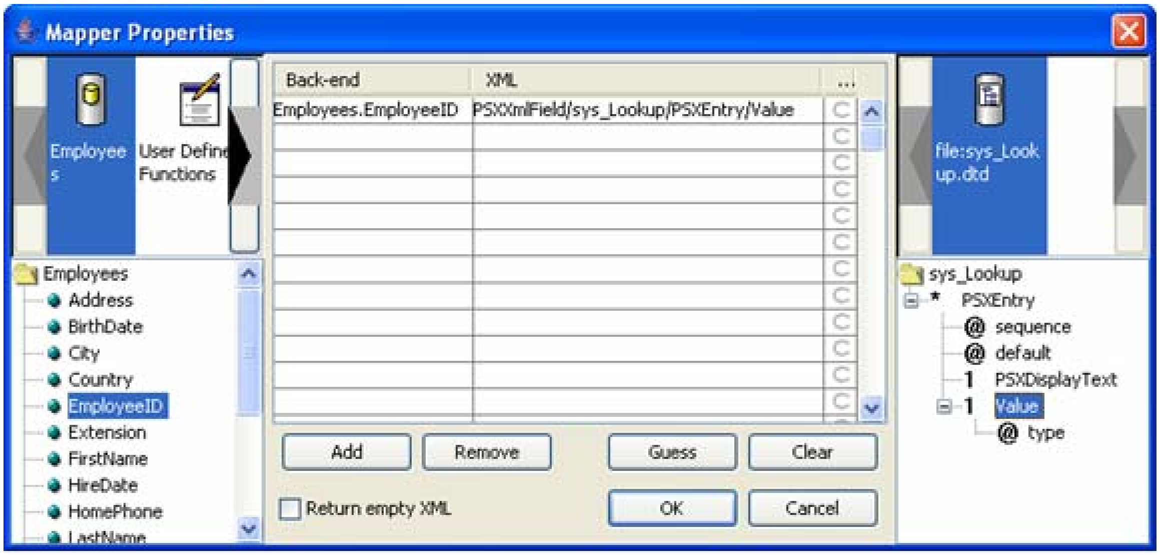


Figure 196: Mapping Employees.EmployeeID to the Value node

12 Use the sys_Concat UDF to concatenate the Employee.LastName. Employee.FirstName, and HomePhone columns with appropriate text, then map the result to the PSXDisplayText Element.

a) On the left side of the Mapper Properties dialog, choose *User Defined Functions*.

b) Expand the Global Folder, then the Generic Subfolder. In the Generic Subfolder, click *sys_Concat* and drag it to the first empty row of the Backend Column.

The Rhythmyx Workbench displays the Function Properties dialog.

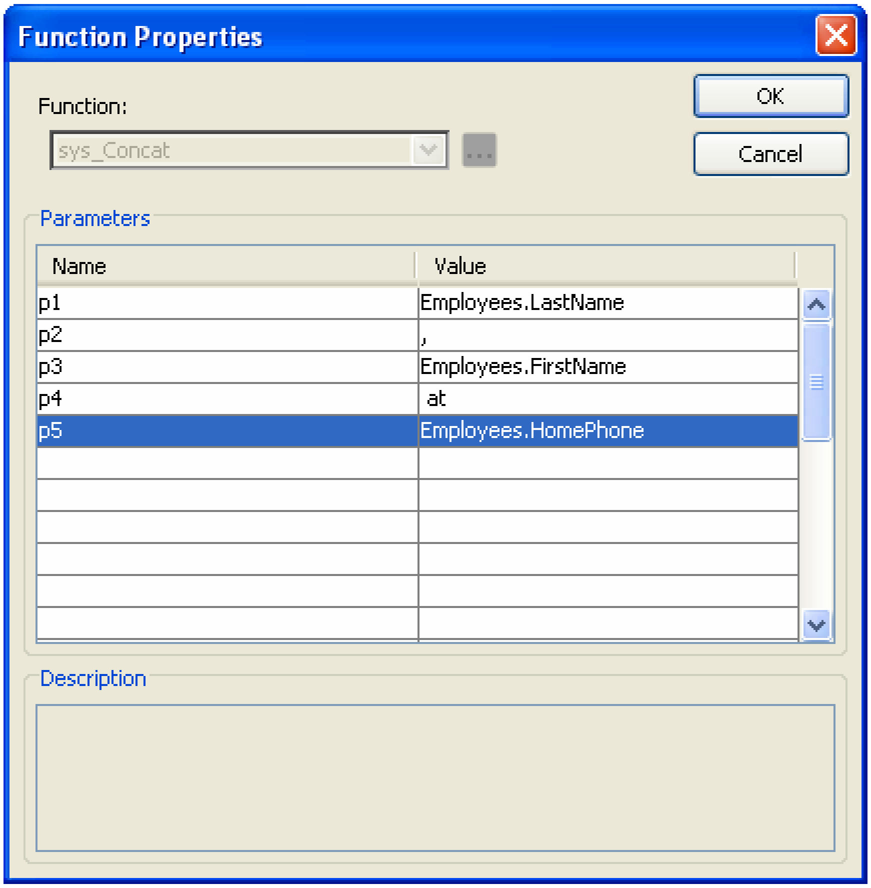
c) Click in the Value column of the p1 row and use the Value Selector to add the Backend Column *Employees.LastName*.

d) Click in the Value column of the p2 row and use the Value Selector to add the Literal value ", " (comma<space>).

e) Click in the Value column of the p3 row and use the Value Selector to add the Backend Column *Employees.FirstName*.

f) Click in the Value column of the p4 row and use the Value Selector to add the Literal value " at " (<space>at<space>).

g) Click in the Value column of the p5 row and use the Value Selector to add the Backend Column *Employees.HomePhone*.



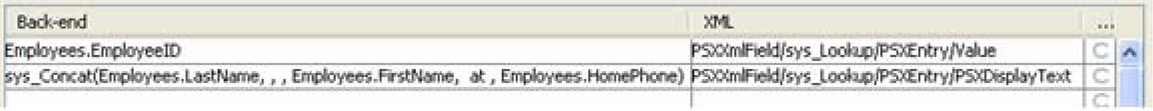
Function: sys_Concat

Name	Value
p1	Employees.LastName
p2	,
p3	Employees.FirstName
p4	at
p5	Employees.HomePhone

Figure 197: Lookup UDF definition

h) Click the [**OK**] button to save the UDF definition.

i) On the right side of the Mapper Properties dialog, click on the PSXDisplayText Element and drag it to the same row as the sys_Concat UDF.



Back-end	XML
Employees.EmployeeID	PSXXmlField/sys_Lookup/PSXEntry/Value
sys_Concat(Employees.LastName, , , Employees.FirstName, at , Employees.HomePhone)	PSXXmlField/sys_Lookup/PSXEntry/PSXDisplayText

Figure 198: Completed lookup mappings. Only the mappings are illustrated.

j) Click the [**OK**] button to save the mappings.

13 Close the Data Pipe tab at the bottom of the editor.

14 Click the save button in the Rhythmyx Workbench to save the application.

15 Click the green arrow to start the application.

Once the application has started, it is loaded into memory and can respond to requests. It is a good idea to test the application at this point to ensure that it is returning the results you expect. To test the application:

1 Open the Rhythmyx application (if it is not already open).

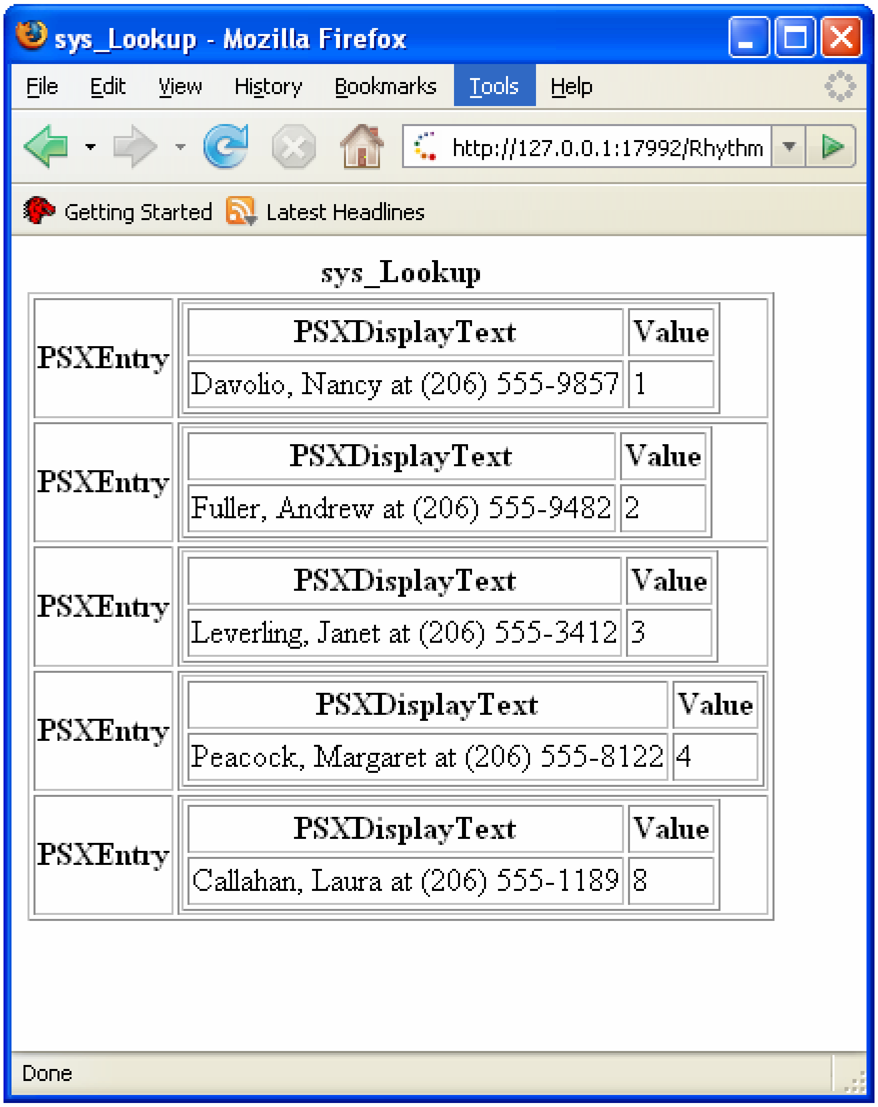
2 Right-click on the rffEmployNameAndNumber query and from the popup menu, choose *Request Properties*.

The Rhythmyx Workbench displays the Request Properties dialog.

3 Click the [**Copy to clipboard**] button.

4 Open a browser window or tab and past the copied URL into the Address or URL field. Press the <Enter> key. Log in to Rhythmyx using your standard credentials.

The browser displays the output document with default formatting.

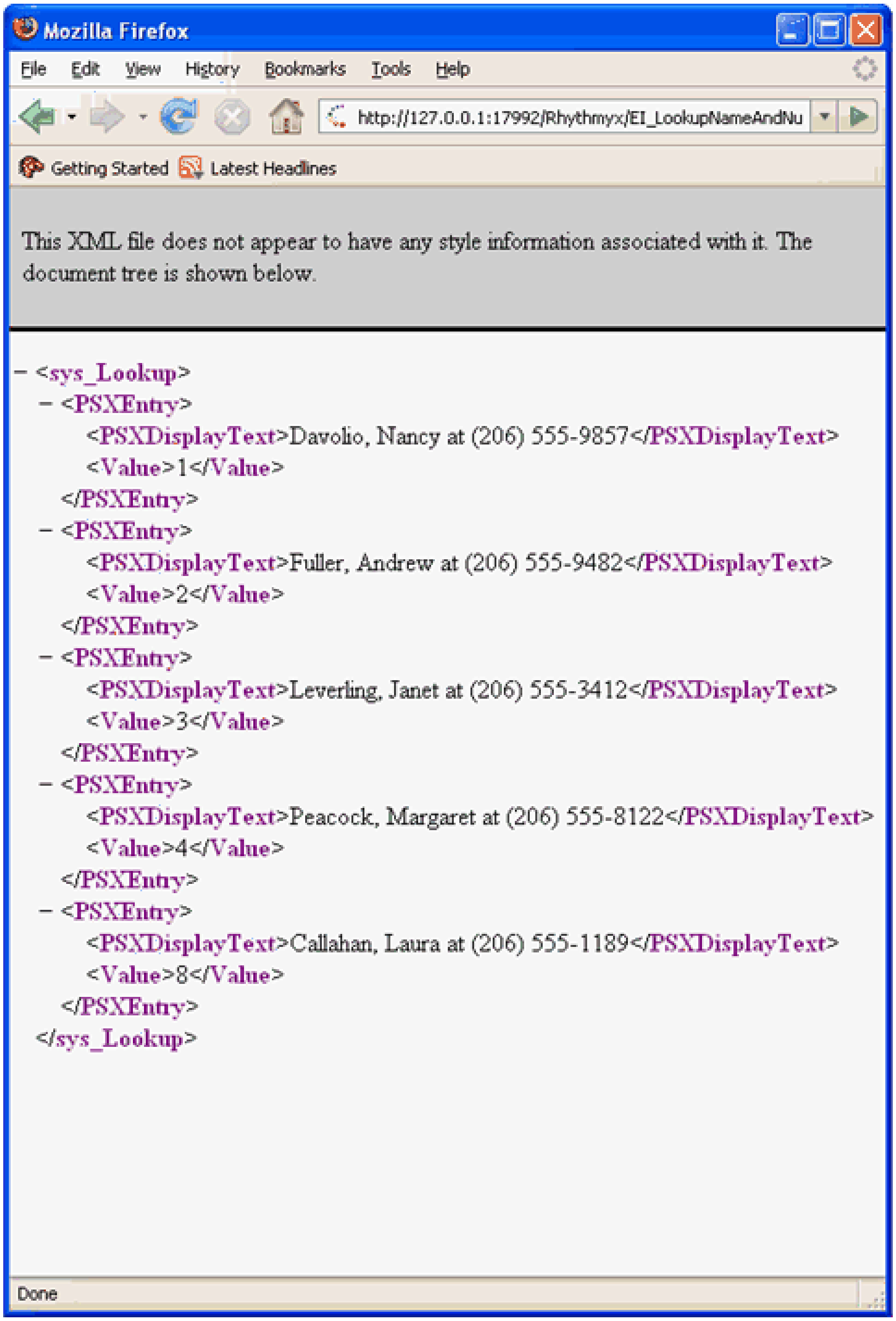


sys_Lookup

PSXEntry	PSXDisplayText Davolio, Nancy at (206) 555-9857	Value 1
PSXEntry	PSXDisplayText Fuller, Andrew at (206) 555-9482	Value 2
PSXEntry	PSXDisplayText Leverling, Janet at (206) 555-3412	Value 3
PSXEntry	PSXDisplayText Peacock, Margaret at (206) 555-8122	Value 4
PSXEntry	PSXDisplayText Callahan, Laura at (206) 555-1189	Value 8

Figure 199: Lookup output formatted as HTML

Change the extension in the address field from *html* to *xml* to see the XML document.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1:17992/Rhythmyx/EI_LookupNameAndNu
Getting Started Latest Headlines
Done

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <sys_Lookup>
  - <PSXEntry>
      <PSXDisplayText>Davolio, Nancy at (206) 555-9857</PSXDisplayText>
      <Value>1</Value>
    </PSXEntry>
  - <PSXEntry>
      <PSXDisplayText>Fuller, Andrew at (206) 555-9482</PSXDisplayText>
      <Value>2</Value>
    </PSXEntry>
  - <PSXEntry>
      <PSXDisplayText>Leverling, Janet at (206) 555-3412</PSXDisplayText>
      <Value>3</Value>
    </PSXEntry>
  - <PSXEntry>
      <PSXDisplayText>Peacock, Margaret at (206) 555-8122</PSXDisplayText>
      <Value>4</Value>
    </PSXEntry>
  - <PSXEntry>
      <PSXDisplayText>Callahan, Laura at (206) 555-1189</PSXDisplayText>
      <Value>8</Value>
    </PSXEntry>
  </sys_Lookup>
```

Figure 200: Lookup output formatted as XML

Populating a Control with Results from an Application

Now that we have a lookup application to return data from the external repository, we can use that data to populate the list control. We will modify the event_contact field in the Event Content Type to use the sys_DropDownSingle control, with the choices populated by the lookup application we created.

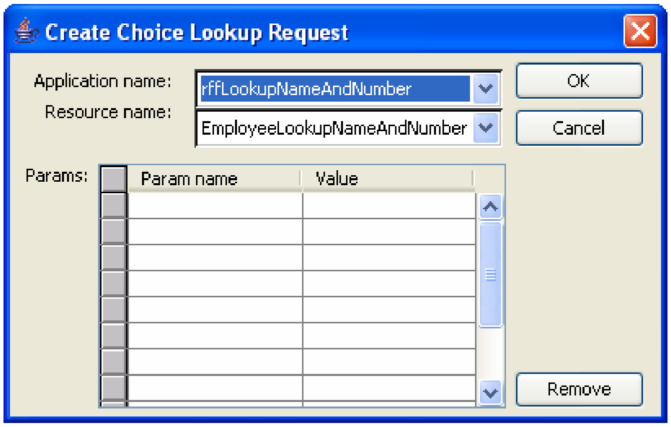
NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

1. In the Rhythmyx Workbench, open the rffEvent Content Type.
2. Click on the tab for the event_location child editor.
3. Select the event_contact field and click the [**All Properties**] button.

   The Rhythmyx Workbench displays the Field Properties dialog.
4. In the Control field, select *sys_DropDownSingle*.
5. Click the browse button ... next to the Control field.

   The Rhythmyx Workbench displays the Control Properties dialog.
6. Click on the Choices tab.
7. Click the Retrieve from xml application radio button.
8. Click the browse button next to the associated field.

   The Rhythmyx Workbench displays the Create Choice Lookup Request dialog.
9. In the Application Name drop list, choose *rffLookupNameAndNumber*.
10. In the Resource drop list, choose *EmployeeLookupNameAndNumber*.



Create Choice Lookup Request

Application name: rffLookupNameAndNumber

Resource name: EmployeeLookupNameAndNumber

Params:

Param name	Value

Figure 201: Specifying a lookup application for a list control

11. Click the [**OK**] button to save the request definition.
12. On the Control Properties dialog, click the [**OK**] button. On the Field Properties dialog, click the [**OK**] button.
13. Save the rffEvent Content Type.

To confirm your changes, create a test Content Item and verify that the drop list includes the expected values.

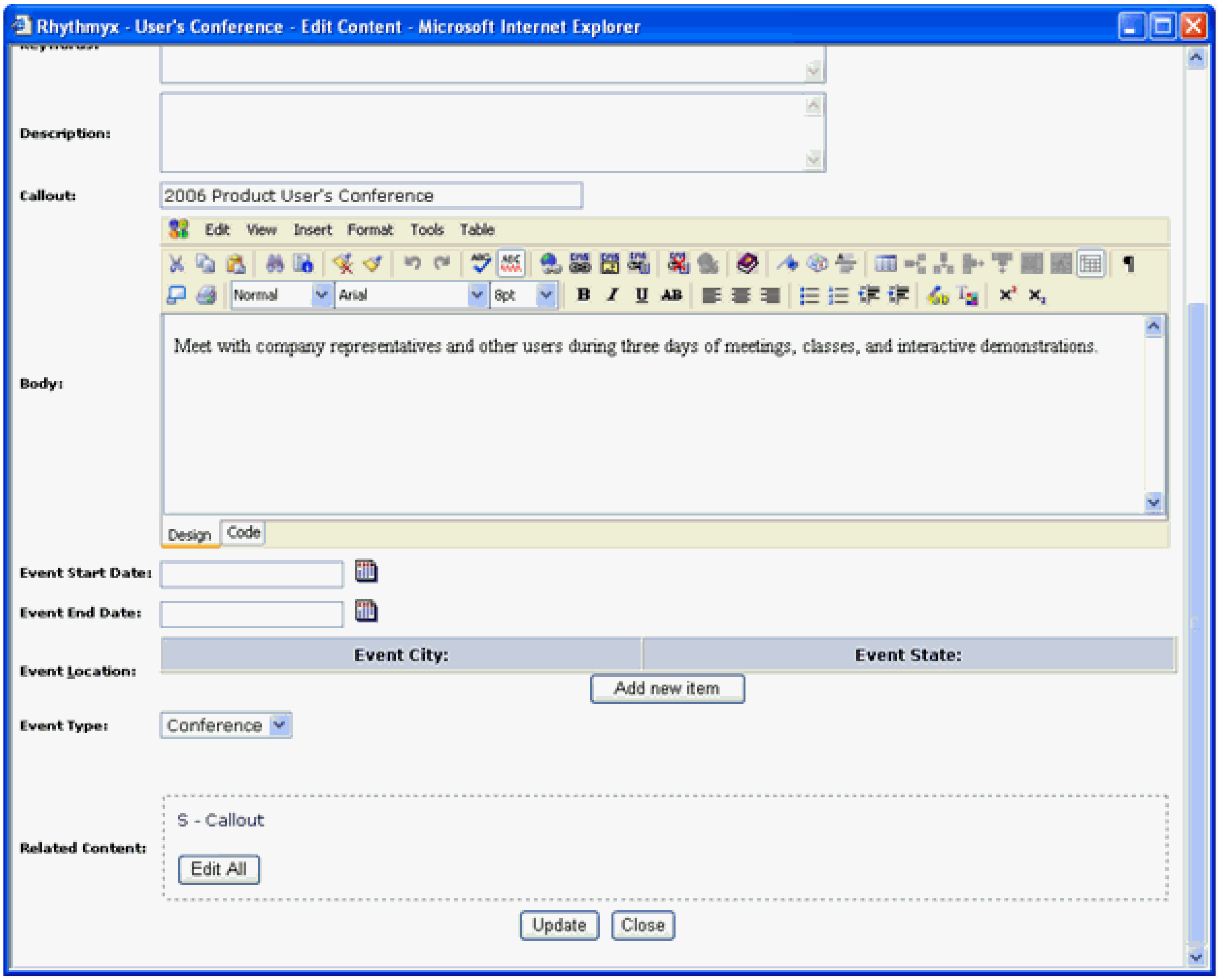
The Event Content Editor

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.



Rhythmyx - User's Conference - Edit Content - Microsoft Internet Explorer

Description:

Callout: 2006 Product User's Conference

Body: Meet with company representatives and other users during three days of meetings, classes, and interactive demonstrations.

Event Start Date:

Event End Date:

Event Location:

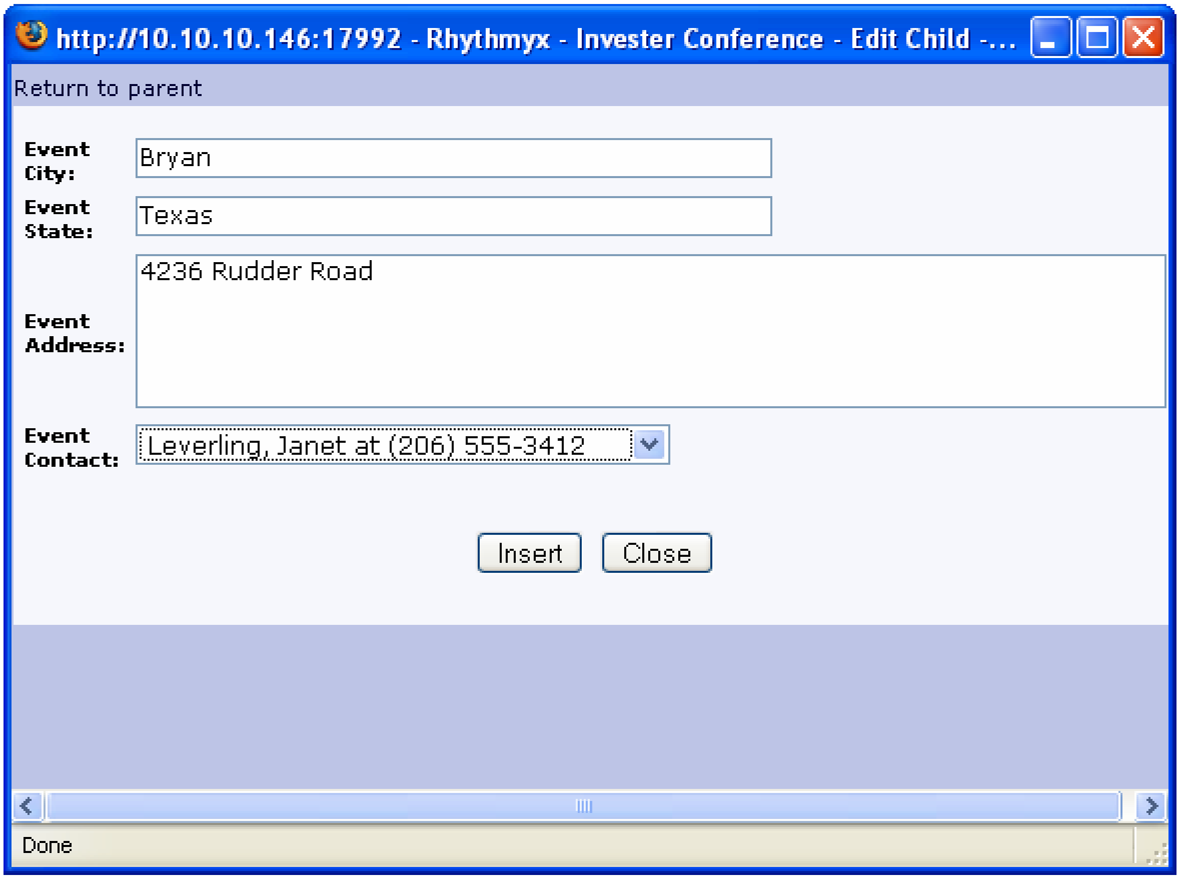
Event City:	Event State:

Event Type: Conference

Related Content: S - Callout

Figure 202: Event_Test Content Editor

The user clicks [**Add new item**] to view the entire table for *Event Location*:



http://10.10.10.146:17992 - Rhythmyx - Invester Conference - Edit Child -...
Return to parent
Insert
Close
Done

Event City: Bryan

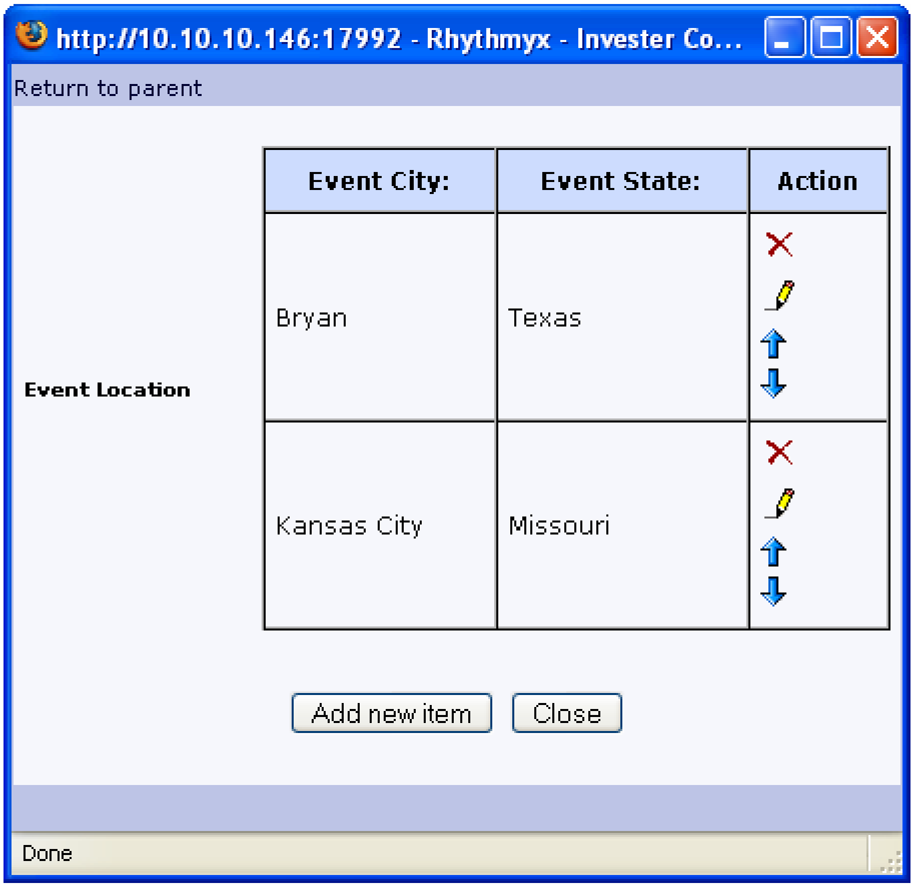
Event State: Texas

Event Address: 4236 Rudder Road

Event Contact: Leverling, Janet at (206) 555-3412

Figure 203: Child Table editor

After the user inserts one new location, the Content Editor provides a page for adding and editing additional ones:



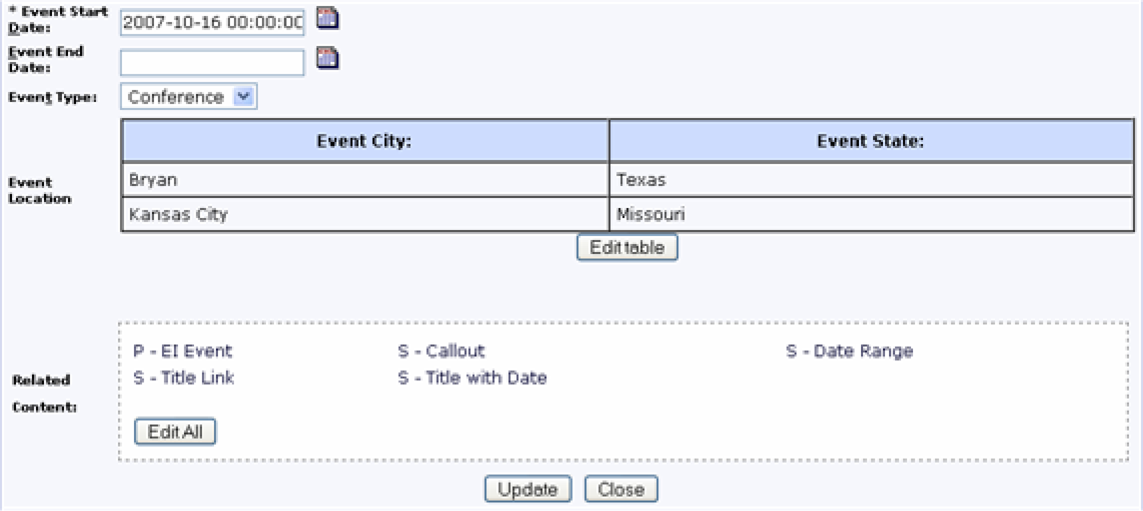
Event Location

Event City:	Event State:	Action
Bryan	Texas	✕ ✎ ↑ ↓
Kansas City	Missouri	✕ ✎ ↑ ↓

Figure 204: 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:



* Event Start Date: 2007-10-16 00:00:0C

Event End Date:

Event Type: Conference

Event Location

Event City:	Event State:
Bryan	Texas
Kansas City	Missouri

Related Content:

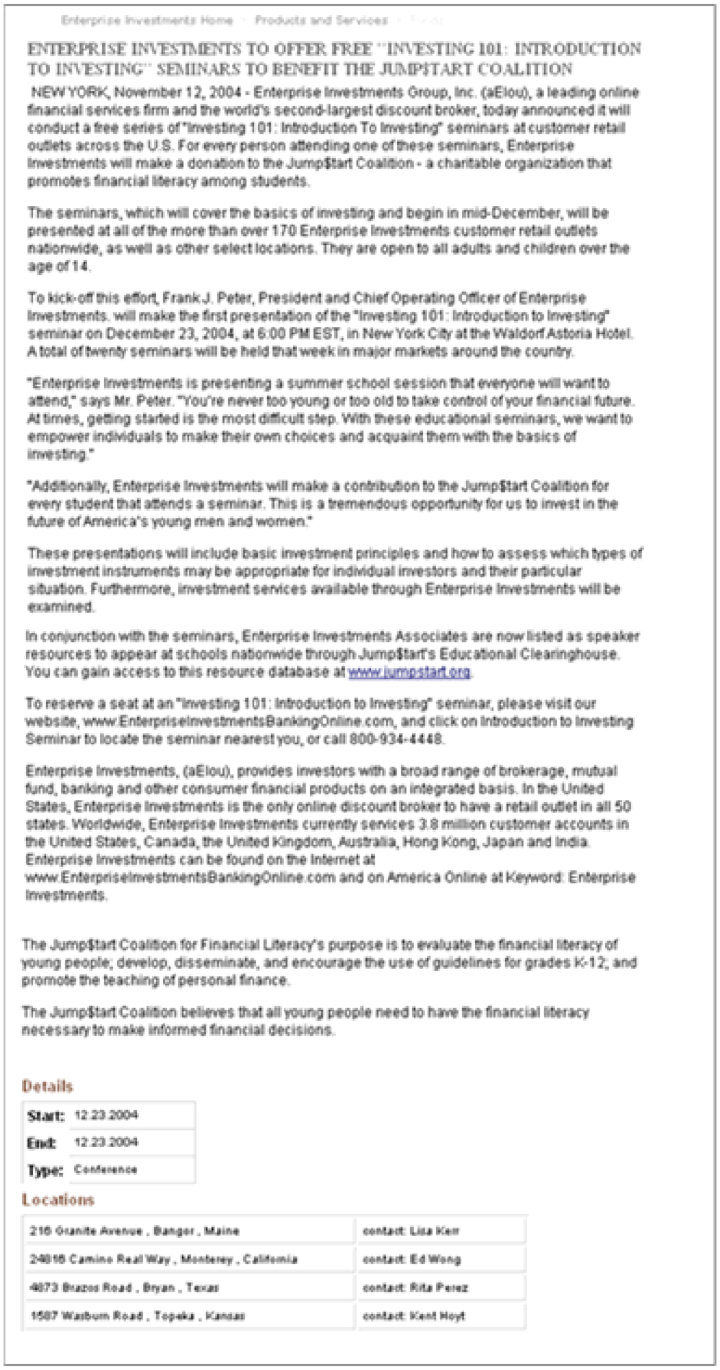
P - EI Event
S - Title Link
S - Callout
S - Title with Date
S - Date Range

Figure 205: 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 164) 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 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 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.

To reserve a seat at an "Investing 101: Introduction to Investing" seminar, please visit our website, 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 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

Start:	12.23.2004
End:	12.23.2004
Type:	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 206: 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 (excluding 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*.

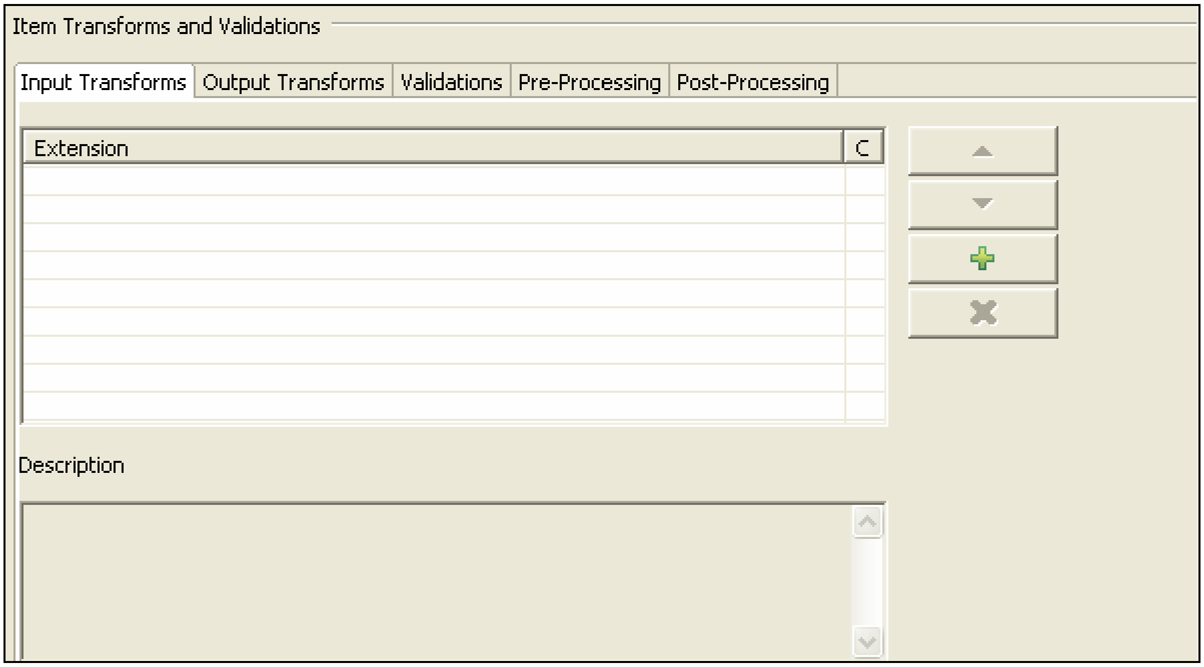


Figure 207: Item Transforms and Validations section

Adding Pre-processing Extensions (see below), 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.

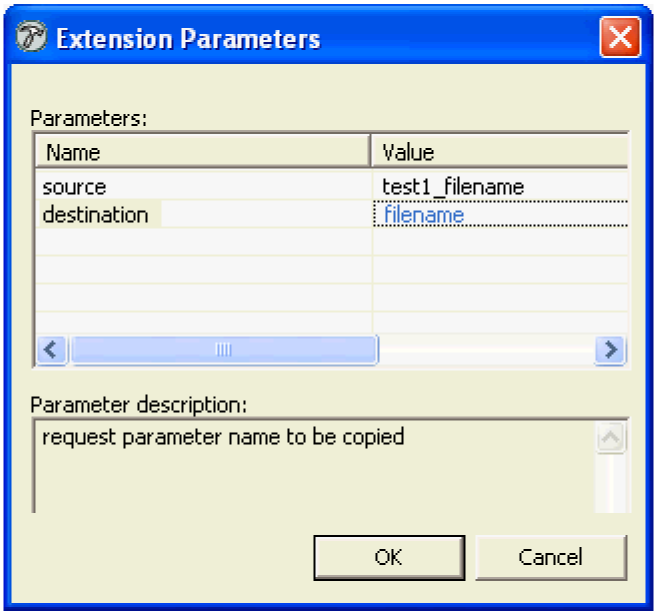
sys_CopyParameter copies values into fields that are used in FastForward's default location scheme. Location Schemes are discussed in the chapter ***Configuring Publishing*** (see page 309).

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.



Extension Parameters

Parameters:

Name	Value
source	test1_filename
destination	filename

Parameter description:

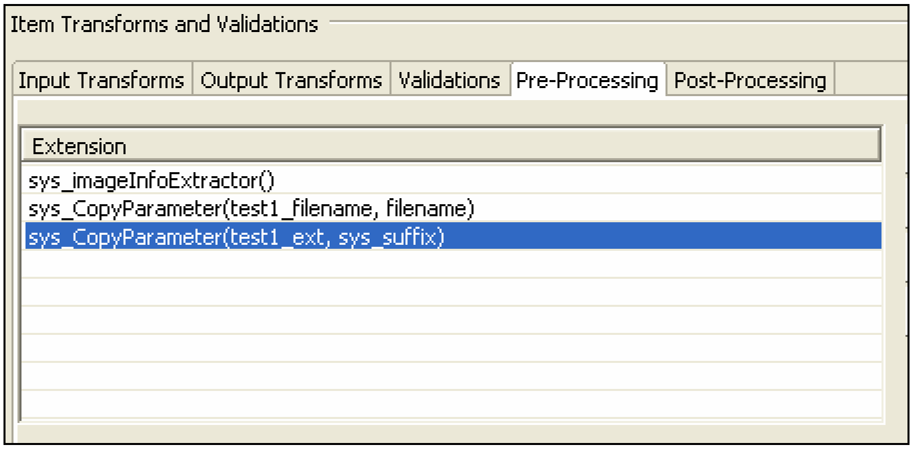
request parameter name to be copied

Figure 208: 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:



sys_imageInfoExtractor()
sys_CopyParameter(test1_filename, filename)
sys_CopyParameter(test1_ext, sys_suffix)

Figure 209: 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_textExtractor, extracts the text as simple text with no formatting 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. If you override the default text extractor for a file type in the Server Administrator's full-text search sub-tab, the new text extractor is also used with the sys_textExtractor exit. For additional information about overriding default text extractors in the full-text search engine, see the Rhythmyx Server Administrator online help.

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_textExtractor extension does not limit the size of text extracted from the 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

Source	Name	Label	Control Name	Occur	Data Type	Format
shared	sharedbinary/item_file_attachment_size	File Size:	sys_EditBox	optional	integer	none
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 239) and ***Image Content Type Creation*** (see page 239).

Once you define the fields, you must add the sys_textExtractor 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_textExtractor extension.

   Rhythmyx displays the Extension Parameters dialog with a list of parameters for the sys_textExtractor 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	Required. 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. 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.	PSXSingleHTMLParameter/item_file_attachment

Name	Description	Value
OutputParam	Required. The name of the Content Type field in which the extracted content will be stored. 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.	body
FileTypeParam	Optional. 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.	item_file_attachment_type
ErrorMessageParam	Optional. 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. 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.	extraction_error
OutputEncoding	Optional. Specifies the character encoding to use for the text output. If no value is specified for this parameter, the text is output in WINDOWS-1252. The following encodings are also valid: ▪ Shift_JIS (Japanese) ▪ EUC_KR (Korean) ▪ GB2312 (Simple Chinese) ▪ Big5 (Traditional Chinese) If the input text includes characters outside of the specified character set, that text may be lost.	

5 Click the [**OK**] button to save the parameter specifications.

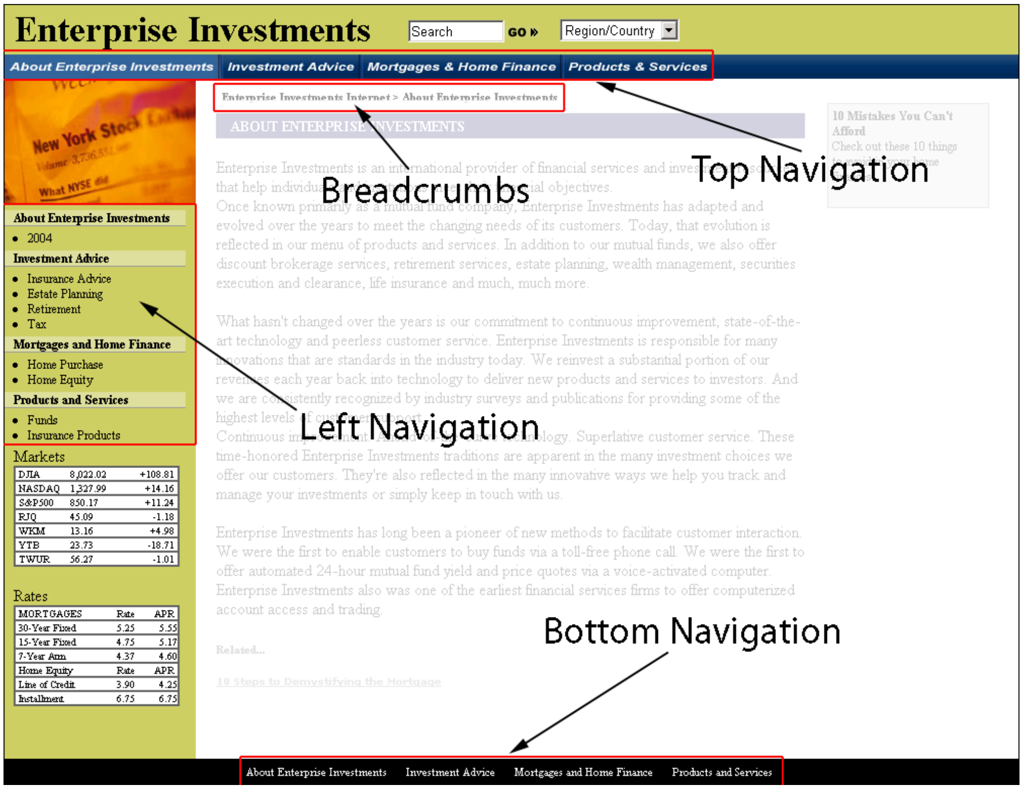
6 Save the Content Editor.

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



ABOUT ENTERPRISE INVESTMENTS

Enterprise Investments is an international provider of financial services and inves[illegible] that help individua[illegible]ial objectives.
Once known primarily as a mutual fund company, Enterprise Investments has adapted and evolved over the years to meet the changing needs of its customers. Today, that evolution is reflected in our menu of products and services. In addition to our mutual funds, we also offer discount brokerage services, retirement services, estate planning, wealth management, securities execution and clearance, life insurance and much, much more.

What hasn't changed over the years is our commitment to continuous improvement, state-of-the-art technology and peerless customer service. Enterprise Investments is responsible for many innovations that are standards in the industry today. We reinvest a substantial portion of our revenues each year back into technology to deliver new products and services to investors. And we are consistently recognized by industry surveys and publications for providing some of the highest levels of customer support.
Continuous improvement. Ahead-of-the-curve technology. Superlative customer service. These time-honored Enterprise Investments traditions are apparent in the many investment choices we offer our customers. They're also reflected in the many innovative ways we help you track and manage your investments or simply keep in touch with us.

Enterprise Investments has long been a pioneer of new methods to facilitate customer interaction. We were the first to enable customers to buy funds via a toll-free phone call. We were the first to offer automated 24-hour mutual fund yield and price quotes via a voice-activated computer. Enterprise Investments also was one of the earliest financial services firms to offer computerized account access and trading.

Related...

10 Steps to Demystifying the Mortgage

10 Mistakes You Can't Afford
Check out these 10 things [illegible]

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 210: 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.

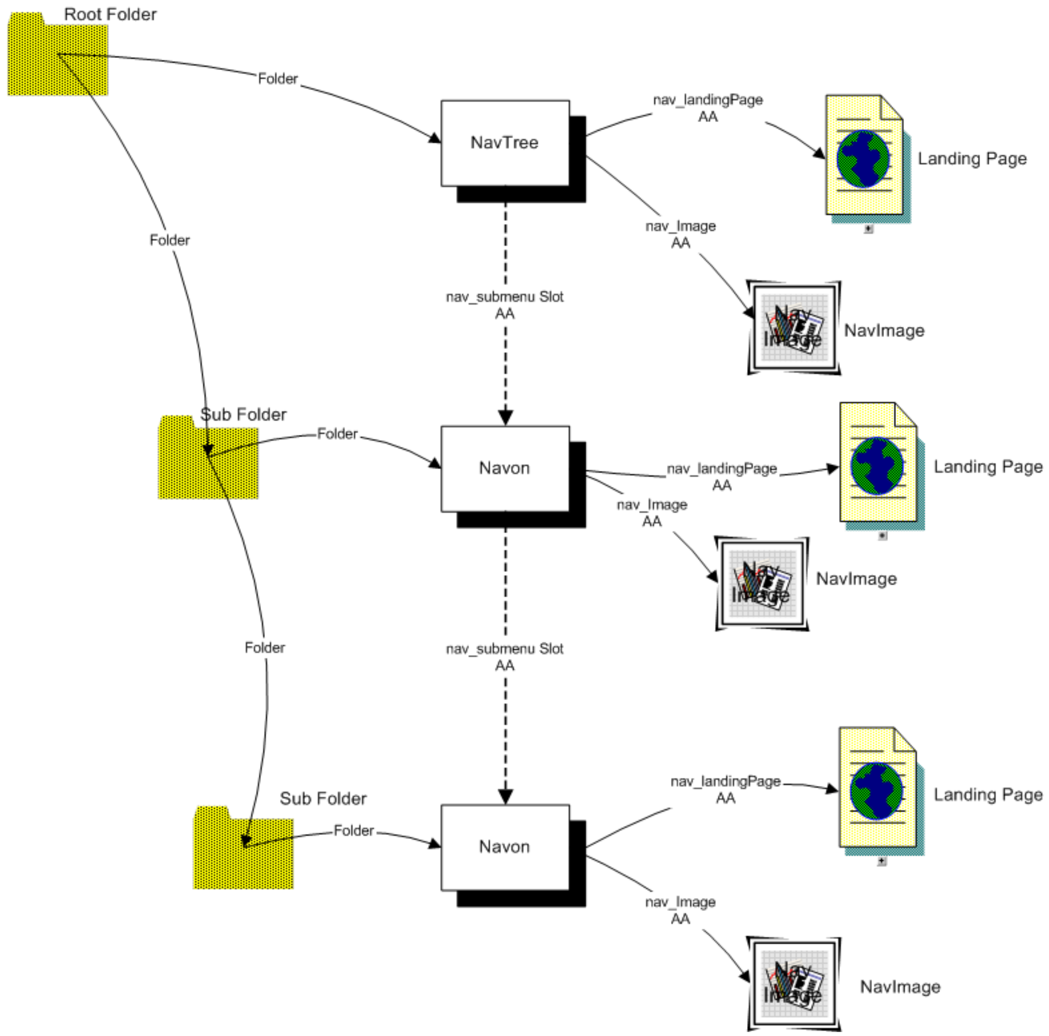


Figure 211: Relationships between Navigational Elements

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:

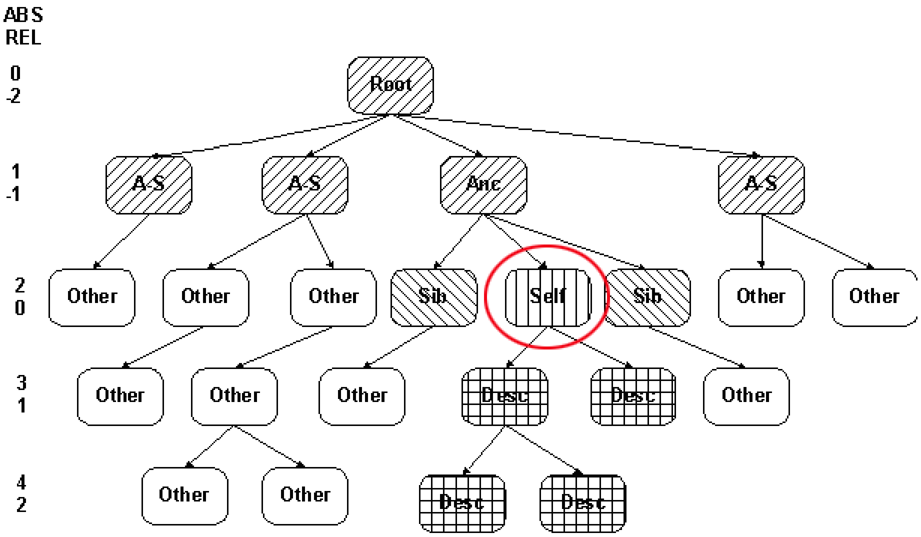


Figure 212: Navigation Relationships

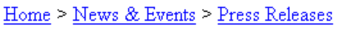
The node where the assembly of the Item starts is the SELF node. All predecessors of the SELF node in the path to the root are known as ANCESTOR nodes. (The immediate predecessor in this path is referred to as the PARENT) Other nodes that share the same PARENT as the SELF node are SIBLING nodes. Nodes 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 213: Breadcrumbs Without Formatting

When integrated into a Page, this Template inherits the look and feel of the page that includes it:

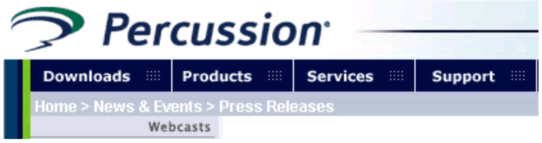


Figure 214: Page with Breadcrumbs

Maintaining Managed Navigation Content Items

In the chapter ***Setting Up the Publishing Site and Basic Navigation*** (on page 67), 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.

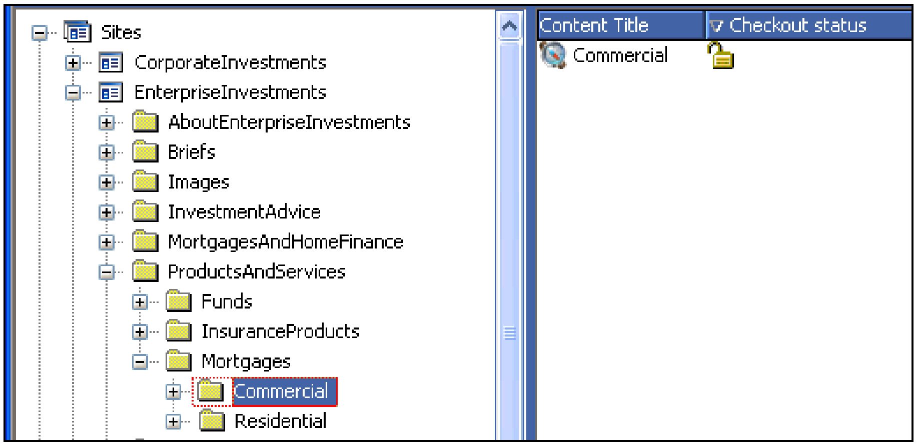
6 Close the Active Assembly Table Editor

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.

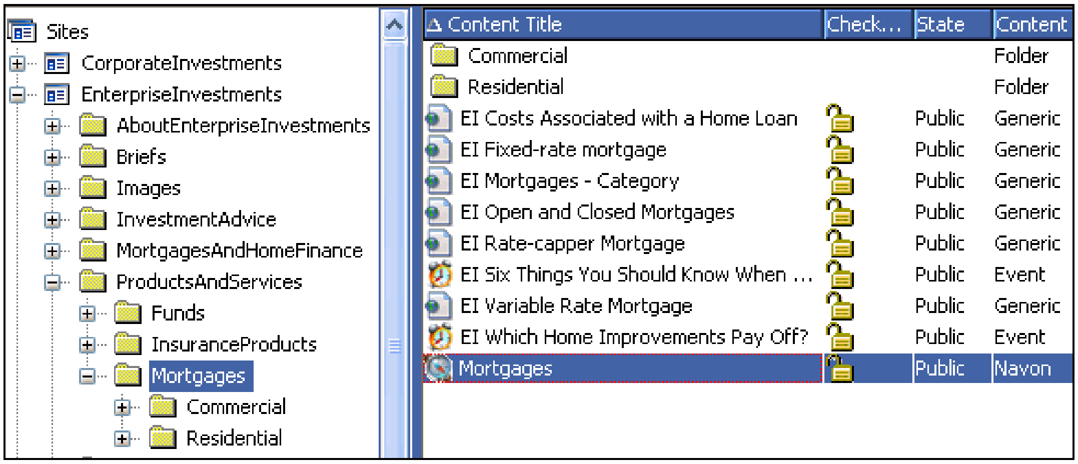
In the following graphic, an administrator has split the Mortgages folder into two subfolders, Commercial and Residential.



Content Title	Checkout status
Commercial	🔓

Figure 215: Splitting navigation sections

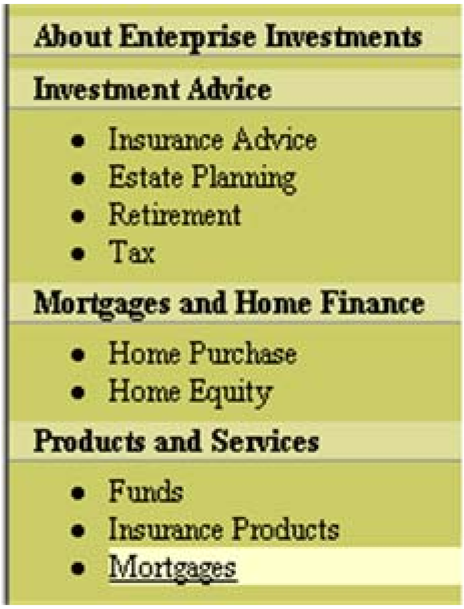
If you leave the Navon in the parent folder, Mortgages, and remove the Navons from the subfolders Commercial and Residential, only a link to item that the Mortgages Navon appears in the Navigation sections of the Web Site. In the following graphic, the Navon is left in the Mortgages folder.



Content Title	Check...	State	Content
Commercial			Folder
Residential			Folder
EI Costs Associated with a Home Loan	🔓	Public	Generic
EI Fixed-rate mortgage	🔓	Public	Generic
EI Mortgages - Category	🔓	Public	Generic
EI Open and Closed Mortgages	🔓	Public	Generic
EI Rate-capper Mortgage	🔓	Public	Generic
EI Six Things You Should Know When ...	🔓	Public	Event
EI Variable Rate Mortgage	🔓	Public	Generic
EI Which Home Improvements Pay Off?	🔓	Public	Event
Mortgages	🔓	Public	Navon

Figure 216: Subdivided folders with Navon in parent folder

One of the navigation sections of Web site appears as follows. There is a link to a Mortgages page, but no links to Commercial Mortgages or Residential Mortgages pages.



About Enterprise Investments

Investment Advice

- Insurance Advice
- Estate Planning
- Retirement
- Tax

Mortgages and Home Finance

- Home Purchase
- Home Equity

Products and Services

- Funds
- Insurance Products
- Mortgages

Figure 217: Navigation without Split Sections

Another option is to add the two sub-folders and remove the Navon from the Mortgages parent folder, but leave the Navons in the Commercial Mortgages and Residential Mortgages sub-folders so that the navigation sections of your Site can link to a page in the subfolders.

To split a Site Folder and include navigation links to the subfolders only:

1. Log into Content Explorer under and administrative Community 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 create two sub-folders, Commercial Mortgages and Residential Mortgages. When we create these Folders, a Navon is added to each of them automatically.
3. Delete the Navon Item from the parent Folder (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 parent Folder to the new descendant Folders.
5. Edit the Navon in each sub-folder. At the bottom of the page, click [**Edit All**] and assign landing pages (pages in the sub-folder) to the Nav Landing Page Slot of each new Navon.
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 Slot of the Navon in the parent Folder of the Mortgages Folder (Products and Services in our example).

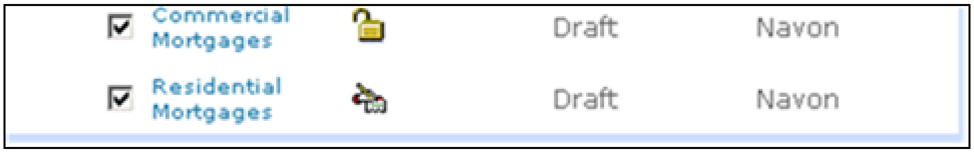
7 Open the Products and Services Folder and locate its Navon.

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.



☑	Commercial Mortgages		Draft	Navon
☑	Residential Mortgages		Draft	Navon

Figure 218: 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.

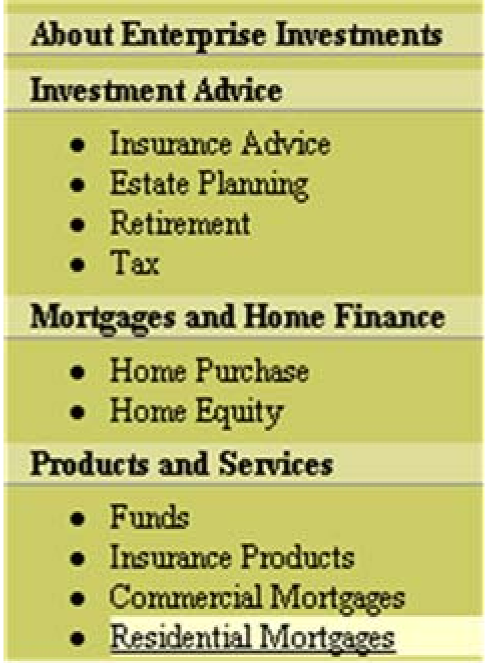


Figure 219: Navigation with New Split Sections

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:

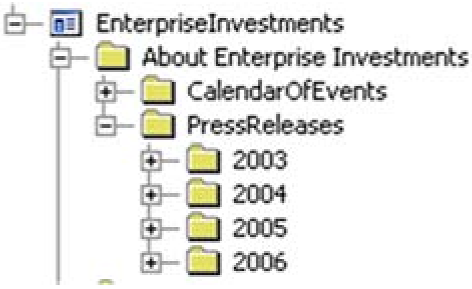


Figure 220: Press Releases by Year

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.

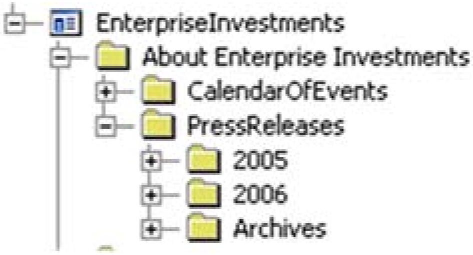


Figure 221: Press Releases by Year with Archive Folder

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.

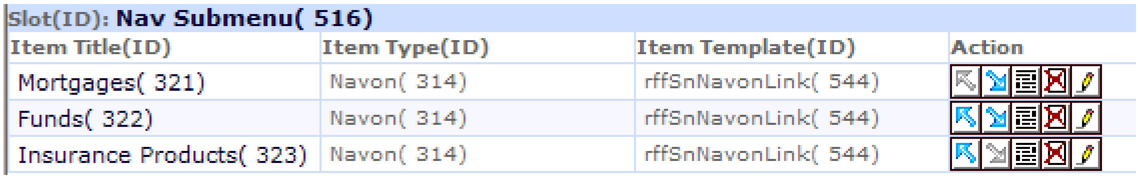
Reordering a Submenu

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(516)**

Item Title(ID)	Item Type(ID)	Item Template(ID)	Action
Mortgages(321)	Navon(314)	rffSnNavonLink(544)	↖ ↘ ☰ ☒ ✎
Funds(322)	Navon(314)	rffSnNavonLink(544)	↖ ↘ ☰ ☒ ✎
Insurance Products(323)	Navon(314)	rffSnNavonLink(544)	↖ ↘ ☰ ☒ ✎

Figure 222: Active Assembly Table Editor

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.

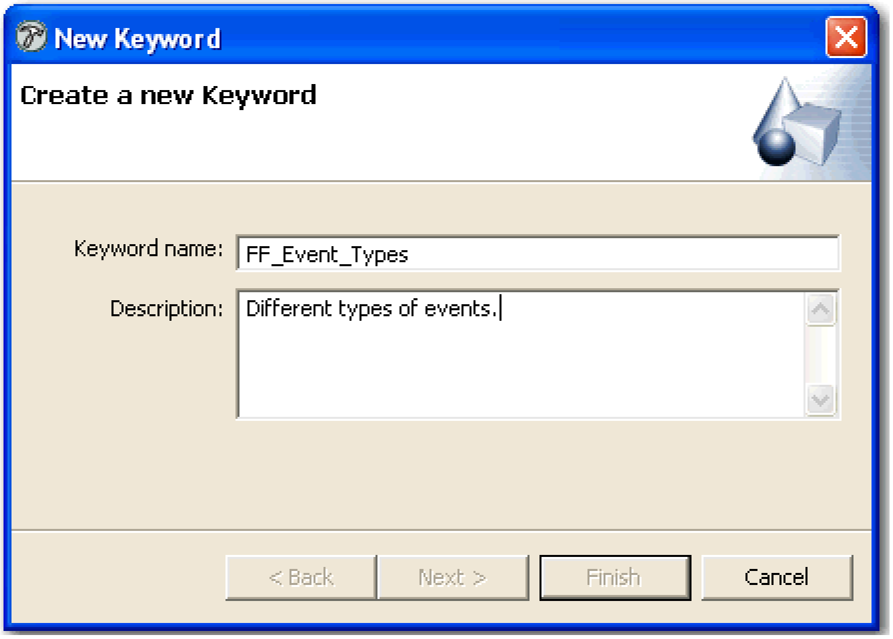
FastForward provides the *FF_Event_Types* Keyword with several choices for populating the sys_DropDownSingle control used with the Event Content Type's *event_type* field. Here we will demonstrate how you can create your own version of the *FF_Event_Types* Keyword and Keyword Choices.

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*.



Keyword name: FF_Event_Types

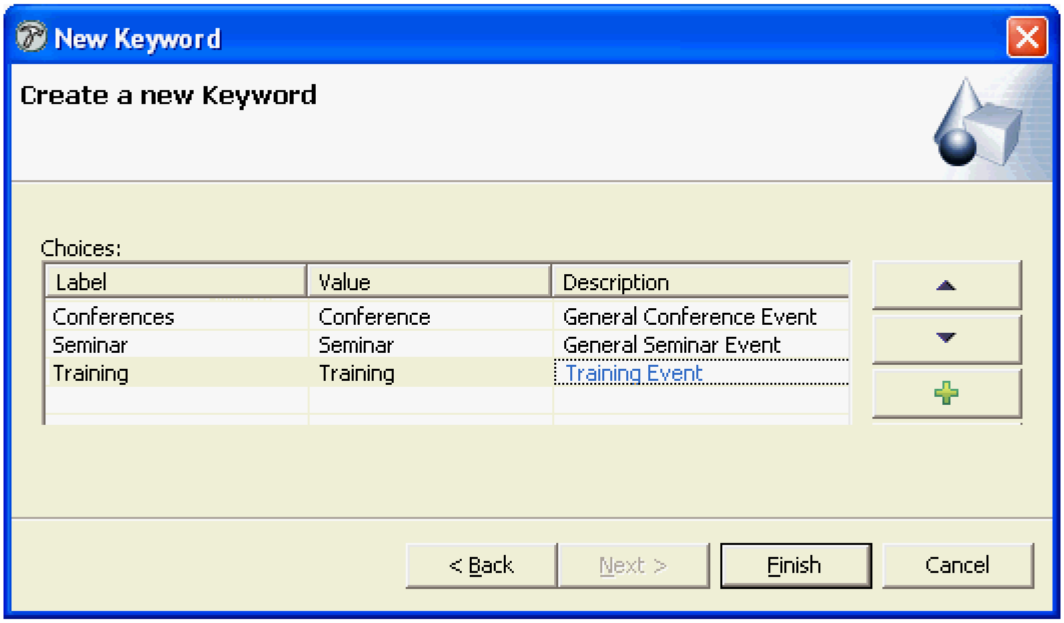
Description: Different types of events.

Figure 223: 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 224: 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.

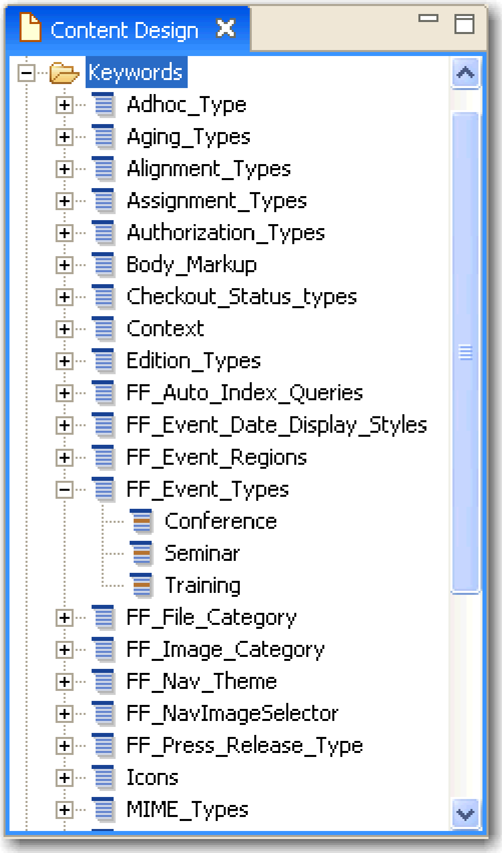


Figure 225: Keyword and choices added

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 252), 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:

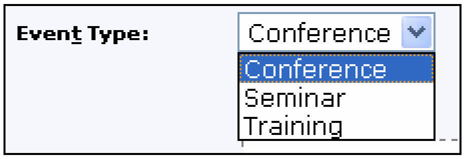
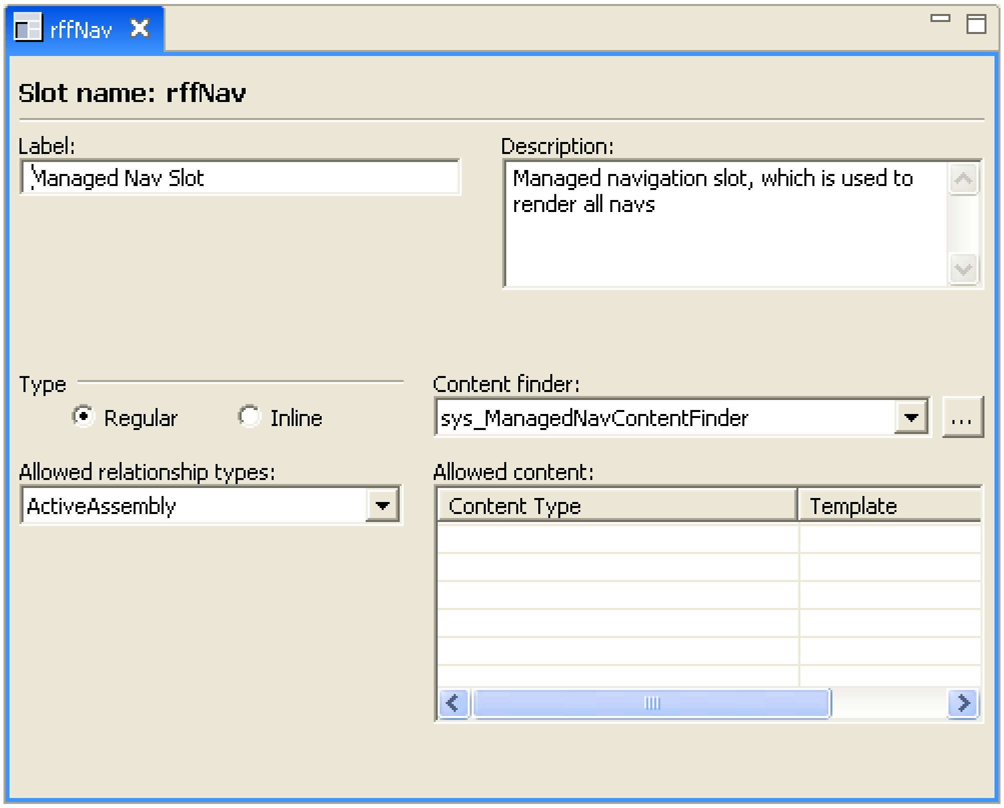

Event Type: Conference

Figure 226: 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 95).

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.



Slot name: rffNav

Label: Managed Nav Slot

Description: Managed navigation slot, which is used to render all navs

Type: ● Regular ○ Inline

Content finder: sys_ManagedNavContentFinder

Allowed relationship types: ActiveAssembly

Allowed content:

Content Type	Template

Figure 227: 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

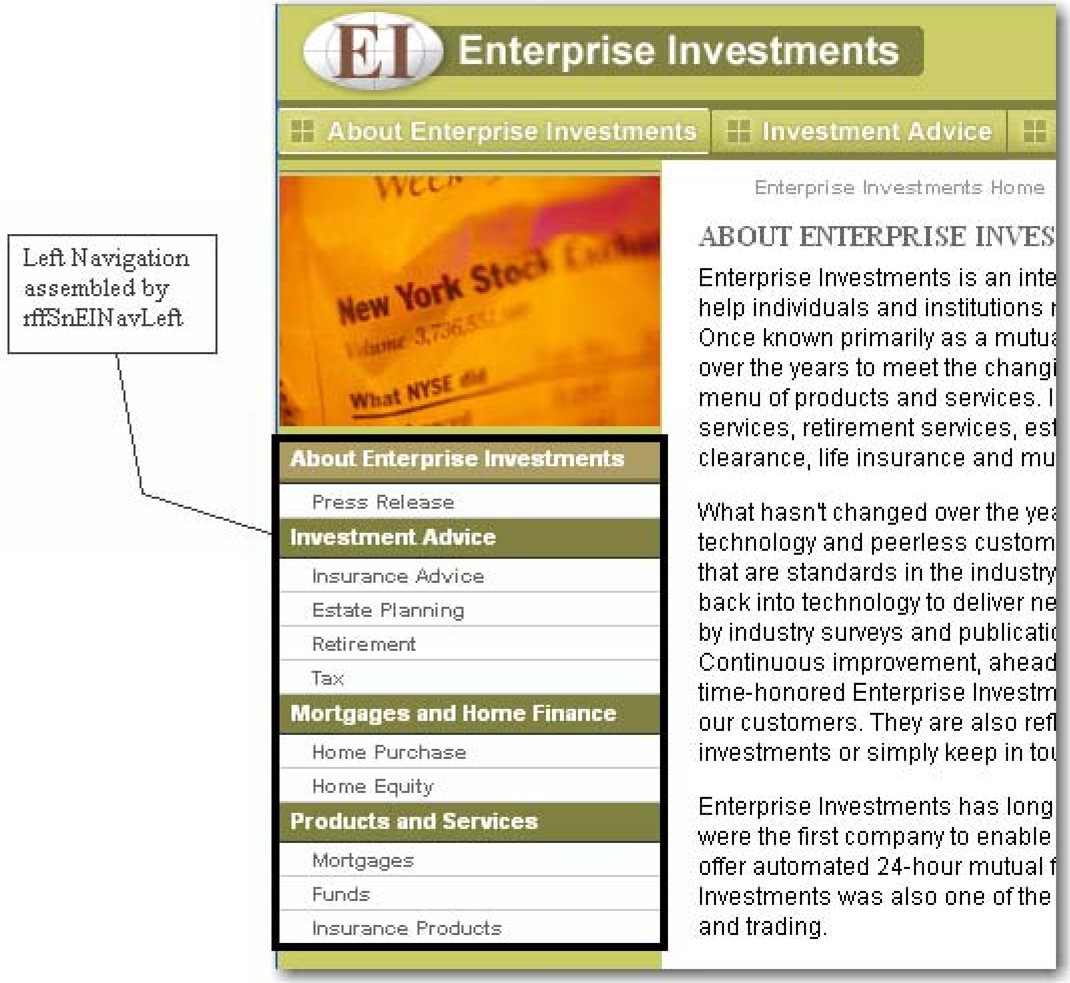
Managed Navigation Templates process the navigation tree XML to produce an HTML output. This output is merged with the Cascading Stylesheets to produce the final formatted output.

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 rffSnEINavLeft Template.



ABOUT ENTERPRISE INVES

Enterprise Investments is an inte
help individuals and institutions r
Once known primarily as a mutua
over the years to meet the changi
menu of products and services. I
services, retirement services, est
clearance, life insurance and mu

What hasn't changed over the yea
technology and peerless custom
that are standards in the industry
back into technology to deliver ne
by industry surveys and publicatio
Continuous improvement, ahead
time-honored Enterprise Investm
our customers. They are also refl
investments or simply keep in tou

Enterprise Investments has long
were the first company to enable
offer automated 24-hour mutual f
Investments was also one of the
and trading.

Figure 228: 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 306). 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 predominantly of four macros:

- #rootlevel
- #firstlevel
- #secondlevel
- #rendersectionimage

The body also includes some additional Velocity markup:

```
    #renderSectionImage($nav.self)
    #if ($imageurl)
       <div>
     <a href="$sectionlink">
          <img src="$imageurl" alt="$sectiontitle">
     </a>
    </div>
    #end
 #rootlevel($nav.root)
```

This code does three things:

1 Calls the #renderSectionImage macro with the self node.

```
#renderSectionImage($nav.self)
```

This macro generates the navigation image used at the top of the navigation bar.

2 Builds the image tag to display the navigation image.

```
#if ($imageurl)
   <div>
 <a href="$sectionlink">
      <img src="$imageurl" alt="$sectiontitle">
 </a>
</div>
#end
```

This code tests whether the $imageurl variable has a value. If it has no value, processing continues without building the image tag. If it does have a value, an image tag is generated with an anchor tag linking it to the associated landing page. See the discussion of the #renderSectionImage macro below for detail about how the values of the variables in this code are generated.

3 Calls the #rootlevel macro to begin building the navigation bar.

```
#rootlevel($nav.root)
```

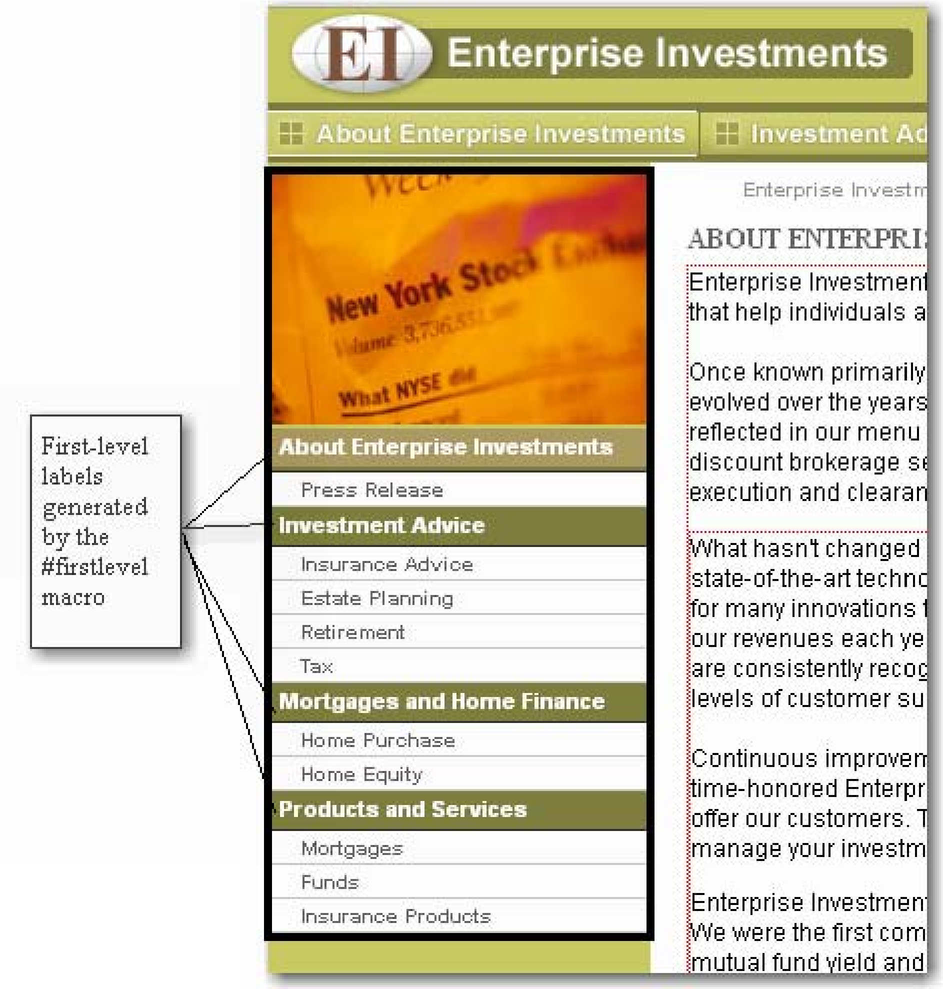
The #rootlevel macro builds the top image and builds the first level of the navigation bar. The macro consists of the following code:

```
 #macro(rootlevel $node)
   #set($submenu = $node.getNodes("nav:submenu"))
    #if ($node)
     #foreach ($navon in $submenu)
       #firstlevel ($navon)
```

```
      #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. Finally, it tests that the $node parameter passed into the macro contains a value. If it contains no value, processing terminates. If it does contain a value, the macro iterates over the set of Navon children of the navigation root and calls the `#firstlevel` macro for each one.

The #firstlevel macro defines the formatting of the first level of the navigation.



First-level labels generated by the #firstlevel macro

Figure 229: 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. For each Navon child, the #secondlevel macro is called.

The #secondlevel macro defines the formatting for the second level of the navigation.

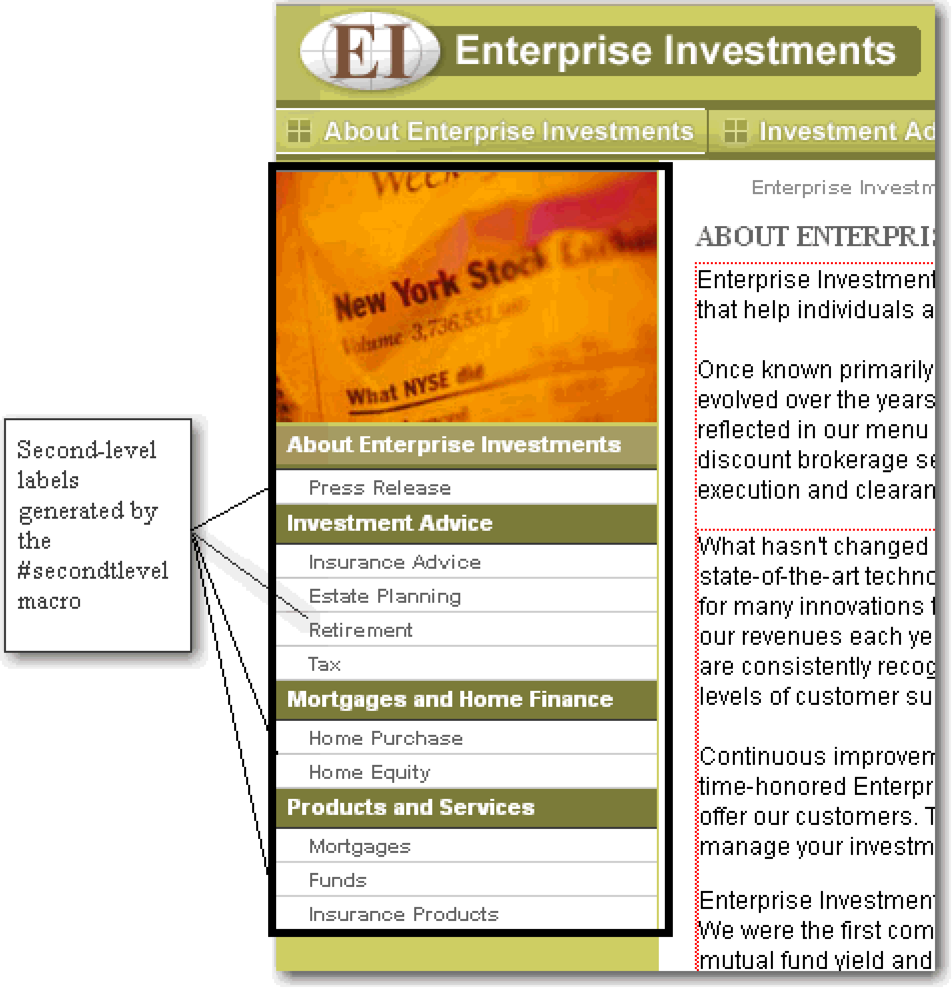


Figure 230: Left Navigation with second-level Navons highlighted

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())
  #if ( $landing_page )
  <li class="$indentclass">
    <a href="$landing_page">$title</a>
  </li>
  #end
#end
```

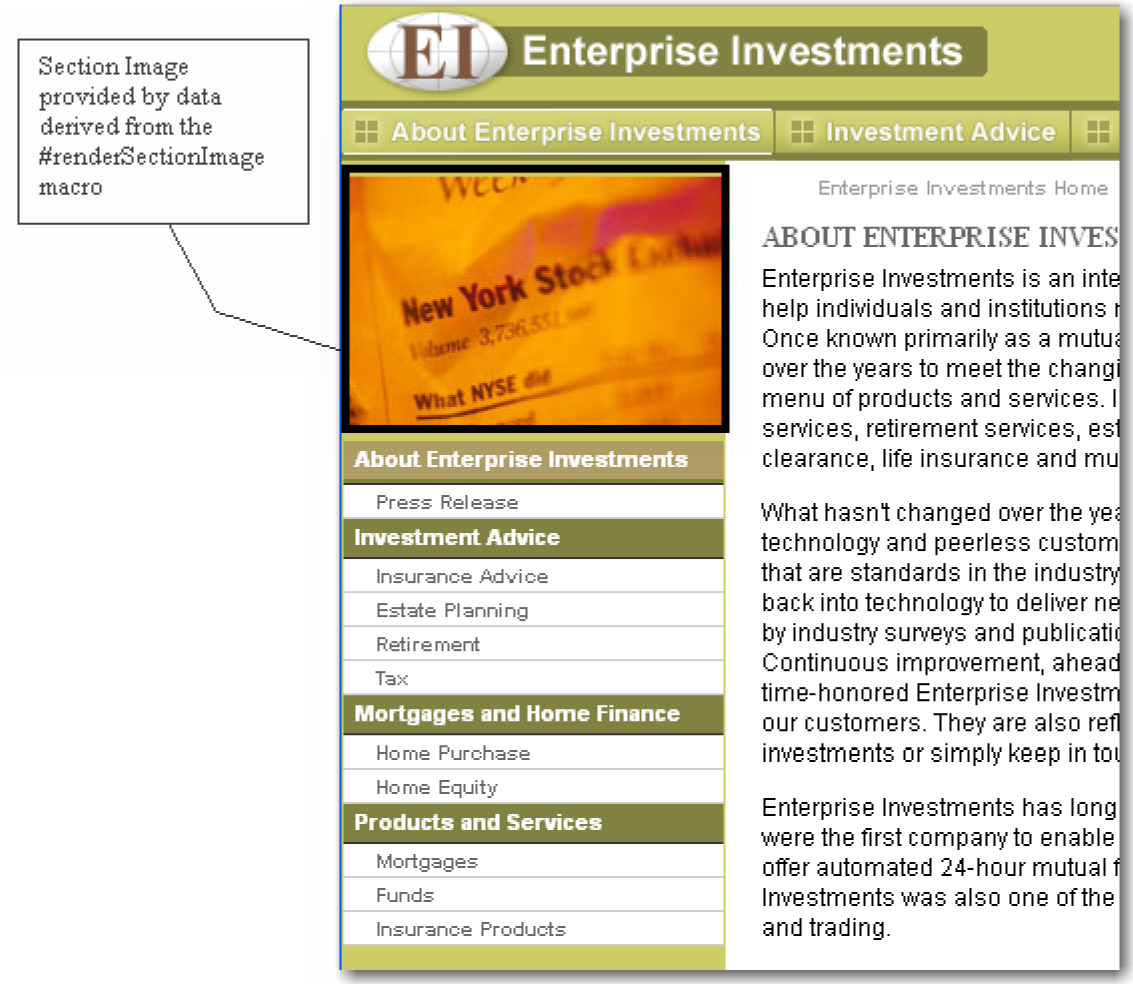
Like the other macros in this Template, this macro requires the $node parameter, which is passed from the #firstlevel macro when calling the #secondlevel macro. The macro beings by setting the same set of parameters as the #firstlevel macro.

- $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).

Having reviewed the other macros, let us now examine the `#renderSectionImage` macro, which provides the values for the variables used to build the section image tag that we examined earlier:



Section Image provided by data derived from the #renderSectionImage macro

ABOUT ENTERPRISE INVES

Enterprise Investments is an inte
help individuals and institutions r
Once known primarily as a mutu
over the years to meet the changi
menu of products and services. I
services, retirement services, est
clearance, life insurance and mu

What hasn't changed over the yea
technology and peerless custom
that are standards in the industry
back into technology to deliver ne
by industry surveys and publicati
Continuous improvement, ahead
time-honored Enterprise Investm
our customers. They are also ref
investments or simply keep in tou

Enterprise Investments has long
were the first company to enable
offer automated 24-hour mutual f
Investments was also one of the
and trading.

Figure 231: Left Navigation with Section Image highlighted

The code for this macro is:

```
#macro(renderSectionImage $navnode)
   #set($images = $navnode.getNodes("nav:image"))
```

```
      #foreach($image in $images)
         #if ($image.getProperty("selector").String == "section")
            #set($sectionImage = $image)
         #end
      #end
      #if (! $sectionImage)
         #if ($navnode.getParent())
            #renderSectionImage($navnode.getParent())
         #end
      #else
         #set($imageurl = "#imageurl($sectionImage 'active')")
         #set($sectionlink = $navnode.getProperty("nav:url").String)
         #set($sectiontitle =
   $navnode.getProperty("nav:landingPage").Node.getProperty("displaytitle")
   .String)
      #end
   #end
```

The macro starts by setting the value of the $images variable to set of navigation images in the Navon passed in when calling the macro. It then iterates over the set to find an image where the value of the Selector field is *section* and sets that image as the value of the value of the $sectionImage variable. If none of the images have the value *section* in the Selector field, the macro calls the parent Navon to find one. Once a value has been defined for $sectionImage, the macro sets the variables that are used to define the tags adding the navigation image:

- $imageurl is set to the URL of the active version of the image.
- $sectionlink is set to the URL of the Navon's landing page.
- $sectiontitle is set to the value of the Display Title field of the Navon.

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 297) produces the following HTML when previewed:

```
   <img
   src="http://10.10.10.100:9662/Rhythmyx/assembler/render?sys_revision=2&

      sys_siteid=301&sys_authtype=0&sys_contentid=482&sys_variantid=516&sys
   _folderid=482&sys_context=0"
      id="img_a482" class="sectionImage" alt="About Enterprise Investments"
   border="0"/>
         <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>
```

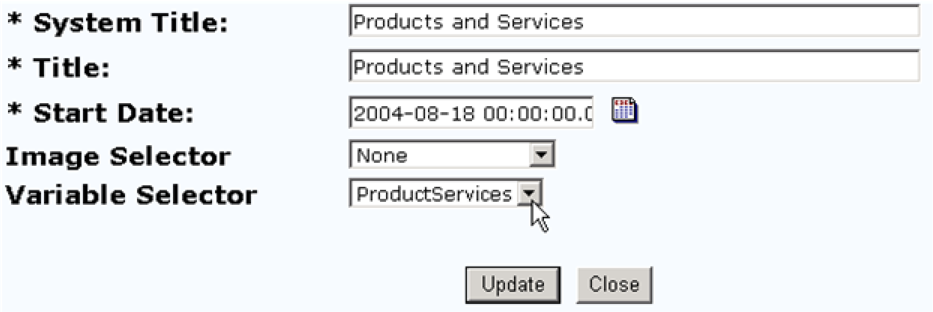
Note the class attributes, which specify style classes defined in the rxs_styles.css stylesheet. Thus, you have substantial control over the format of the final output of the navigation by modifying the stylesheet or defining your own stylesheet and pointing your output to that file. You could even implement 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.



*** System Title:**	Products and Services
*** Title:**	Products and Services
*** Start Date:**	2004-08-18 00:00:00.0
Image Selector	None
Variable Selector	ProductServices

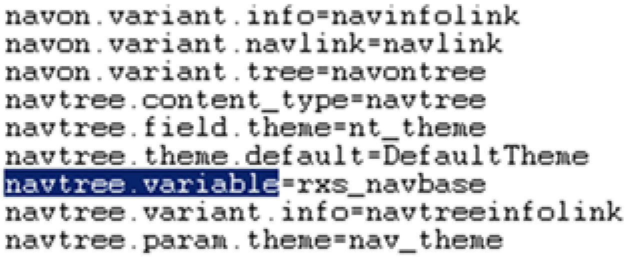
Figure 232: Defining an Alternate Variable for a Site Section

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

FastForward 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 (`<Rhythmyxroot>\rxconfig\server\Navigation.properties`).



```
navon.variant.info=navinfolink
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=navtreeinfolink
navtree.param.theme=nav_theme
```

Figure 233: 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

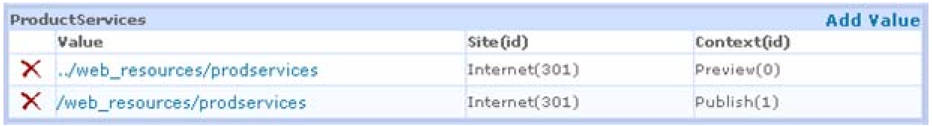
For the purposes of this procedure, we will assume that you are using the default Variable Selector Context Variable, rxs_navbase.

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)
✕	../web_resources/prodservices	Internet(301)	Preview(0)
✕	/web_resources/prodservices	Internet(301)	Publish(1)

Figure 234: 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.

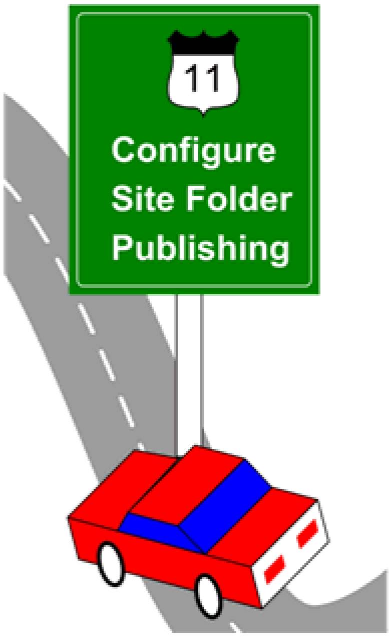
Navon properties are only used in Templates.

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: ▪ ANCESTOR: a node in the path of the Navon higher than the PARENT Navon node ▪ DESCENDANT: A node in the path after the Navon ▪ NONE: No other category applies ▪ PARENT: The immediate predecessor of the Navon in its path. ▪ SELF: The Navon itself. ▪ SIBLING: Another Navon that shares the PARENT of the Navon.
nav:url	String	The URL of the Navon's landing page.
nav:landingPage	Node	The landing page Content Item.

Variable	Data Type	Description
nav:leaf	Boolean	Boolean specifying whether the Navon is a leaf (in other words, has no children) ▪ True: The Navon is a leaf (has no children). ▪ False: The Navon is not a leaf (has children).
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

CHAPTER 9

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 67) by setting up the Site's folder hierarchy in Rhythmyx Content Explorer and registering the Site, which involved specifying the root Site Folder, address, and other information.

Before you implement the other publishing components, let's review 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 the Rhythmyx Publishing system are the Publishing engine in the Rhythmyx server, Delivery Types, 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.
- 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 Delivery Type handles the physical delivery of published content to the final output location.

Other components of the Rhythmyx publishing system are Contexts, Location Schemes, and Variables.

- 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. 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 on a file system to publish content.
  - creating URLs so that content items can link to each other when they appear in a browser.
- 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.

Publishing begins when a publishing request is submitted to the Rhythmyx server. A publishing request may be generated automatically as a scheduled task, or it may be manually requested. When the Rhythmyx server receives the publishing request, it generates a Publishing job. The Publishing job processes the Content Lists to generate a list of Content Items to publish, and submits them to the Assembly engine. When the Publishing job receives the assembled Content Items, it calls the Delivery Type to deliver the assembled content to the final output location.

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 67), so we will not duplicate it in this section.

- Delivery Type

  For the purposes of this chapter, we will be implementing delivery to a File System. The Standard File System Delivery Type included with Rhythmyx meets the needs of most implementations, so we will use it in this chapter. If you need to deliver your output via SSL, see ***Setting Up SSL*** (on page 401). If you need to implement delivery to a database, see ***Database Publishing in Rhythmyx*** (on page 363).

- Content Lists

  The following Content Lists are specified in the example specification:

  - 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 (Full Binary Content List). Publishing binary Content Items typically takes longer than publishing non-binary content, so in most implementations, a separate Content List is implemented to publish them. This Content List is typically run less frequently than the Content List that publishes non-binary content.
  - 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 (Full Non-binary Content List). As noted above, most implementations include separate Content Lists to publishing binary and non-binary content. In most cases, the non-binary Content List is run more frequently than the binary Content List because non-binary content can usually be published more quickly.
  - A Content List that selects all new and modified public Content Items (incremental Content List) and specifies that they should be delivered to the registered Site on the file system. This is another standard Content List in most implementations which allows periodic updates to the published Site without taking the time and resources to completely re-publish the Site.

Note: Earlier Versions of Rhythmyx (Version 6.5.2 and earlier) required a separate Content List to unpublish expired Content Items. In Rhythmyx Version 6.6 unpublishing is handled by the Edition.

- Editions

  The following Editions are listed in the example specification.

  - An Edition scheduled to run once a week that publishes all items ready to be published to the Enterprise Investments site, and unpublishes all Content Items that have been removed or have entered the Archive State. Most implementations include an edition that republishes the Site at intervals, though the frequency varies from one implementation to another.
  - 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 Content Items that 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 the example specification:

  - A preview Context

    Rhythmyx includes this Context by default. It must exist so that users can preview assembled content in Content Explorer before publishing it.

  - A publishing Context

    This Context tells the publishing engine where to deliver the Content Items that have been generated. All implementations require at least one publishing Context.

  - A Context for assembling items

    In most cases, the URL for links between published Content Items is different from the URL to which the individual Content Items are published (Most Web servers base these links on the Site address rather than on the publishing root defined for the Site.) A special Context is required to provide these URLs. In Rhythmyx, this Context is generally referred to as an assembly Context. Note that there are some cases, (such as publishing to a database target) where this Context is not necessary.

- Location Schemes

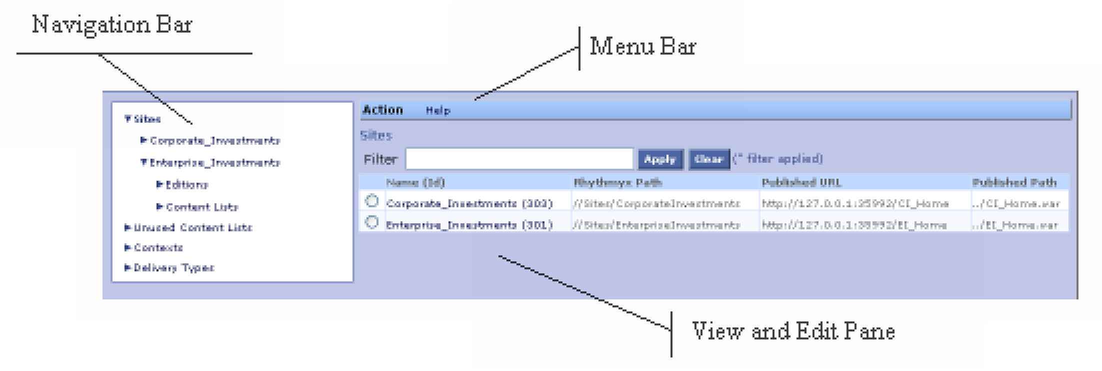
  The following Location Schemes are listed in your specifications:

  - 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 DesignTab

The publishing components in your system, including Sites, Delivery Type Configurations, Content Lists, Editions, Contexts, Location Schemes, and Variables are maintained or configured in the Publishing Design tab of Content Explorer. This tab is only available to users with the correct access rights.

On the left side, the Publishing Design tab displays a navigation pane with links to the editors for adding and modifying different publishing components. The right side of the Publishing Design tab is the Edit and View pane, which displays the editor for the selected object.



	Name (Id)	Rhythmyx Path	Published URL	Published Path
○	Corporate_Investments (303)	//Sites/CorporateInvestments	http://127.0.0.1:35992/CI_Home	../CI_Home.war
○	Enterprise_Investments (301)	//Sites/EnterpriseInvestments	http://127.0.0.1:35992/EI_Home	../EI_Home.war

Figure 235: Parts of the Publishing Design Tab

Defining Content Lists

A Content List is a Rhythmyx query that defines which Content Items are extracted from the database for publishing.

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 all eligible Content Items on the Site (a Full Publish) or only those content items that have been added or updated since the last Publication run (a Normal 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 makes 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 three Content Lists:

- one that selects all Public binary Content Items (EI_FullBinary)
- one that selects all Public non-binary Content Items (EI_FullNonBinary)
- one that selects new and modified Content Items (EI_Incremental)

The following topics describe how to implement these Content Lists. FastForward includes additional Content Lists that we will not recreate in this section.

Note: The data in the following procedures is included as examples. Use the data specified in your development plan when defining your own Content Lists.

Defining the Full Binary Content List

The first type of Content List we will illustrate is the Full Binary. Full Binary Content Lists publish all binary content that is in a Public state within the specified Site Folder.

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

To define the Full Binary Content List:

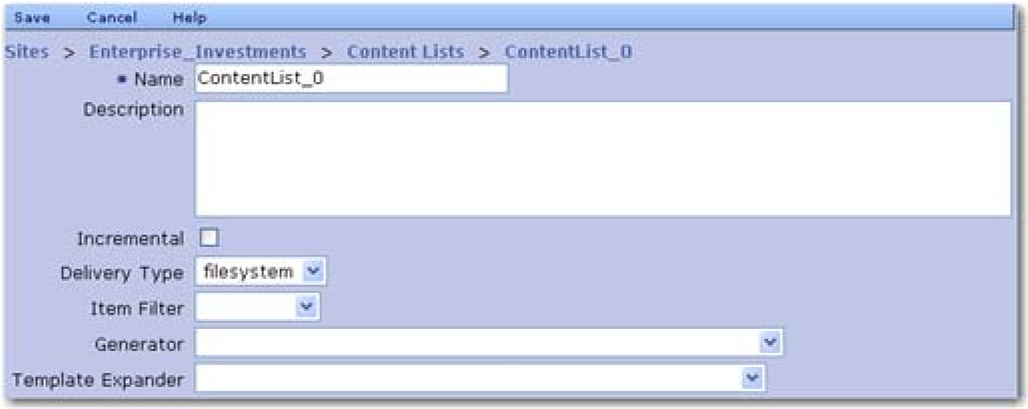
1. Log in to Rhythmyx Content Explorer.
2. Click the Publishing Design tab.
3. In the Navigation pane on the left side of the dialog, double-click on the Unused Content Lists link.

   Content Explorer displays the List Content List dialog.

Note: Rhythmyx includes two versions of this dialog. The version accessed in this way lists only Content Lists that are not currently included in an Edition. You can also access a version of the dialog under the Editions node of the Navigations pane. That version lists only Content Lists that are associated with an Edition.

4. In the Menu bar, choose *Action* > *Create New Content List*.

Content Explorer displays the Content List editor.



Sites > Enterprise_Investments > Content Lists > ContentList_0

* Name: ContentList_0

Description:

Incremental: □

Delivery Type: filesystem

Item Filter:

Generator:

Template Expander:

.

Figure 236: Content List Editor when creating a new Content List

5 The Name field is mandatory and defaults to ContentList_0. This field is editable. It is a best practice to give the Content List a unique Name. Content List names must begin with a letter, and can contain letters, numbers, underscores, hyphens, or dots (periods), but cannot contain spaces. Enter *rffEiFullBinary* as the name the Content List.

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 We want this Content List to publish all Public Content Items of the specified Content Types for the Site, so leave the Incremental checkbox unchecked.

8 Choose the Delivery Type used for the Content List. The Delivery Type determines how assembled content will be delivered to the target location. Options include all Delivery Types defined in the system. The following Delivery Types are included in Rhythmyx installations by default:

- filesystem

  Delivers content to the local file system, including any mapped or mounted drive locations.

- ftp

  Delivers content via FTP.

- sftp

  Delivers content via secure FTP

- database

  Delivers content to a database.

For the purposes of this exercise, we will choose *filesystem*.

9 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.

Choose *Public*. We want to publish all Content Items in a public State in the Content List.

10 The Generator field specifies the Content List Generator extension that creates the list of Content Items to be published. The following Content List Generators are included in Rhythmyx installations by default:

- Java/Global/percussion/system/sys_PublishedSiteItems

  Outputs the list of Content Items currently published on the Site; must be used with the sys_SiteTemplateExpander. Used primarily for unpublishing in legacy systems implemented on Rhythmyx Version 6.5 or earlier.
- 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 demand publishing. 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.

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 196) 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 the Template Expander extension that chooses 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.
- 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. When you choose this option, the Content List Editor adds the **siteid** and **default_template** fields below the **Template Expander** drop list.

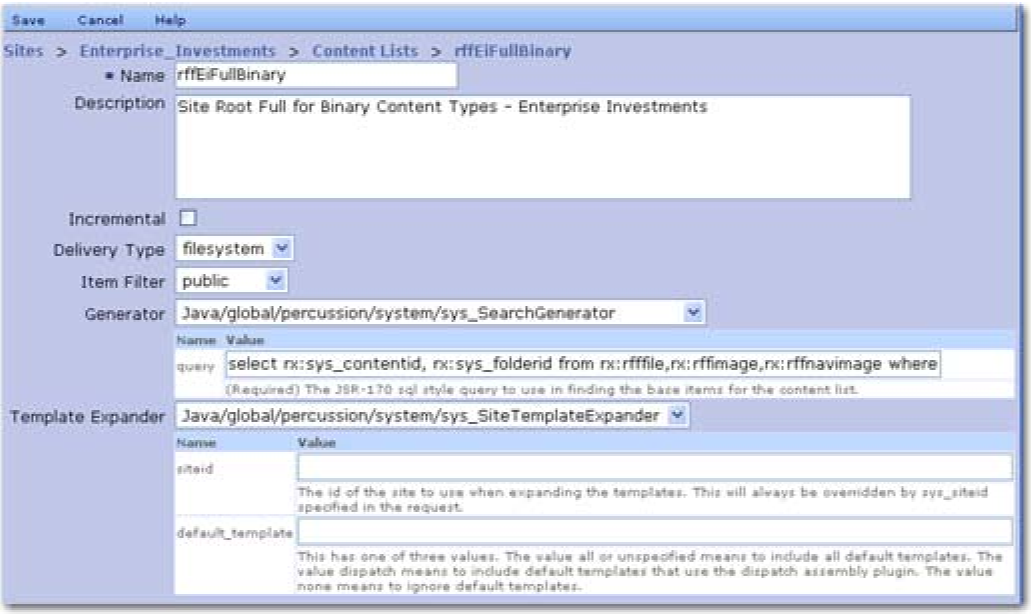
13 In the **siteid** field below the Template Expander field enter 301, the site id for Enterprise Investments.

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*.

Your completed editor should resemble the following screenshot:



Sites > Enterprise_Investments > Content Lists > rffEIFullBinary

* Name: rffEiFullBinary

Description: Site Root Full for Binary Content Types - Enterprise Investments

Incremental: □

Delivery Type: filesystem

Item Filter: public

Generator: Java/global/percussion/system/sys_SearchGenerator

Name	Value
query	select rx:sys_contentid, rx:sys_folderid from rx:rfffile,rx:rffimage,rx:rffnavimage where (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

Name	Value
siteid	The id of the site to use when expanding the templates. This will always be overidden by sys_siteid specified 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.

Figure 237: Enterprise Investments Full Binary Content List

15 To save the Content List, in the Menu bar, click *Save*.

Defining the Full Non-Binary Content List

The second Content List we define is a Full Non-binary Content List. This Content List publishes all non-binary Content Items within the specified Site Folder. Standard practice is to implement separate Content Lists for binary and non-binary content because binary content typically takes longer to publish. You can thus schedule separate and more frequent publishing of non-binary content as well as including it with binary content.

We can create this Content List by copying the ***Full Binary Content List*** (see page 314) and modifying the appropriate data.

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

To define the Full Non-Binary Content List:

1 On the Publishing Design tab, click on the Unused Content Lists tab to display the list of Content Lists not associated with an Edition.

2 Select the radio button next to the rffEiFullBinary Content List.

3 In the Menu bar, choose *Action > Copy Selected Content List*.

Rhythmyx copies the Content List and displays it in the Content List editor. The copied Content List has the name *Copy_of_rffEiFullBinary*.

4 Change the Name to *rffEiFullNonBinary*.

5 Change the Description to *Site Root Full for Non-Binary Content Types - Enterprise Investments*.

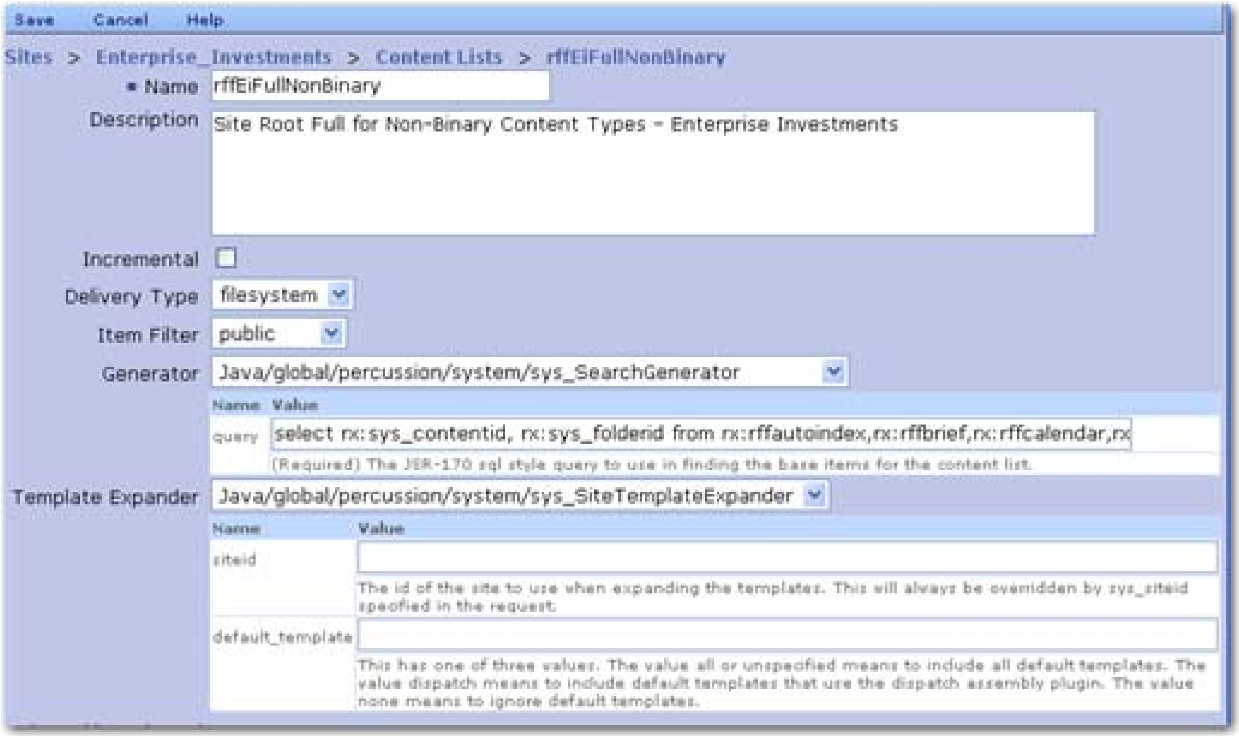
6 Change the Query to

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, Contact, Event, External link, Generic Word, Generic, Home, and Press Release Content Items in the Site Folder path //Sites/EnterpriseInvestments and its Subfolders.

7 For the remaining fields, the Full Nonbinary Content List uses the same values as the Full Binary Content List. Do not change any of these values.

Your completed editor should resemble the following screenshot:



Sites > Enterprise_Investments > Content Lists > rffEiFullNonBinary

* Name: rffEiFullNonBinary

Description: Site Root Full for Non-Binary Content Types - Enterprise Investments

Incremental: □

Delivery Type: filesystem

Item Filter: public

Generator: Java/global/percussion/system/sys_SearchGenerator

Name	Value
query	select rx:sys_contentid, rx:sys_folderid from rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx (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

Name	Value
siteid	 The id of the site to use when expanding the templates. This will always be overridden by sys_siteid specified 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.

Figure 238: Enterprise Investments Full Non-binary Content List

8 In the Menu bar, click *Save*.

Defining the Incremental Content List

An Incremental Publish differs from a Full Publish in that it publishes only new Content Items and Content Items that have changed since they were last published. Full Publishing publishes every Content Item in a Public State.

Many customers perform a Full Publish infrequently (such as every one or two weeks), and an incremental publish very frequently (one or more times a day). Since fewer items are published during Incremental Publishing, fewer system resources are used, decreasing processing time.

We can create this Content List by copying an existing Content List (we will use the Full Binary Content List) and changing the appropriate data.

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

To define the Incremental Publish Content List:

1 Copy the EI_FullBinary Content List. For details see steps 1-3 in ***Defining the Full Non-Binary Content List*** (on page 318).

2 Change the Name field to *rffEiIncremental*.

3 Change the Description field to *Site Root Incremental - Enterprise Investments*.

4 Since this is an Incremental Content List, check the **Incremental** checkbox. Checking this box triggers incremental publishing processing. This processing republishes Content Items that were changed since the last publishing run, publishes Content Items that have become Public since the last publishing run, and removes Content Items that expired since the last publishing run. Any Content Items that link to newly published or unpublished Content Items will also be republished.

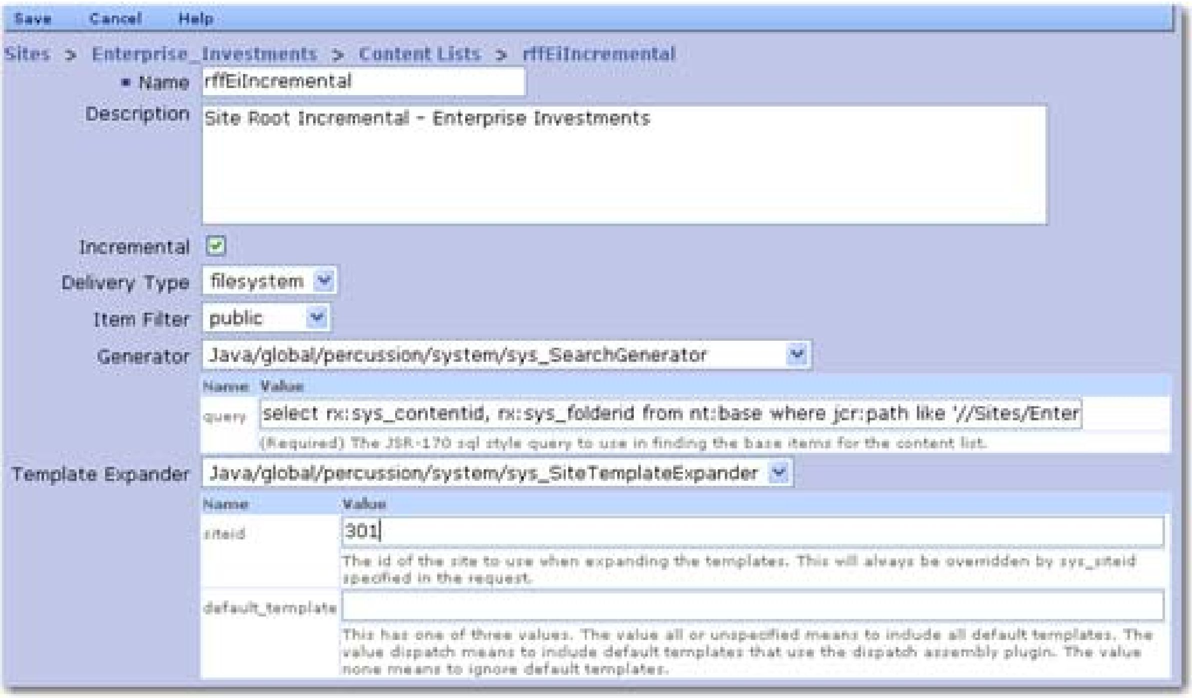
5 Change the **Query** to:

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 and its Subfolders.

6 For the remaining fields, the Incremental Content List uses the same values as the Full Binary Content List. Do not change any of these values.

Your completed editor should resemble the following screenshot:



Sites > Enterprise_Investments > Content Lists > rffEiIncremental

* Name: rffEiIncremental

Description: Site Root Incremental - Enterprise Investments

Incremental: ☑

Delivery Type: filesystem

Item Filter: public

Generator: Java/global/percussion/system/sys_SearchGenerator

Name	Value
query	select rx:sys_contentid, rx:sys_folderid from nt:base where jcr:path like '//Sites/Enter (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

Name	Value
siteid	301 The id of the site to use when expanding the templates. This will always be overridden by sys_siteid specified 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.

Figure 239: Incremental Content List

7 In the Menu bar, choose *Save*.

Defining Contexts and Location Schemes

A Context is a location or environment in which a path to an assembled output is generated. 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. Different paths are generated in each case.

Location Schemes are associated with specific contexts, and specify the rules for generating paths within that Context. 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.

The FastForward specification requires three Contexts:

- A preview Context

  This Context is included in Rhythmyx by default. It enables previewing of content by linking to referenced content such as images and other pages. It cannot be modified or deleted.
- A publishing Context

  This Context tells the Publisher where on the file system to deliver the assembled output HTML files. Every system that publishes to a file system (either directly or via FTP), requires such a Context. Such a Context is often referred to as a publish Context. In our example, we will name this Context *Publish*.
- A Context for assembling items

  This Context creates links to other Content Items. While some systems only need a publish Context, most need an additional Context, often referred to as an assembly Context, for two reasons. First, the URL for links between Content Items may be different from the location where the output files reside, even for content published to a file system. Second, when publishing to a database or other alternative output repository, content may be stored and retrieved using a different approach than a pre-generated location; for example, a system might pass a request parameter to a Web application to retrieve the correct content. In our example, we will name this Context Site Folder Assembly.

Creating the Publish Context and its Generic Location Scheme

To illustrate the creation of Contexts and Location Schemes, we will examine how the Publish Context and its Generic Location Scheme were created.

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

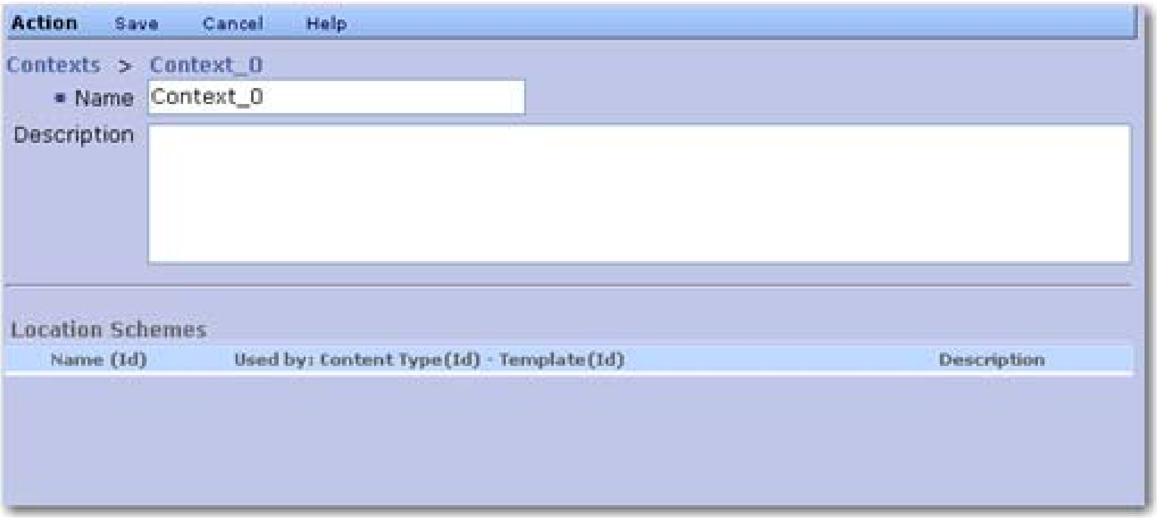
To create a Publish Context:

1. Log in to Rhythmyx Content Explorer.
2. Click the Publishing Design tab.
3. In the Navigation pane, double-click on the Contexts node.

Content Explorer displays the Context list.

4 In the Menu bar, choose *Action* > *Create Context*.

5 Content Explorer displays the Context editor.



* Name: Context_0

Description:

Figure 240: New Context

6 In the Name field, enter *Publish*.

7 In the Description field, enter *Create the appropriate path for the publishing location, related to but not identical with the assembly location.*

8 In the Menu bar, click the *Save* button.

Rhythmyx saves the Context and returns you to the Context List page.

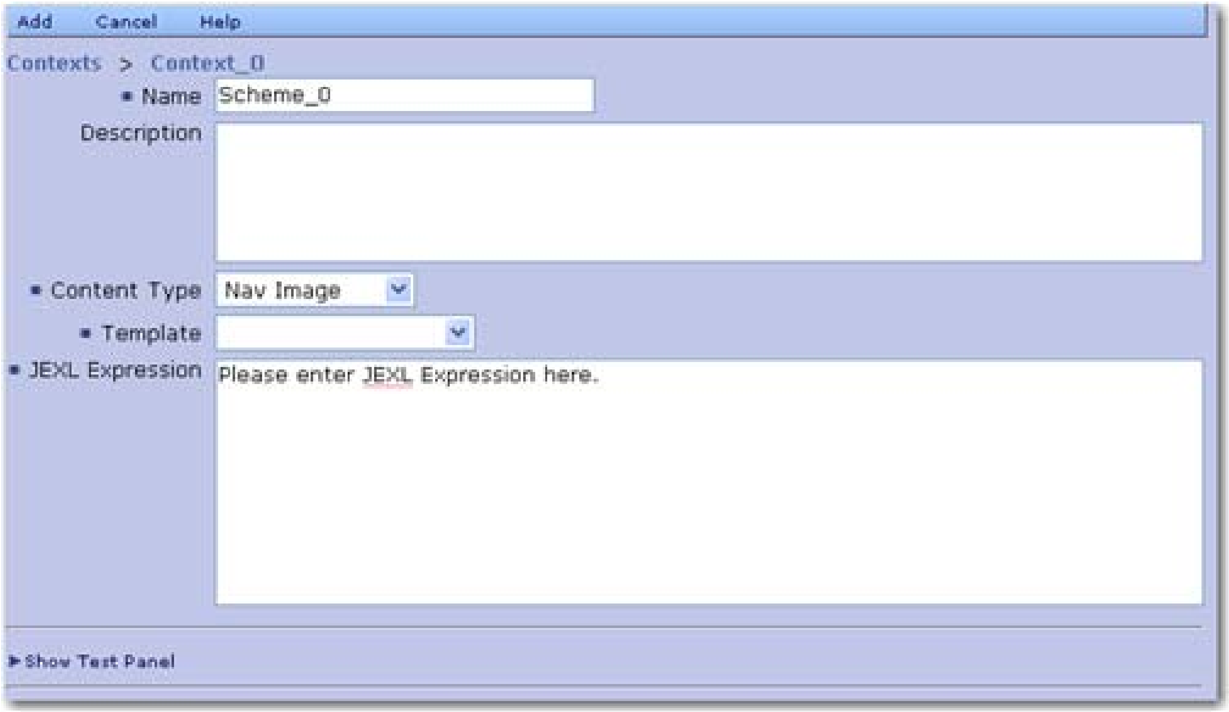
Next, add a Location Scheme to the Context:

1 Click on the name of your new Context to open it.

Rhythmyx displays the Context in the View and Edit pane.

2 In the Menu bar, choose *Action* > *Create Location Scheme*.

Content Explorer displays the Scheme Editor.



Name: Scheme_0

Content Type: Nav Image

JEXL Expression: Please enter JEXL Expression here.

Figure 241: New Location Scheme

3 In the **Name** field, enter *Generic*.

4 In the **Description** field, enter *Generic location generation for publishing.*

5 In the **Content Type** drop list, choose *Generic*. This choice specifies that this Location Scheme 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.

6 In the **Template Type** drop list, choose *D - EI Generic*. Now this is the default Location Scheme for the *Generic Content Type* when it uses the *D - EI Generic* Template.

7 In the **Expression** field, 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_su
ffix', '.html')
```

This JEXL expression uses bindings (variables or functions) already defined for Rhythmyx. See ***Bindings*** (see page 141) 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	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. 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.
$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')	$rx.location returns a hypertext link. $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.

8 You should always test new and modified JEXL expressions to ensure that they produce the desired results. We will use the *About EI HomePage Image (NYSE Papers).jpg* Content Item to test the JEXL expression:

a) Click the Show Test Panel link.

The Location Scheme Editor expands to display the test panel.

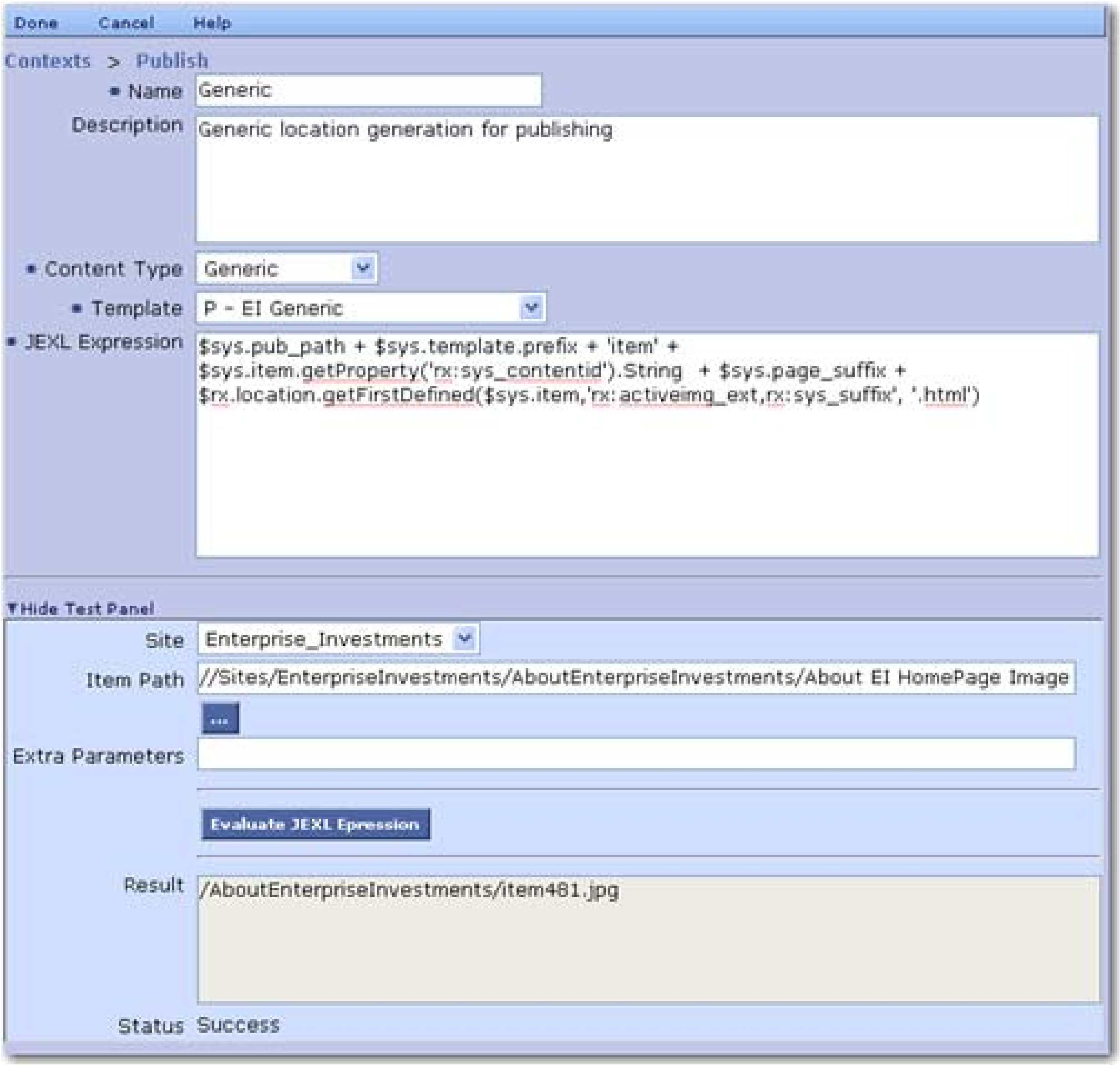
b) In the Site drop list, choose *Enterprise Investments*.

c) Click the ... button below the Item Path field and browse to the *About EI HomePage Image (NYSE Papers).jpg* Content Item.

d) None of the JEXL functions used in this expression require any Additional Parameters, so leave this field blank. If any of the JEXL functions used in the expression did require additional parameters, we would add them to the field as name=value pairs, using ampersands ("&") to separate pairs; for example: `sys_contentid=301&sys_revision=1&sys_slotid=501`.

e) Click the [**Evaluate JEXL Expression**] button.

Rhythmyx evaluates the expression and returns the results. If the expression can be evaluated, the Status indicates *Success* and the evaluated output is displayed in the Results field:



* Name: Generic

Description: Generic location generation for publishing

* Content Type: Generic

* Template: P - EI Generic

* JEXL Expression:

```
$sys.pub_path + $sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String  + $sys.page_suffix +
$rx.location.getFirstDefined($sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')
```

Site: Enterprise_Investments

Item Path: //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/About EI HomePage Image

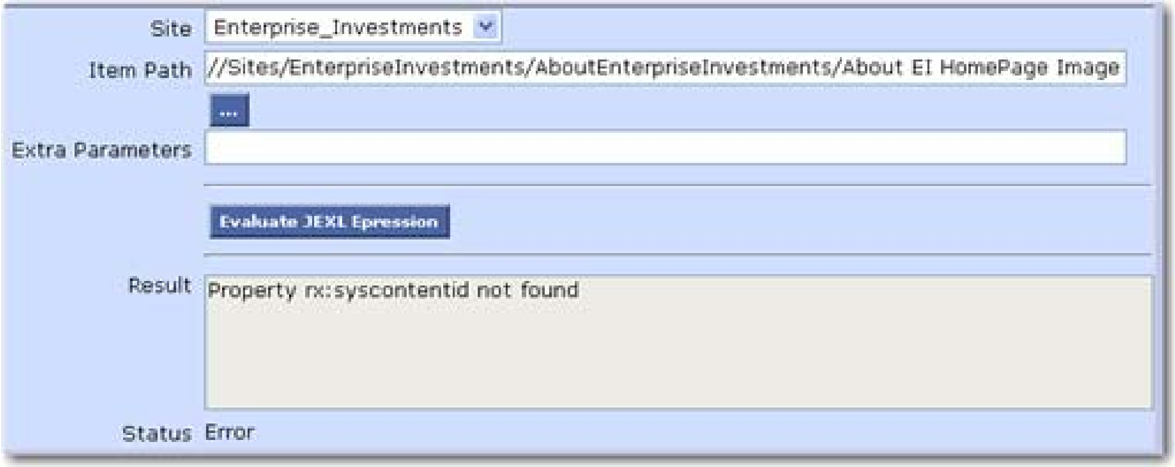
Extra Parameters:

Result: /AboutEnterpriseInvestments/item481.jpg

Status: Success

Figure 242: Successful Location Scheme

If the JEXL expression cannot be evaluated successfully, the Status indicates Error and the text describing the error is displayed in the Results field:



Site: Enterprise_Investments

Item Path: //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/About EI HomePage Image

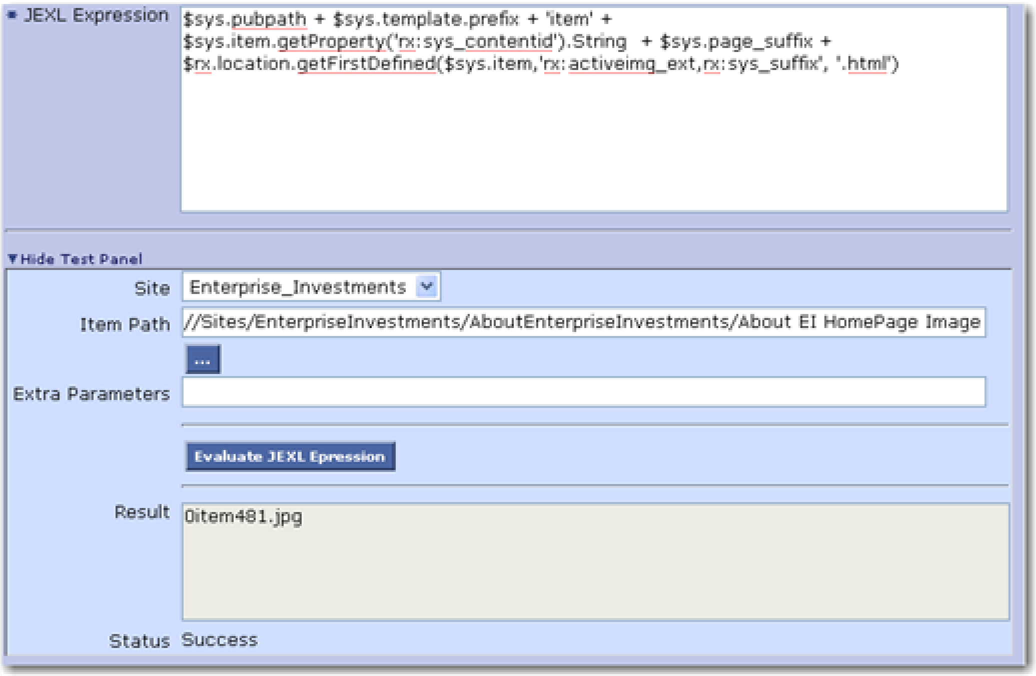
Extra Parameters:

Result: Property rx:syscontentid not found

Status: Error

Figure 243: Test Panel showing an error in the JEXL expression. In this case the property rx:sys_contentid was specified incorrectly without an underscore.

Note that a JEXL expression may return a Success status while the evaluated result does not return expected results. Unexpected results may occur simply because the JEXL expression needs additional work. For example, you may want to include a Template-defined prefix or suffix in your output, but the JEXL expression you defined a JEXL expression may not include that data. In some cases unexpected results may occur because the JEXL expression includes an error, but the error does not prevent successful processing of the expression. In the example below, the variable $sys.pub_path was defined without the underscore (as $sys.pubpath). The expression was processed successfully, but the result is incorrect.



JEXL Expression:

```
$sys.pubpath + $sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String  + $sys.page_suffix +
$rx.location.getFirstDefined($sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')
```

Site: Enterprise_Investments

Item Path: //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/About EI HomePage Image

Extra Parameters:

Result: 0item481.jpg

Status: Success

Figure 244: Testing a JEXL showing a Status of success while producing an erroneous output

Common Errors in JEXL Expressions

Some common errors that occur when creating JEXL expressions include:

- Misspelling a function.

  For example, if the expression were coded as `...$rx.location.getFirstDefind($sys.item,'rx:activeimg_ext,rx:sys_suffix', '.html')`, the system would return an error when testing the expression. The correct function name is `$rx.location.getFirstDefined`.

- Not enclosing the test condition of an IF function in parentheses.

  For example, if the expression were coded as `if $sys.crossSiteLink...`, the system would return an error when testing the expression. The correct syntax is `if ($sys.crossSiteLink)`.

- When testing a string value, not enclosing the value in quotation marks.

  For example, if the expression were coded as `if ($sys.site.path=\\EnterpriseInvestments\InvestmentAdvice.. )`, the system would return an error when testing the expression. The correct syntax is `if ($sys.site.path='\\EnterpriseInvestments\InvestmentAdvice').`

- Not ending consequent statements with semicolons (;).

  For example, the code `if ($sys.crossSiteLink) $prefix = $sys.site.url else $prefix = $sys.variables.rxs_urlroot` results in an error when testing the expression. The correct syntax is `if ($sys.crossSiteLink) $prefix = $sys.site.url; else $prefix = $sys.variables.rxs_urlroot;`.

- Using an incorrect JEXL variable

  For example, using $sys.site.path rather than $sys.pub_path. The two variables return different results ($sys.site.path returns the Site ID, while $sys.pub_path returns the Folder path in Content Explorer).

Be sure to review the Result when you test your JEXL expression. The returned results may not match your desired output. Moreover, the system returns a Success status if the JEXL expression can be processed without the JEXL processing engine returning an error. In some cases incorrect input may result in erroneous output even though the Status is Success.

Creating Additional Contexts

Like Location Schemes, Contexts are often very similar. For example, the Site_Folder_Assembly Context in FastForward has the same set of Location Schemes as the Publish Context, but with different JEXL expressions to generate different results. Thus, you can often create a new Context quickly and easily by copying an existing Context. When you copy a Context, all of the Location Schemes in the Context are also copied.

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

To create the Site Folder Context:

1 On the Publishing Design tab, click the Contexts link to display the list of Contexts.

2 Select the radio button next to the Publish Context.

3 In the Menu bar, choose *Action* > *Copy Selected Context*.

Rhythmyx copies the Context and displays it in the Context editor.

4 Change the Name to *Site_Folder_Assembly*.

5 Change the Description to *Create the appropriate path for the site folder assembly location*.

6 Each of the Location Schemes was copied with the Context with the name *Copy_of_<Location Scheme>*, where Location Scheme is the name of the Location Scheme. Edit each Location Scheme to change the JEXL expression to generate the desired results. You may also want to modify the names of the Location Schemes, at least removing the "Copy_of_" text.

7 In the Menu bar, click Done.

Creating Additional Location Schemes

In many cases,. Locations Schemes are very similar, and you can create additional Location Schemes quickly and easily by copying an existing Location Scheme. For example, the InactiveNavImage Location Scheme in the Publish Context is used when publishing Content Items of the NavImage Content Type using the B-Inactive Image Template, but the JEXL expression is very similar to the expression used in the Generic Location Scheme:

```
$sys.pub_path + $sys.template.prefix + 'item' +
$sys.item.getProperty('rx:sys_contentid').String  +
$rx.location.getFirstDefined($sys.item,'rx:inactiveimg_ext,rx:sys_suffix
', '.gif')
```

Compare this expression to the expression used in the Generic Location Scheme illustrated in the topic ***Creating the Publish Context and its Generic Location Scheme*** (on page 321). Thus, we can quickly create the InactiveNavImage Location Scheme by copying the Generic Location Scheme and modifying the appropriate data.

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

To create the InactiveNavImage Location Scheme:

1 In the Publishing tab, open the Publish Location Scheme.

2 Select the radio button next to the Generic Location Scheme.

3 In the Menu bar, choose *Action* > *Copy Selected Location Scheme*.

Rhythmyx copies the Location Scheme and displays it in the Location Scheme editor. The copied Content List has the name *Copy_of_Generic*.

4 Change the Name field to *InactiveNavImage*.

5 Change the Description field to *Inactive navigation image publishing location*.

6 In the Content Type field, choose *Nav Image*.

7 In the Template field, choose *B - Inactive Image*.

8 Modify the JEXL Expression using the example code illustrated above.

9 Test the expression to ensure that it returns the expected results.

10 In the Menu bar, click *Done*.

Creating Editions

Now that we have created Content Lists and Contexts we are ready to create 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:

- First, because a page is composed from multiple items, and these items themselves may be groupings of other items, more elemental items should be published first, followed by higher level items, and finally by pages. In other words, images should be published before articles because articles will typically include images. Articles should be published before indices because index items typically include article items.
- Second, different Content Lists typically have different publishing rules, which specify the response to errors generated in earlier Content Lists, such as omitting an item if a child item is not published, publishing the item without the child item, or publishing the item with an error message.

You can also define one or more tasks to run with your Edition. You can run tasks either before the Edition itself is run or after the Edition completes processing. For example, you can define a pre-Edition task to establish a connection to the remote Web server before publishing the content and breaking the connection after the Edition is complete; or you might define a post-Edition task to run a link checker.

The FastForward specification includes the following 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. We will call this Edition rffEIFull. It will include the following Content Lists:
  - rffEiFullBinary
  - rffEiFullNonbinary
- 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 will call this Edition rffEIIncremental. It will include the rffEiIncremental Content List.

We will add a task to each of these Editions to run a link checker after the Edition is finished to ensure that none of the links in the Edition are broken. (Note: See the document *Setting Up the Rhythmyx Production Environment* for details about defining a publishing schedule.)

FastForward includes additional Editions; the procedures to create those Editions are essentially the same as for the Editions we illustrate.

Full Publish Edition

In this exercise we will 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 or has been purged from the system.

For the purposes of this exercise, we will assume that the following design elements have been created:

- Content lists
  - EI_Binary

    - EI_NonBinary
- Contexts
    - Publish
    - Site Folder Assembly

We will add both Content Lists to this Edition. Each Content List will be associated with both listed Contexts.

We will also add a task that queries the W3C Link Checker to validate all of the links in the published Site.

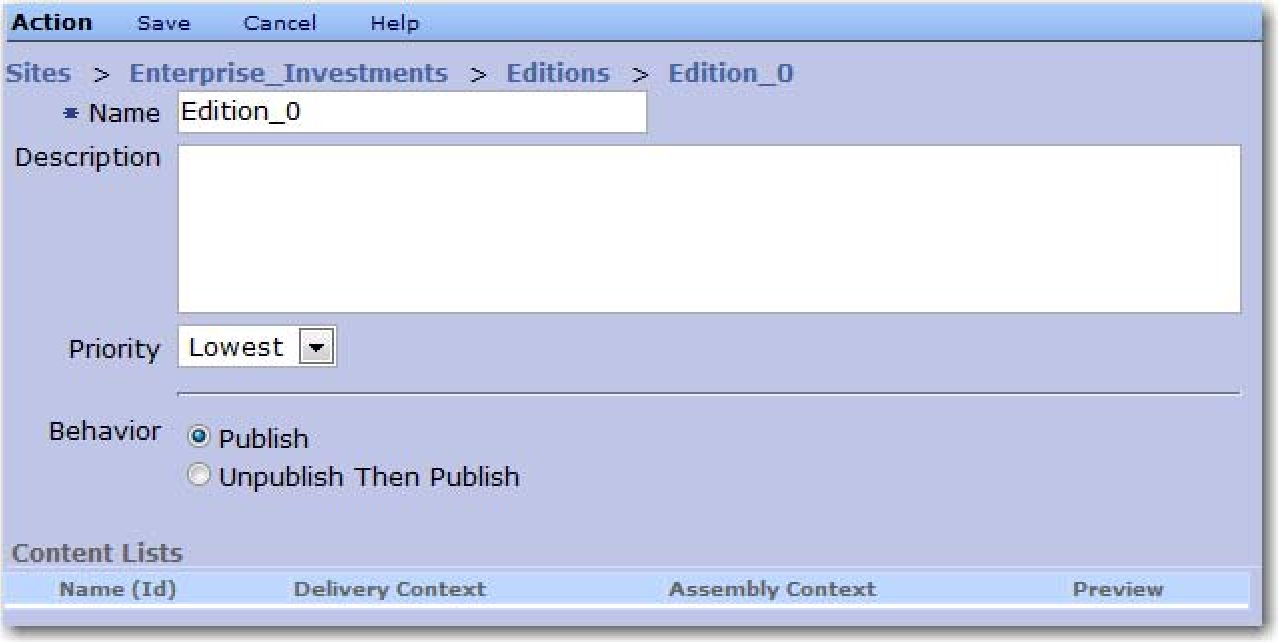
NOTE: The data in this procedure is included as an example. Substitute the data for your own objects.

To create the Full Publish Edition:

1. Log in to Rhythmyx Content Explorer.
2. Click the Publishing Design tab.
3. Expand the Sites node. Expand the Enterprise Investments node. Click on the Editions node.

   Content Explorer displays the Editions List. Since we have not yet defined any Editions, the list is empty.
4. In the Menu bar, choose *Action* > *Create New Edition*.

   Content Explorer displays the Edition Editor.



* Name: Edition_0

Description:

Priority: Lowest

Behavior: ● Publish ○ Unpublish Then Publish

Figure 245: New Edition Editor

5. In the Name field, enter *EI_Full*.
6. In the Description field, enter *Publishes all Content Items in a Public State on the Enterprise Investments Site*.

7 In the Priority field, choose *Lowest*. This option specifies that the Edition will not interrupt the processing of other Editions, and that it will be interrupted by processing on Editions with a priority of Low or higher. Typically, full Editions are specified with a priority of Low or Lowest.

8 Select the Unpublish Then Publish radio button. This option removes any expired Content Items from the Site, then publishes all Content Items in a Public State.

9 To add the rffEIBinary ContentList:

a) In the Menu bar, choose *Action* > *Add Content List Association*.

Context Explorer displays the Edition - Add or Edit Content List dialog.

b) In the Content List table, check the box of the *rffEiFullBinary* and *rffEiFullNonBinary* Content Lists.

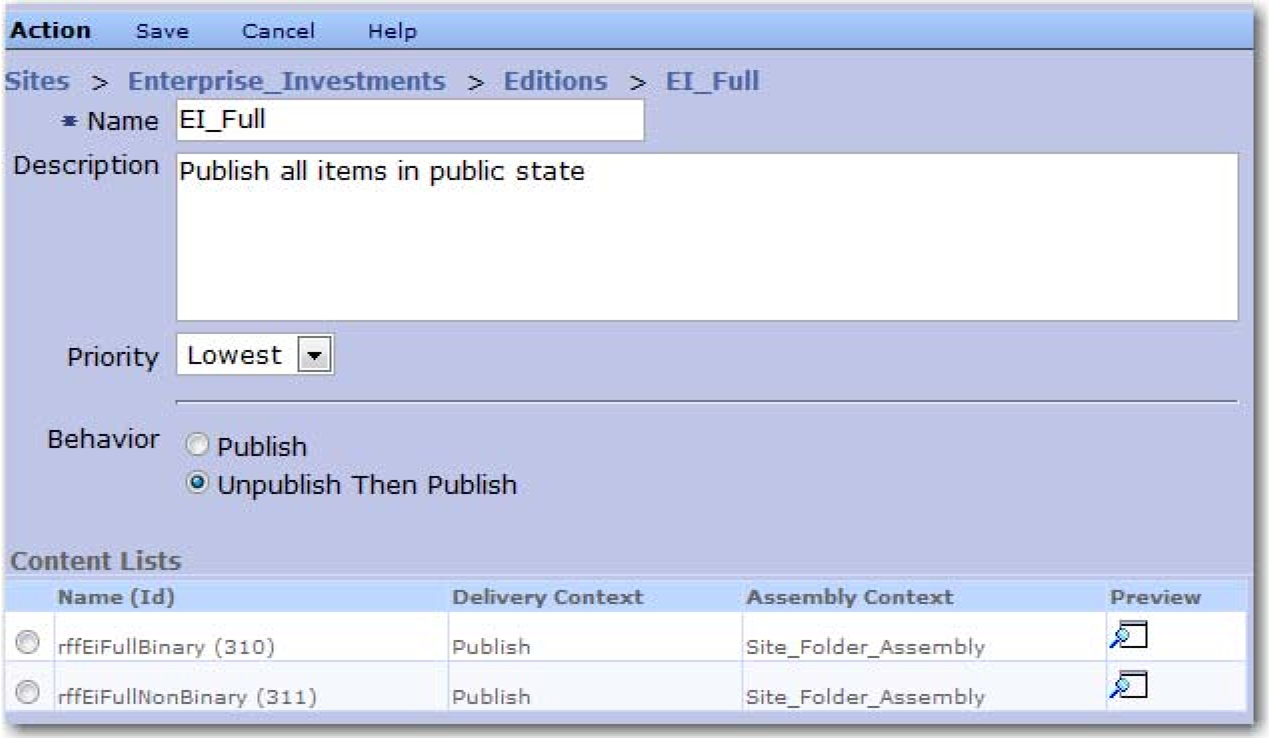
c) In the Assembly Context drop list, choose *Site Folder Assembly*.

d) In the Delivery Context drop list, choose *Publish*.

The AuthType field is used when maintaining legacy systems implemented in Rhythmxy Version 5.7 and earlier. Do not enter a value in this field.

e) Click the [**Save**] button to save the associations.

The following graphic illustrates the Edition after the Content Lists have been added:



Sites > Enterprise_Investments > Editions > EI_Full

* Name: EI_Full

Description: Publish all items in public state

Priority: Lowest

Behavior: ○ Publish ● Unpublish Then Publish

Content Lists

	Name (Id)	Delivery Context	Assembly Context	Preview
○	rffEiFullBinary (310)	Publish	Site_Folder_Assembly	🔍
○	rffEiFullNonBinary (311)	Publish	Site_Folder_Assembly	🔍

Figure 246: Edition with Content Lists

10 To add the link checker task:

a) In the Menu bar, choose *Action* > *Add Post Task*.

Content Explorer adds the Post Tasks table to the Edition editor with a row for the new task.

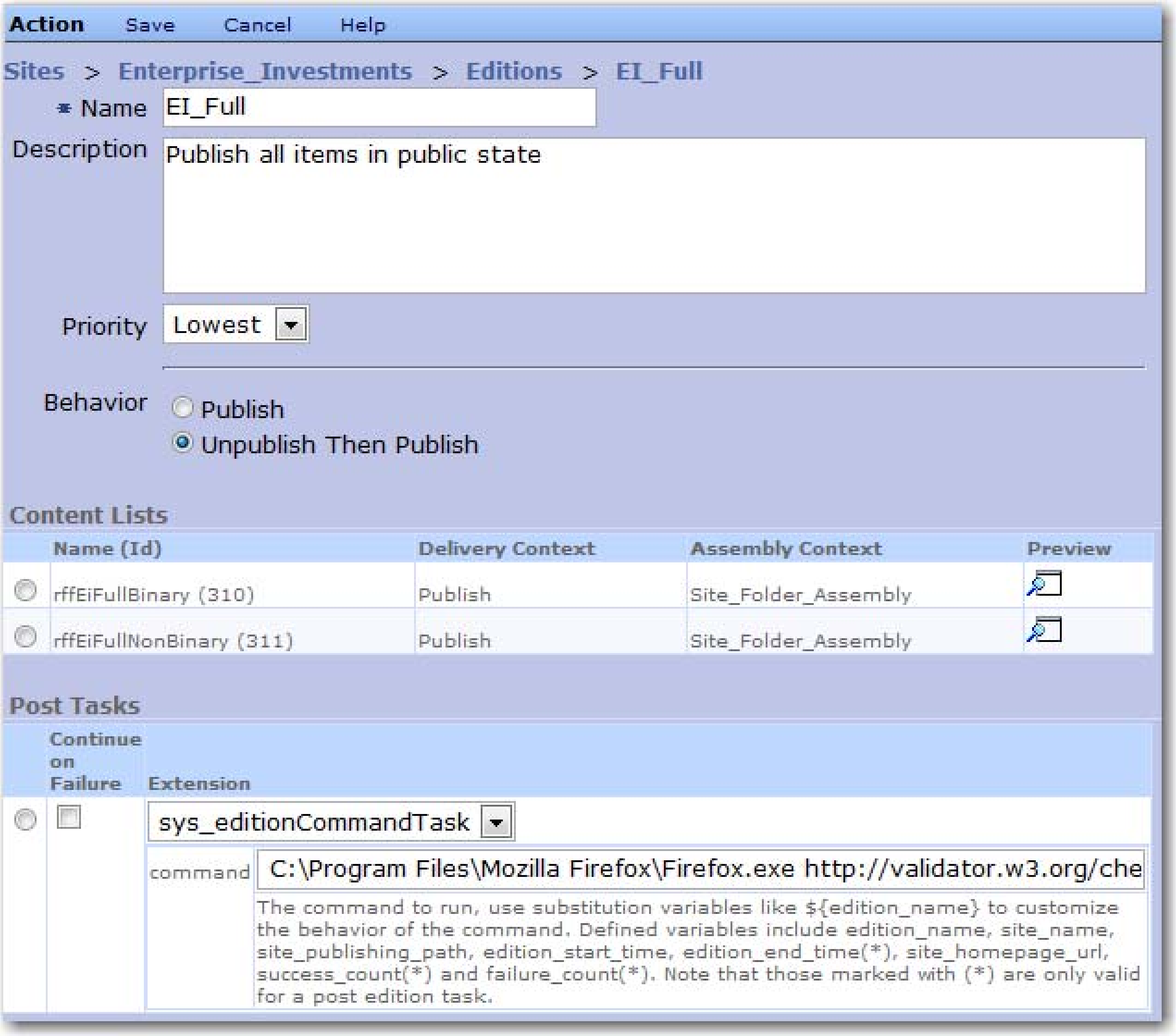
b) As checking links is a task we want to execute after publication of the Edition is complete, leave the Continue on Failure box unchecked. This box is only useful for tasks run before processing of the Edition begins. It indicates that processing of the Edition should continue even if processing of the task does not complete successfully.

c) In the Extension drop list, choose sys_editionCommandTask. This is default task extension that ships with Rhythmyx, but custom task extensions can be implemented. For details, see the (Xref to appropriate location in Tech Ref.).

d) When you choose the sys_editionCommandTask, Content Explorer displays the Command field. We will use the Firefox browser to run the linkchecker. Enter *C:\Program Files\Mozilla Firefox\Firefox.exe http://validator.w3.org/checklink?uri=www.enterpriseinvestments.com&hide_type=all&depth=&check=Check*. This command queries the W3C link checker application to check the links in the Enterprise Investments Site using the Firefox Web browser.

11 To save the Edition, in the Menu bar, click [**Save**].

The completed Edition resembles the following screenshot:



* Name: EI_Full

Description: Publish all items in public state

Priority: Lowest

Behavior: ○ Publish ● Unpublish Then Publish

Content Lists

	Name (Id)	Delivery Context	Assembly Context	Preview
○	rffEiFullBinary (310)	Publish	Site_Folder_Assembly	🔍
○	rffEiFullNonBinary (311)	Publish	Site_Folder_Assembly	🔍

Post Tasks

	Continue on Failure	Extension
○	□	sys_editionCommandTask

command: C:\Program Files\Mozilla Firefox\Firefox.exe http://validator.w3.org/che

The command to run, use substitution variables like ${edition_name} to customize the behavior of the command. Defined variables include edition_name, site_name, site_publishing_path, edition_start_time, edition_end_time(*), site_homepage_url, success_count(*) and failure_count(*). Note that those marked with (*) are only valid for a post edition task.

Figure 247: Edition with Tasks

Incremental Edition Edition

In this exercise we will define an incremental Edition, EI_Incremental. An incremental Edition removes any Content Items that have expired or been removed from the Site, publishes any Content Items that have become Public since the last publishing run, and republishes any Content Items that changed since that run.

For the purposes of this exercise, we will assume that the following design elements have been created:

- Content lists
  - rffEIIncremental
- Contexts
  - Publish

- Site Folder Assembly

The Edition will include the rffEiIncremental Content List, which will be associated with both Contexts.

When you want to create an Edition that shares configurations with an existing Edition, you can create the new Edition quickly by copying the existing Edition and modifying the copy. We will use that technique to create the EI_Incremental Edition, copying ***the EI_Full Edition we created previously*** (see "Full Publish Edition" on page 330).

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

To create the EI_Incremental Edition:

1 Log in to Rhythmyx Content Explorer, and go to the Publishing Design tab.

2 Expand the Sites node. Expand the Enterprise Investments node. Click on the Editions node.

Content Explorer displays the Editions List.

3 Select the radio button next to the EI_Full Edition.

4 In the Menu bar, choose *Action* > *Copy Selected Edition*.

Rhythmyx copies the Edition and displays it in the Edition Editor. The new Edition has the name *Copy of EI_Full*, but otherwise has the same data as the original Edition.

5 Change the Name to *EI_Incremenal*.

6 Change the Description to *Publish only new and modified Items in a Public State*.

7 Change the Priority to *Medium*. This option specifies that an Edition specified with a priority of High or Highest will interrupt processing of this Edition, and that this Edition will interrupt processing of Editions with a Low or Lowest priority. Typically, incremental Editions are assigned a priority of Medium.

8 The Unpublish Then Publish radio button is already selected (this configuration is inherited from the EI_Full Edition). This option specifies that expired Content Items will be removed from the Site, and newly Public Content Items will be published, and Content Items modified since the last publishing run will be republished. Do not change this configuration.

9 Remove the rffEIFullBinary and rffEIFullNonbinary Content Lists from the Edition:

a) Click in the radio button of the rrfEIFullBinary Content List.

b) In the Menu bar, choose *Action* > *Delete Selected Row*.

c) Repeat Steps a and b to remove the rffEIFullNonbinary Content List.

10 Add the rffEIIncremental ContentList to the Edition:

a) In the Menu bar, choose *Action* > *Add Content List*.

Context Explorer displays the Edition - Add or Edit Content List dialog.

b) In the Content List table, check the box of the *rffEIIncremental* Content List.

c) In the Assembly Context drop list, choose *Site Folder Assembly*.

d) In the Delivery Context drop list, choose *Publish*.

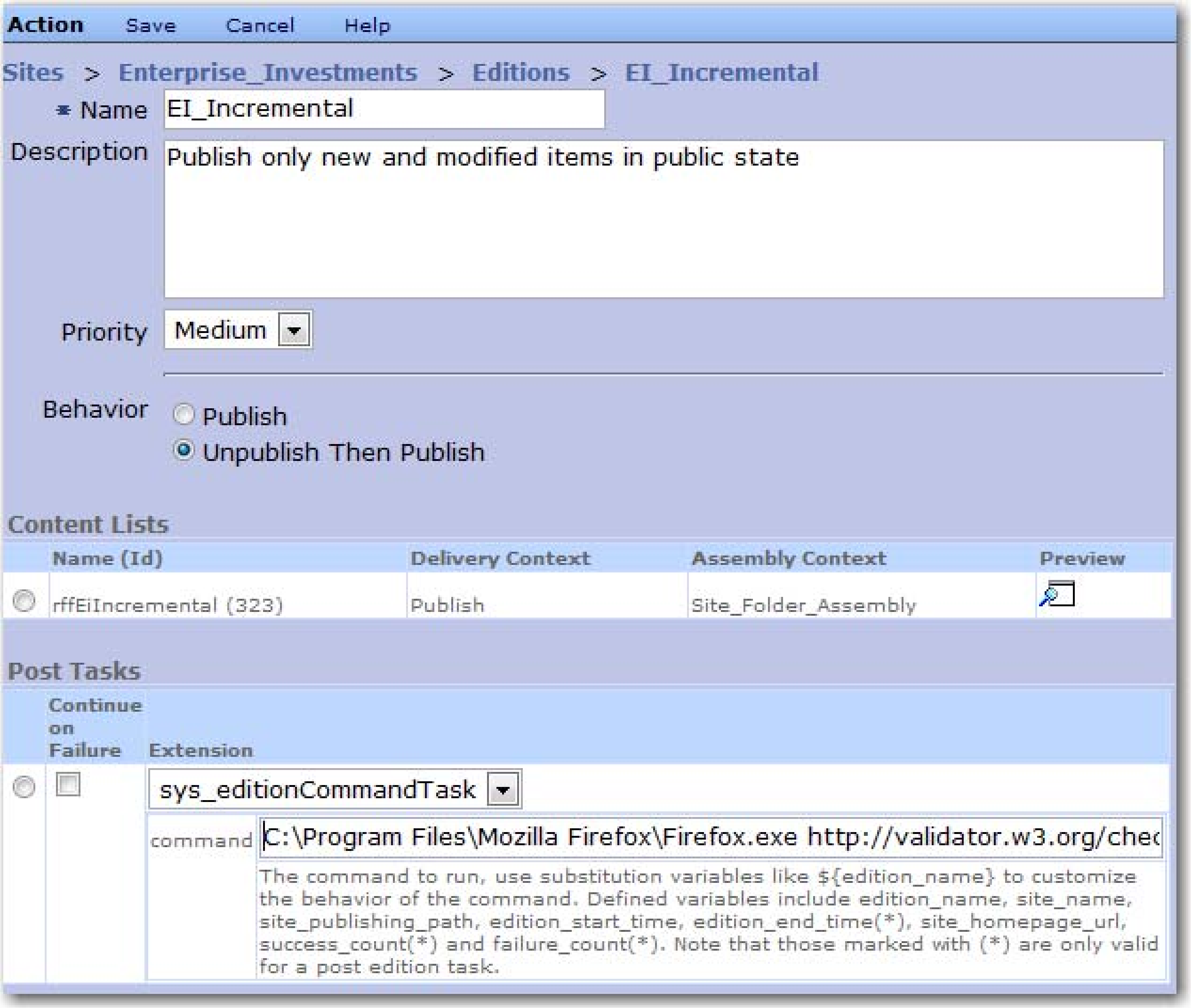
The AuthType field is used when maintaining legacy systems implemented in Rhythmxy Version 5.7 and earlier. Do not enter a value in this field.

e) Click the [**Save**] button to save the association.

11 The post task to run the link checker was copied with the Edition. Do not modify this task configuration.

12 To save the Edition, in the Menu bar, click [**Save**].

The completed Edition resembles the following screenshot:



Sites > Enterprise_Investments > Editions > EI_Incremental

* Name: EI_Incremental

Description: Publish only new and modified items in public state

Priority: Medium

Behavior: ○ Publish ● Unpublish Then Publish

Content Lists

	Name (Id)	Delivery Context	Assembly Context	Preview
○	rffEiIncremental (323)	Publish	Site_Folder_Assembly	

Post Tasks

	Continue on Failure	Extension
○	□	sys_editionCommandTask

command: C:\Program Files\Mozilla Firefox\Firefox.exe http://validator.w3.org/chec

The command to run, use substitution variables like ${edition_name} to customize the behavior of the command. Defined variables include edition_name, site_name, site_publishing_path, edition_start_time, edition_end_time(*), site_homepage_url, success_count(*) and failure_count(*). Note that those marked with (*) are only valid for a post edition task.

Figure 248: Incremental Edition

Testing your Content Lists

After you have added Content Lists to an Edition, you can preview the Content Lists to ensure they are returning the Content Items you expect.

In this example, we will preview the rffEIFullNonbinary Content List associated with the EI_Full Edition.

To preview the rffEIFullNon-binary Content List:

To test your Content Lists:

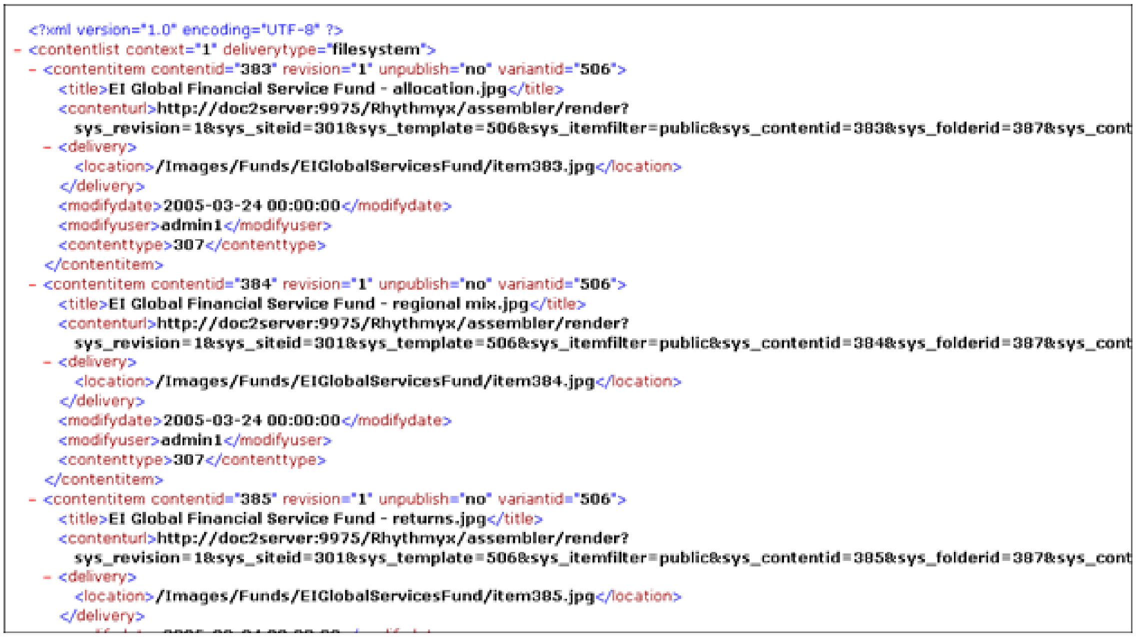
1 Log in to Rhythmyx Content Explorer and go to the Publishing Design tab.

2 Expand the Sites node. Expand the Enterprise Investments node. Expand the Editions node. Click on the EI_Full Edition

Content Explorer displays the EI_Full Edition editor.

3 Click on the Preview icon in the row of the rffEIFullNonbinary Content List

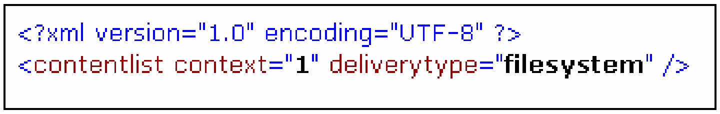
Content Explorer displays a preview of the output of the Content List.



```
  <?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://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" unpublish="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" unpublish="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 249: 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 250: 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

We have now registered or configured the basic publishing components of the implementation, and can publish the Editions we defined to verify that they will produce the output we expect. When testing in the development environment Editions are run manually but in a production environment, publishing an Edition is typically set up as a scheduled task. For details about implementing a scheduled task, see "Scheduling Publishing" in *Setting Up the Production Environment*.

In the following exercises, we will publish both the EI_Full Edtion and the ***EI_Incremental Edition*** (see page 340).

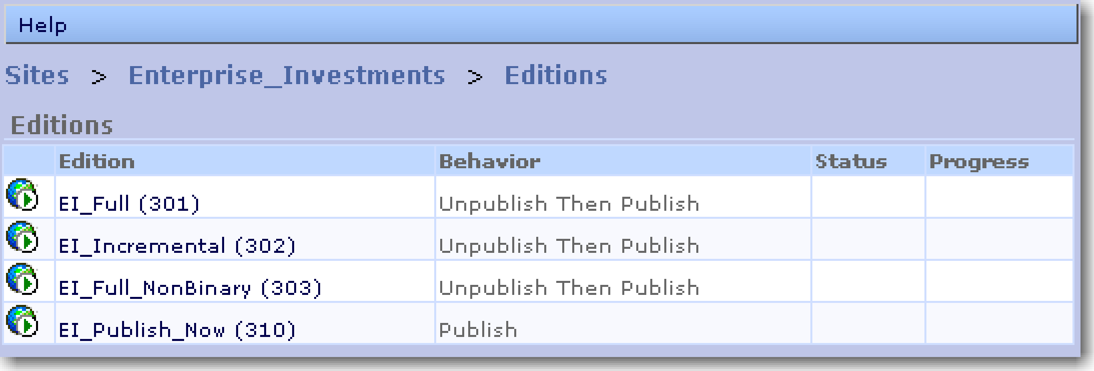
Publishing the Full Publish Edition

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.

While you use the Publishing Design tab to define the configurations of your Publishing implementation, you use the Publishing Runtime tab to run the Editions and access the logs for review.

To publish the Full Publish Edition:

1. Log in to Rhythmyx Content Explorer and click on the Publishing Runtime tab.
2. Expand the Sites node. Expand the EnterpriseInvestments Node. Click Editions.
3. Rhythmyx displays the Runtime Editions List.



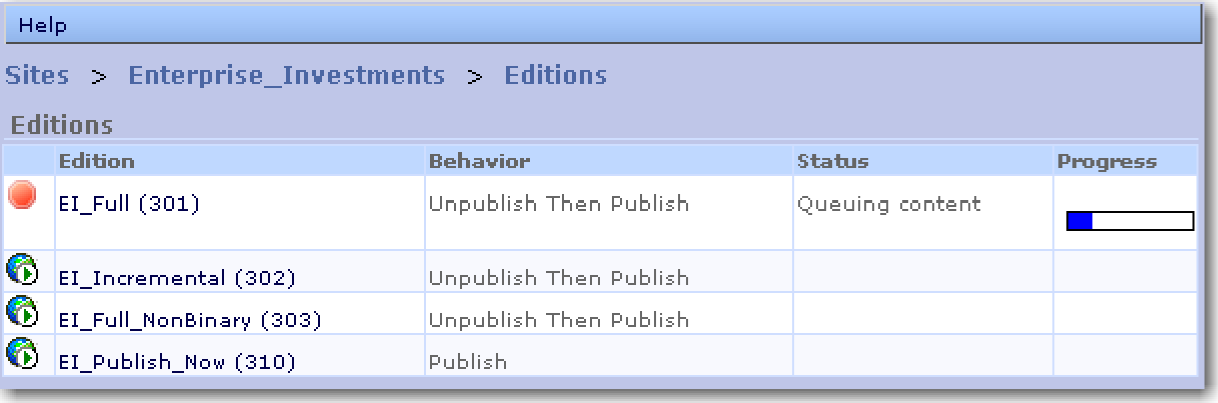
Sites > Enterprise_Investments > Editions

Editions

	Edition	Behavior	Status	Progress
▶	EI_Full (301)	Unpublish Then Publish		
▶	EI_Incremental (302)	Unpublish Then Publish		
▶	EI_Full_NonBinary (303)	Unpublish Then Publish		
▶	EI_Publish_Now (310)	Publish		

Figure 251: Runtime Edition List

4 You can run the Edition here by clicking the ▶ icon in the row of the EI_Full Edition:



Sites > Enterprise_Investments > Editions

Editions

	Edition	Behavior	Status	Progress
●	EI_Full (301)	Unpublish Then Publish	Queuing content	
▶	EI_Incremental (302)	Unpublish Then Publish		
▶	EI_Full_NonBinary (303)	Unpublish Then Publish		
▶	EI_Publish_Now (310)	Publish		

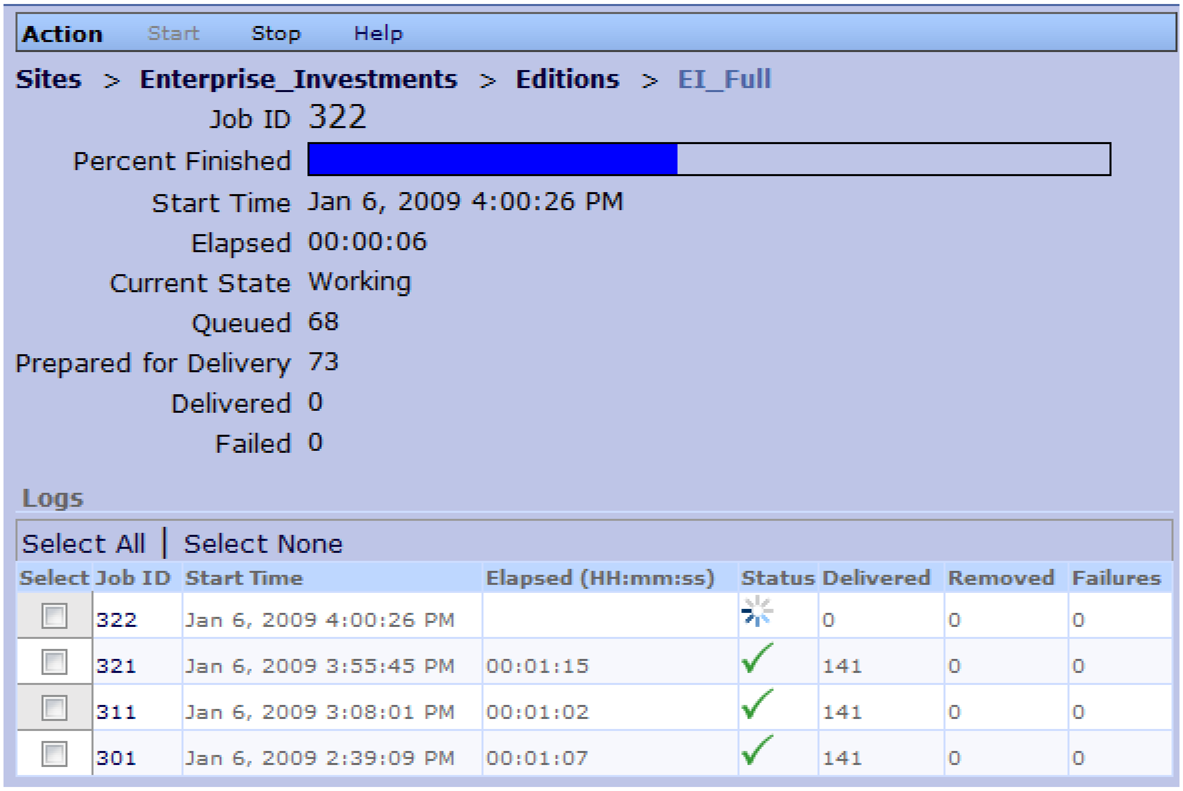
Figure 252: Runtime Edition List showing the EI_Full Edition Running

5 For more details about the Edition, click on the EI_Full link.

Rhythmyx displays the Runtime Edition page.

6 To run the Edition, in the Menu bar, click Start.

Rhythmyx displays detailed information about the running edition, including a progress bar.



Sites > Enterprise_Investments > Editions > EI_Full

Job ID 322

Percent Finished

Start Time Jan 6, 2009 4:00:26 PM

Elapsed 00:00:06

Current State Working

Queued 68

Prepared for Delivery 73

Delivered 0

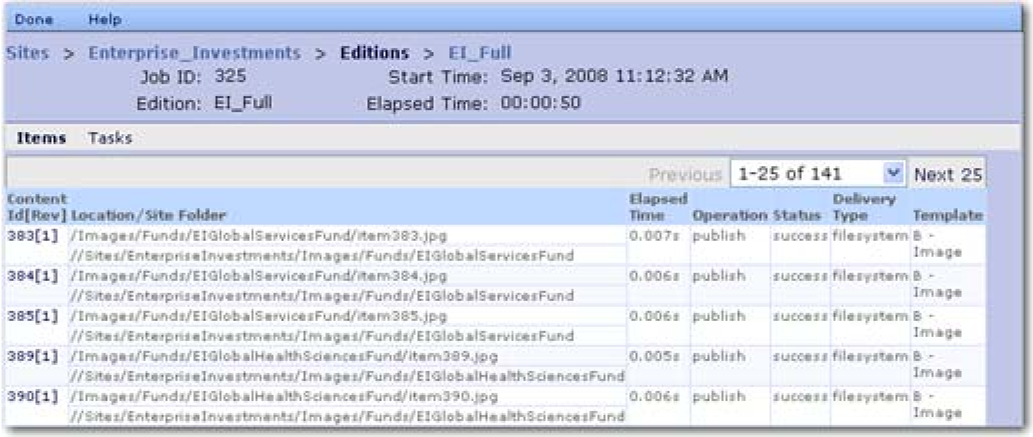
Failed 0

Logs

Select	Job ID	Start Time	Elapsed (HH:mm:ss)	Status	Delivered	Removed	Failures
□	322	Jan 6, 2009 4:00:26 PM			0	0	0
□	321	Jan 6, 2009 3:55:45 PM	00:01:15	✓	141	0	0
□	311	Jan 6, 2009 3:08:01 PM	00:01:02	✓	141	0	0
□	301	Jan 6, 2009 2:39:09 PM	00:01:07	✓	141	0	0

Figure 253: Edition Runtime showing status Information during a publishing run

A log for the new Editions is created and added to the Edition Runtime page as well. When processing of the Edition is complete, you can access the log by clicking on its Job ID.

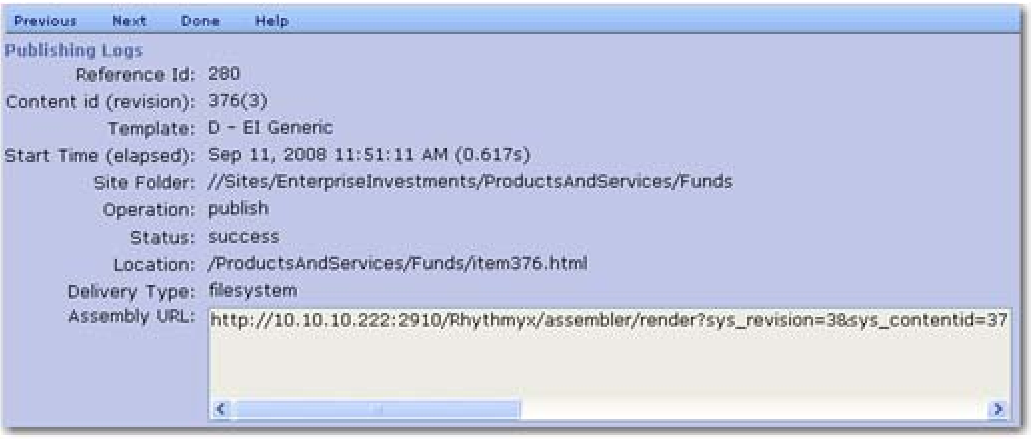


Job ID: 325
Start Time: Sep 3, 2008 11:12:32 AM
Edition: EI_Full
Elapsed Time: 00:00:50

Content Id[Rev]	Location/Site Folder	Elapsed Time	Operation	Status	Delivery Type	Template
383[1]	/Images/Funds/EIGlobalServicesFund/item383.jpg //Sites/EnterpriseInvestments/Images/Funds/EIGlobalServicesFund	0.007s	publish	success	filesystem	B - Image
384[1]	/Images/Funds/EIGlobalServicesFund/item384.jpg //Sites/EnterpriseInvestments/Images/Funds/EIGlobalServicesFund	0.006s	publish	success	filesystem	B - Image
385[1]	/Images/Funds/EIGlobalServicesFund/item385.jpg //Sites/EnterpriseInvestments/Images/Funds/EIGlobalServicesFund	0.006s	publish	success	filesystem	B - Image
389[1]	/Images/Funds/EIGlobalHealthSciencesFund/item389.jpg //Sites/EnterpriseInvestments/Images/Funds/EIGlobalHealthSciencesFund	0.005s	publish	success	filesystem	B - Image
390[1]	/Images/Funds/EIGlobalHealthSciencesFund/item390.jpg //Sites/EnterpriseInvestments/Images/Funds/EIGlobalHealthSciencesFund	0.006s	publish	success	filesystem	B - Image

Figure 254: Publishing Job Log View

Using this view, you can review the overall success of the publishing run of the Edition. To view details about a particular Content Item, click on the Content ID link. Rhythmyx returns the Published Item Detail page with detailed information about the published Content Item.



Reference Id: 280
Content id (revision): 376(3)
Template: D - EI Generic
Start Time (elapsed): Sep 11, 2008 11:51:11 AM (0.617s)
Site Folder: //Sites/EnterpriseInvestments/ProductsAndServices/Funds
Operation: publish
Status: success
Location: /ProductsAndServices/Funds/item376.html
Delivery Type: filesystem
Assembly URL: http://10.10.10.222:2910/Rhythmyx/assembler/render?sys_revision=3&sys_contentid=37

Figure 255: PublishedI tem Details

Publishing the Incremental Edition

For the purpose of this exercise, we assume that all of the Content Items on the Enterprise Investments Site are in a Public State.

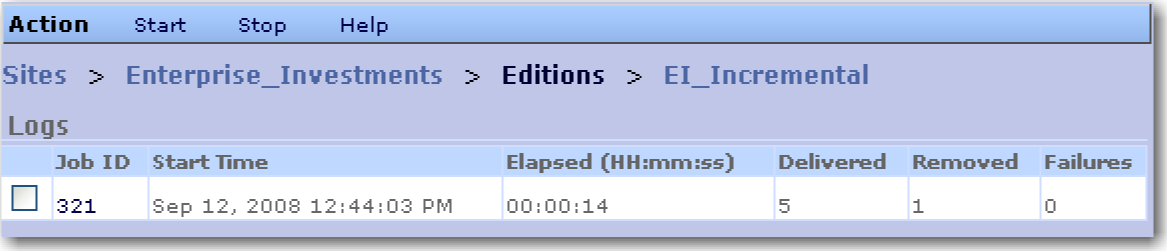
To see results when you publish your Incremental Edition, you must modify Content Items that are in a public State or add some new Content Items and move them to a public State.

In this example, assume the following changes have been made:

- A new Press Release Content Item has been created in the folder Sites/EnterpriseInvestments/AboutEnterpriseInvestments/Press Releases/2008 with the Content ID 703 and has been Transitioned to Public.

- The Content Item About Enterprise Investments Generic Content Item (Content ID 335) has been modified.
- The EI Reinsurance Generic Content Item (Content ID 406) has been Transitioned to the Archive State.

Follow the same steps as when publishing the rffEIFull Edition. When the Edition is complete, check the log. In this case, only five Content Items have been published and one was unpublished.



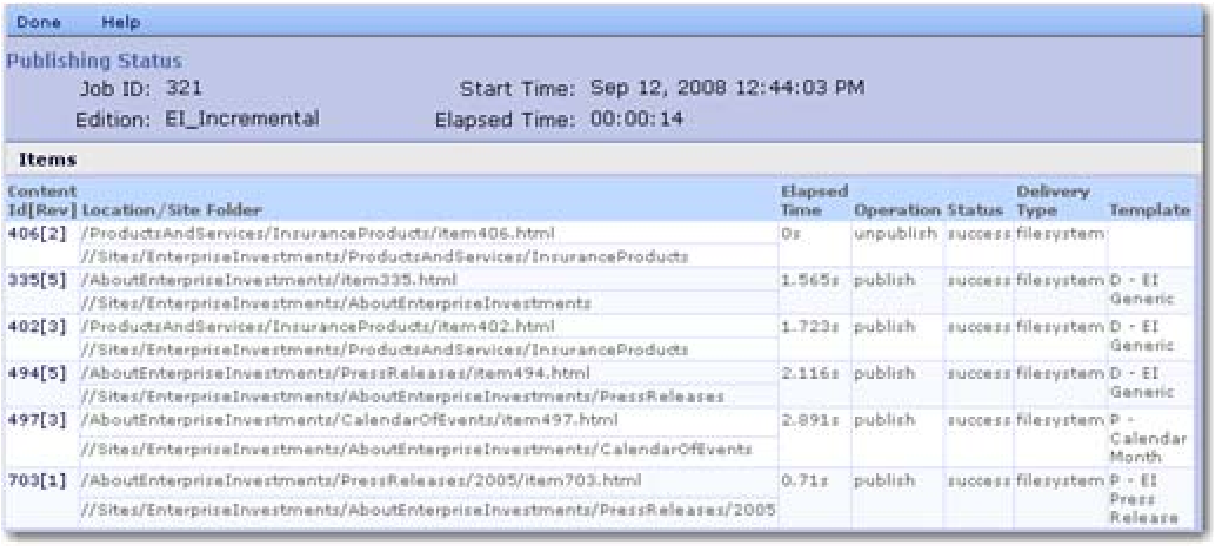
Sites > Enterprise_Investments > Editions > EI_Incremental

Logs

	Job ID	Start Time	Elapsed (HH:mm:ss)	Delivered	Removed	Failures
□	321	Sep 12, 2008 12:44:03 PM	00:00:14	5	1	0

Figure 256: Incremental Edition Log Summary showing five Content Items published and one Content Item unpublished

The log details which Content Items were processed:



Publishing Status

Job ID: 321 Start Time: Sep 12, 2008 12:44:03 PM

Edition: EI_Incremental Elapsed Time: 00:00:14

Items

Content Id[Rev]	Location/Site Folder	Elapsed Time	Operation	Status	Delivery Type	Template
406[2]	/ProductsAndServices/InsuranceProducts/item406.html //Sites/EnterpriseInvestments/ProductsAndServices/InsuranceProducts	0s	unpublish	success	filesystem	
335[5]	/AboutEnterpriseInvestments/item335.html //Sites/EnterpriseInvestments/AboutEnterpriseInvestments	1.565s	publish	success	filesystem	D - EI Generic
402[3]	/ProductsAndServices/InsuranceProducts/item402.html //Sites/EnterpriseInvestments/ProductsAndServices/InsuranceProducts	1.723s	publish	success	filesystem	D - EI Generic
494[5]	/AboutEnterpriseInvestments/PressReleases/item494.html //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/PressReleases	2.116s	publish	success	filesystem	D - EI Generic
497[3]	/AboutEnterpriseInvestments/CalendarOfEvents/item497.html //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/CalendarOfEvents	2.891s	publish	success	filesystem	P - Calendar Month
703[1]	/AboutEnterpriseInvestments/PressReleases/2005/item703.html //Sites/EnterpriseInvestments/AboutEnterpriseInvestments/PressReleases/2005	0.71s	publish	success	filesystem	P - EI Press Release

Figure 257: Detailed log of the incremental publishing run

The Content Item that was moved to archive was unpublished (ID 406). The new Content Item (ID 703) was published and the modified Content Item (ID 335) was republished. Two additional Content Items (IDs 494 and 497) were republished because they included automated indices (all automated indices are always published or republished to ensure that any modifications are included in the index).

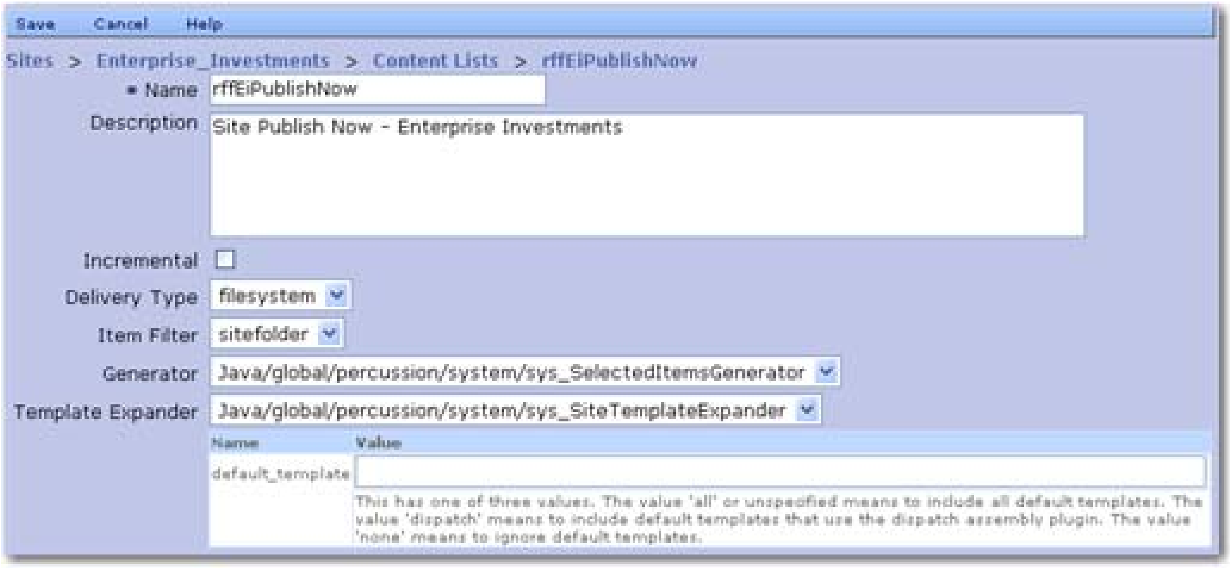
Implementing Demand Publishing (Publish Now)

Demand Publishing allows users to publish individual Content Items at their discretion. A Content Item must be public to be eligible to be published using Demand Publishing. Users can initiate Demand Publishing of a Content Item in two ways:

- In Active Assembly, choose *Tools* > *Publish Now*. Only the root Content Item in Outline View is eligible to be published using this option.
- In Content Explorer, select the Content Item you want to publish, right-click, and from the popup menu, choose *Publish Now*.

Publish Now processing looks for an Edition that includes only one Content List, which uses the sys_SelectedItemsGenerator. To implement Demand Publishing for a Site

- Create a Content List that uses the sys_SelectedItemsGenerator. The following graphic illustrates the configuration of the rffEiPublishNow Content List:



Sites > Enterprise_Investments > Content Lists > rffEiPublishNow

* Name: rffEiPublishNow

Description: Site Publish Now - Enterprise Investments

Incremental: □

Delivery Type: filesystem

Item Filter: sitefolder

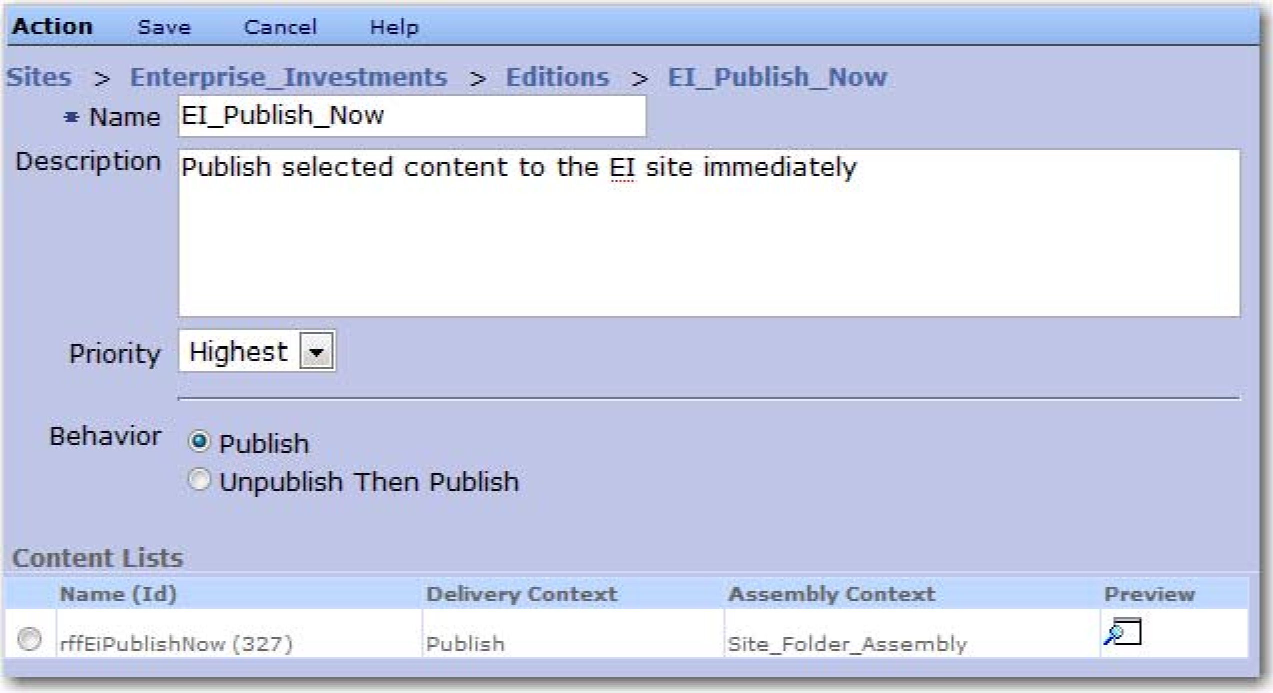
Generator: Java/global/percussion/system/sys_SelectedItemsGenerator

Template Expander: Java/global/percussion/system/sys_SiteTemplateExpander

Name	Value
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.

Figure 258: Publish Now Content List

- Create an Edition that includes the Content List described in the previous bullet point. No other Content Lists should be included in the Edition. In the Priority drop list, choose *Highest* radio button; Demand Editions typically have a higher priority (either High or Highest) and interrupt processing of other Editions. For Behavior, select the Publish radio button; the intent of Demand Publishing is to publish the specified Content Item; no unpublishing processing should be performed. The following graphic illustrates the configuration of the EI_Publish_Now Edition.



* Name: EI_Publish_Now

Description: Publish selected content to the EI site immediately

Priority: Highest

Behavior: ● Publish ○ Unpublish Then Publish

Content Lists

	Name (Id)	Delivery Context	Assembly Context	Preview
○	rffEiPublishNow (327)	Publish	Site_Folder_Assembly	🔍

Figure 259: Publish Now Edition

Setting Up the Corporate Investments Site

Once you have set up one Site, you can quickly set up additional Sites by copying and modifying the configurations of the original Site. In the following exercises, we will create the CorporateInvestments Site by copying and modifying the configurations of the EnterpriseInvestments Site.

Copying a Site is a two-phase process:

- copying the Site Folder Structure, or the Site registration, or both; and
- copying the Editions and Content Lists.

Copying a Site in the Content Tab

If you want to create a Site that uses the Folder structure of an existing Site, copy the Site Folder in the Content tab of Content Explorer. The action launches a wizard that allows you to copy the Site registration of the copied Site as well. In this exercise, we will create the CorporateInvestments Site by copying the EnterpriseInvestments Site, including the Site registration.

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

To copy the EnterpriseInvestments Site Folder structure and Site registration:

1. Change to the Content tab of Content Explorer.
2. Select the EnterpriseInvestments Site. Right-click on from the popup menu choose *Copy*.

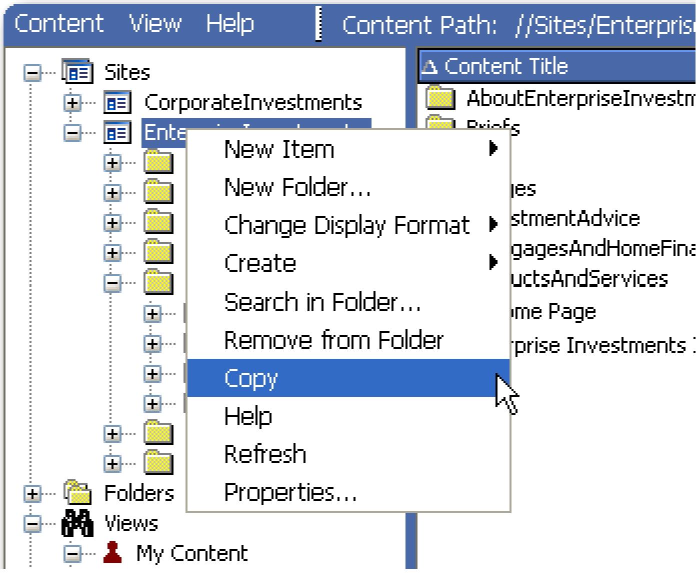


Figure 260: Copying the EnterpriseInvestments Site in the Content tab

3 Select the Sites root node. Right-click and from the popup menu choose *Paste > As New Copy*.

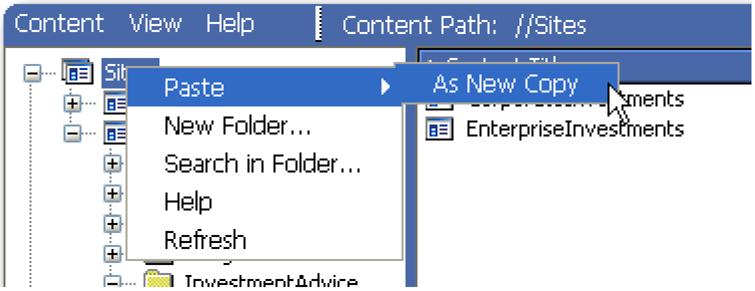
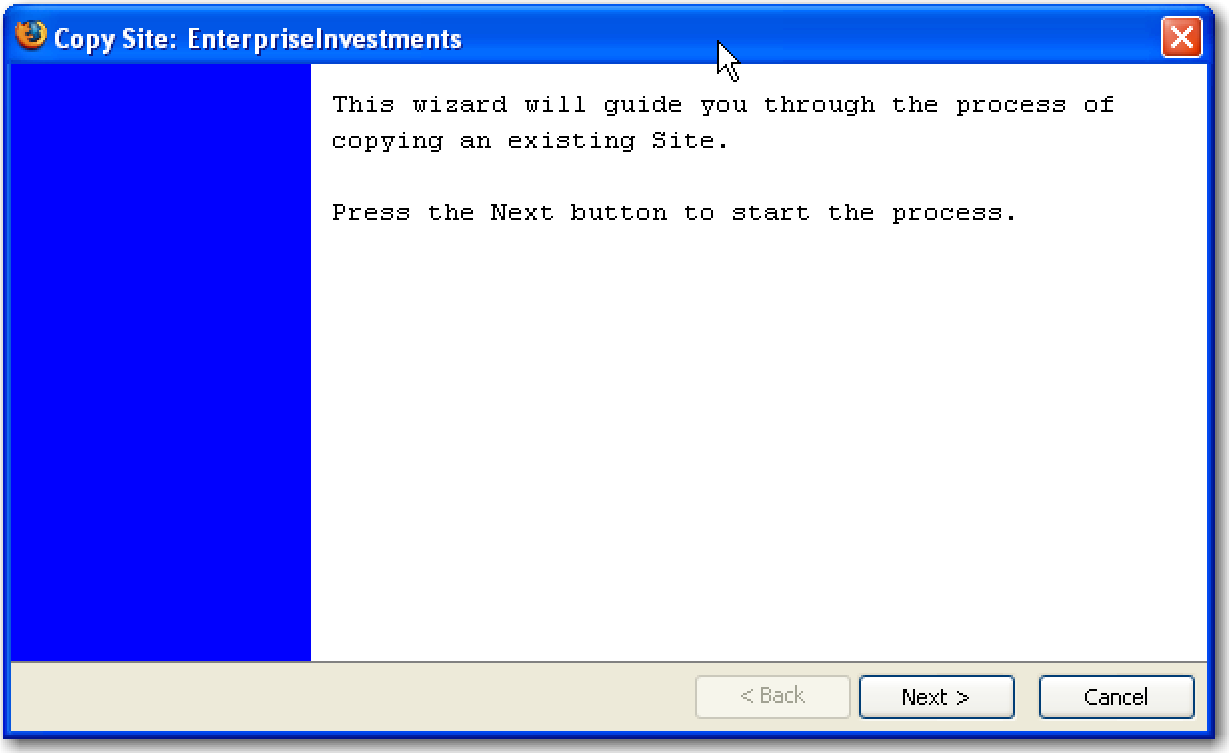


Figure 261: Pasting the Site Folder to create a new Site

Rhythmyx launches the Copy Site wizard.



This wizard will guide you through the process of copying an existing Site.

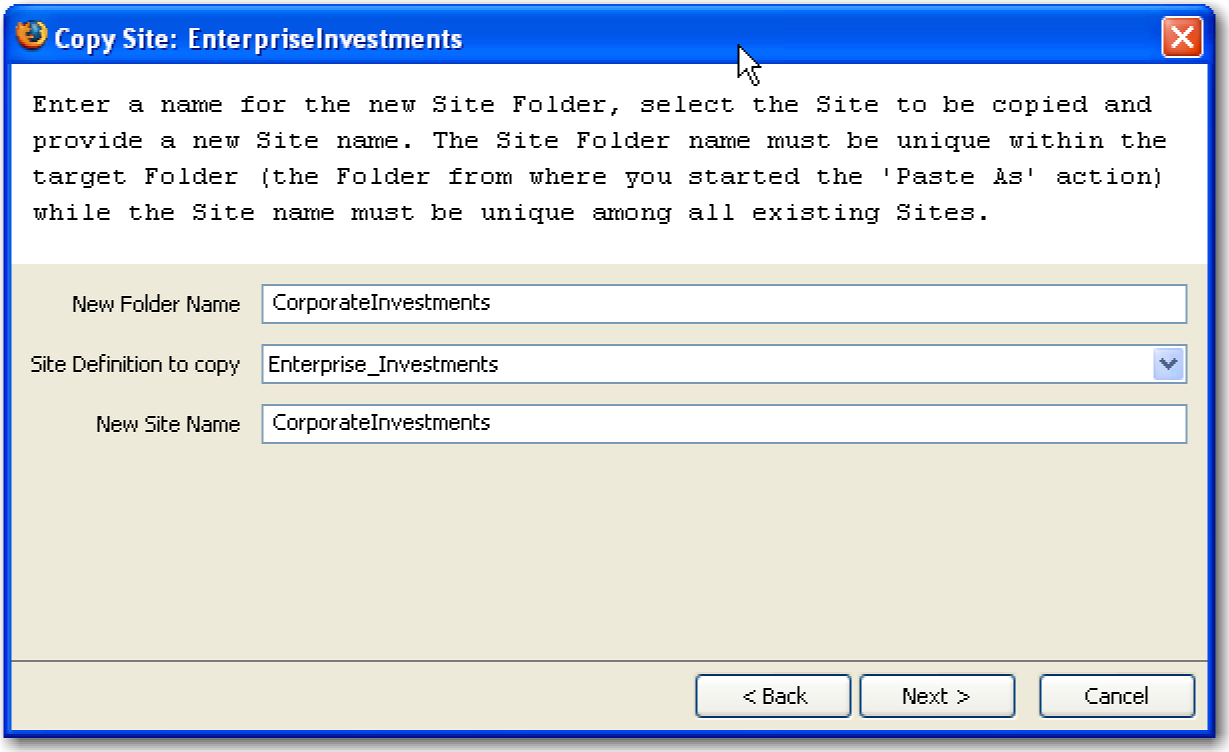
Press the Next button to start the process.

Figure 262: Copy Site Wizard Intro Page

4 Click the [**Next**] button.

Rhythmyx displays the Copy Site Name dialog.

5 In the New Folder Name and New Site Name fields, enter *CorporateInvestments*. In the Site Definition to Copy field, be sure *EnterpriseInvestments* is selected.



Copy Site: EnterpriseInvestments

Enter a name for the new Site Folder, select the Site to be copied and provide a new Site name. The Site Folder name must be unique within the target Folder (the Folder from where you started the 'Paste As' action) while the Site name must be unique among all existing Sites.

New Folder Name: CorporateInvestments

Site Definition to copy: Enterprise_Investments

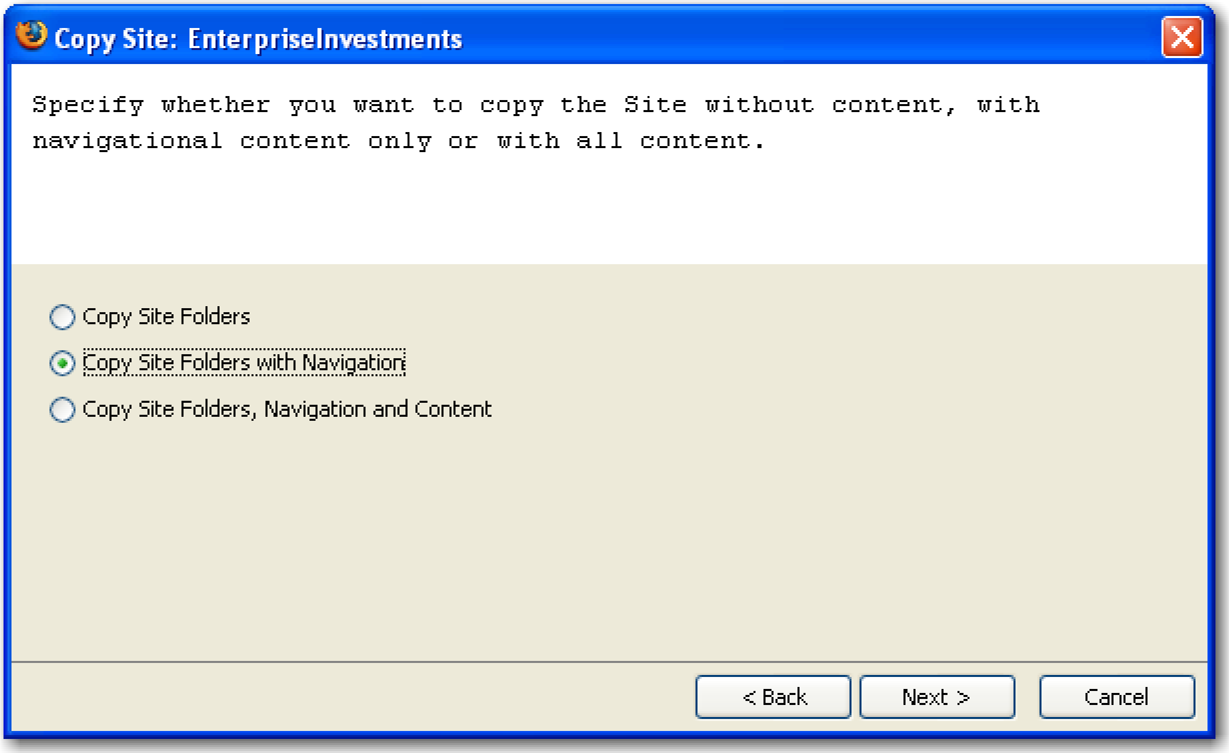
New Site Name: CorporateInvestments

Figure 263: Specifying the Corporate Investments Site on the Copy Site Wizard. Note that the EnterpriseInvestments Site is specified as the Site Definition to Copy.

6 Click the [**Next**] button.

Rhythmyx displays the Copy Site Folders dialog. On this dialog you can choose to copy only the Site Folder structure, the Site Folder structure with navigation (Navtree, Navons, and associated Managed Navigation Content Items) or the Site Folder structure with all Content Items. For the purposes of this exercise, we will copy the Site Folder structure and the navigation.

7 Select the Copy Site Folders with Navigation radio button.



Specify whether you want to copy the Site without content, with navigational content only or with all content.

○ Copy Site Folders

● Copy Site Folders with Navigation

○ Copy Site Folders, Navigation and Content

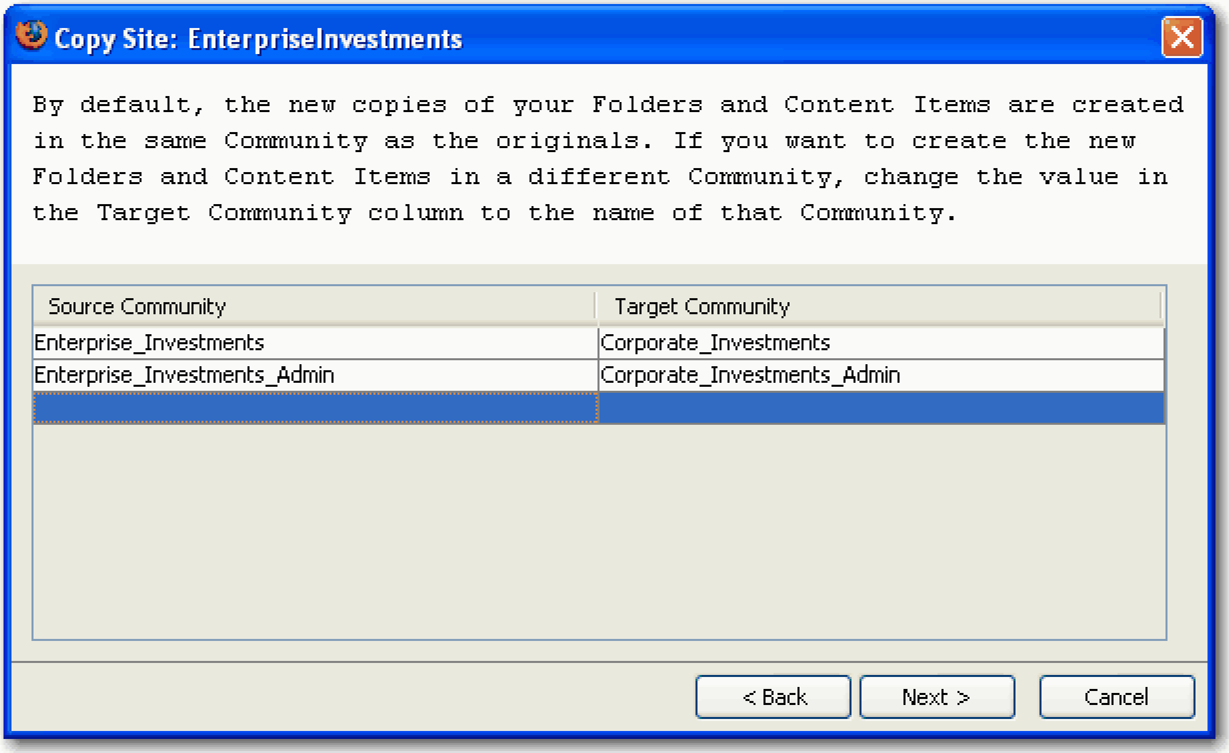
Figure 264: Copy Site Wizard Folders dialog with option selected to copy the Site Folder structure and the navigation

8 Click the [**Next**] button.

Rhythmyx displays the Copy Site Wizard Community dialog. Use this dialog to specify the Communities in which copied Content Items will be created. The default option is to create the New Copy Content Items in the same Community as the Owner Content Item, but you can create the New Copy Content Items in different Communities specifying a different Target Community for each Source Community.

In our case, we want to create any Content Item copied from the Enterprise_Investments Community to be created in the Corporate_Investments Community, and any Content Item copied from the Enterprise_Investments_Admin Community to be created in the Corporate_Investments_Admin Community.

9 In the row of the Enterprise_Investments Community, click the cell in the Target Community column and from the drop list choose Corporate_Investments. In the row of the Enterprise_Investments_Admin Community, click in the Target Community column and from the drop list choose Corporate_Investments_Admin.



Copy Site: EnterpriseInvestments

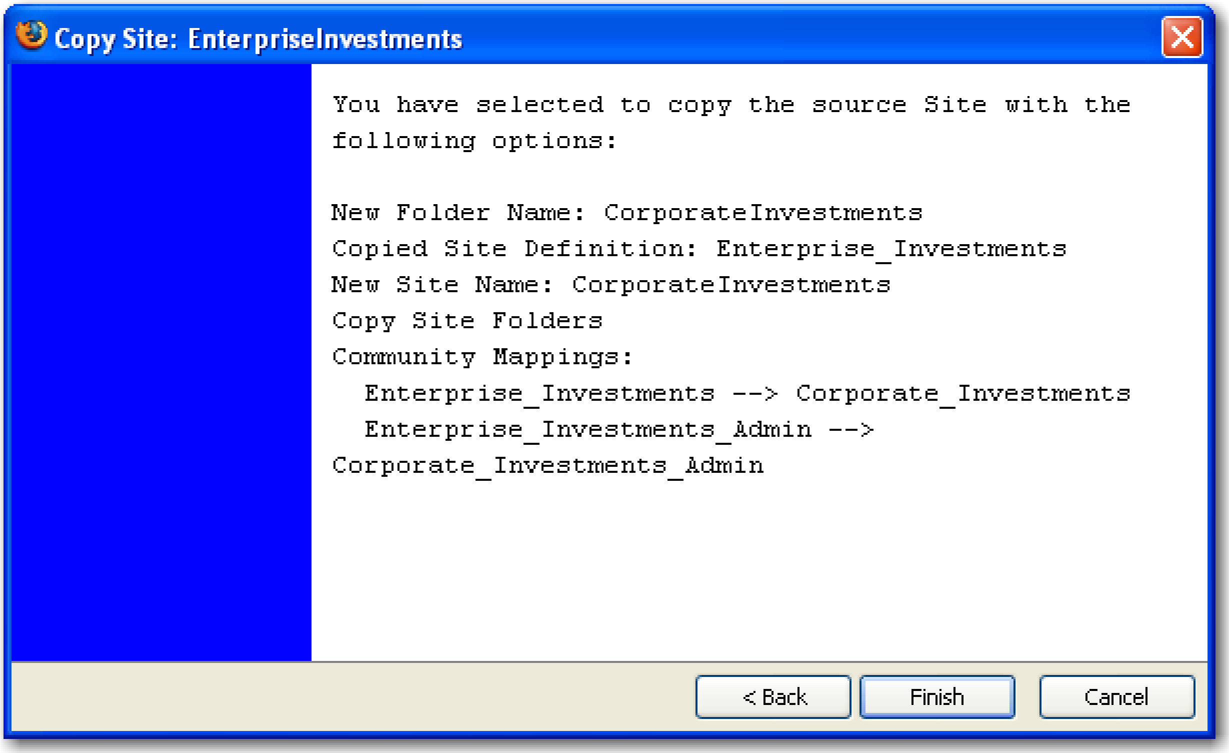
By default, the new copies of your Folders and Content Items are created in the same Community as the originals. If you want to create the new Folders and Content Items in a different Community, change the value in the Target Community column to the name of that Community.

Source Community	Target Community
Enterprise_Investments	Corporate_Investments
Enterprise_Investments_Admin	Corporate_Investments_Admin

Figure 265: Copy Site Community Mapping dialog showing Enterprise Investments Communities mapped to Corporate Investment Communities

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

Rhythmyx displays the Copy Site Wizard summary dialog



Copy Site: EnterpriseInvestments

```
You have selected to copy the source Site with the
following options:

New Folder Name: CorporateInvestments
Copied Site Definition: Enterprise_Investments
New Site Name: CorporateInvestments
Copy Site Folders
Community Mappings:
  Enterprise_Investments --> Corporate_Investments
  Enterprise_Investments_Admin -->
Corporate_Investments_Admin
```

Figure 266: Copy Site Wizard summary dialog showing the specifications for creating the Corporate Investments Site

11 Click the [**Next**] button.

Rhythmyx copies the Site, creating the Site Folder structure you copied and the Content Items you specified to copy. The Site registration you specified is also copied. This process may take a few minutes depending on the options you specified in the wizard.

12 Fill out rest of topic when copy site wizard works correctly.

Copying a Site Registration

In some cases, you may want to create a different Folder Structure for a specific Site. In that case, you may prefer to simply copy the Site registration and create the Site Folder, Subfolders, and navigation separate. In this exercise, we will copy only the EnterpriseInvestements Site registration to create the CorporateInvestments Site registration.

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

To copy the EnterpriseInvestments Site registration:

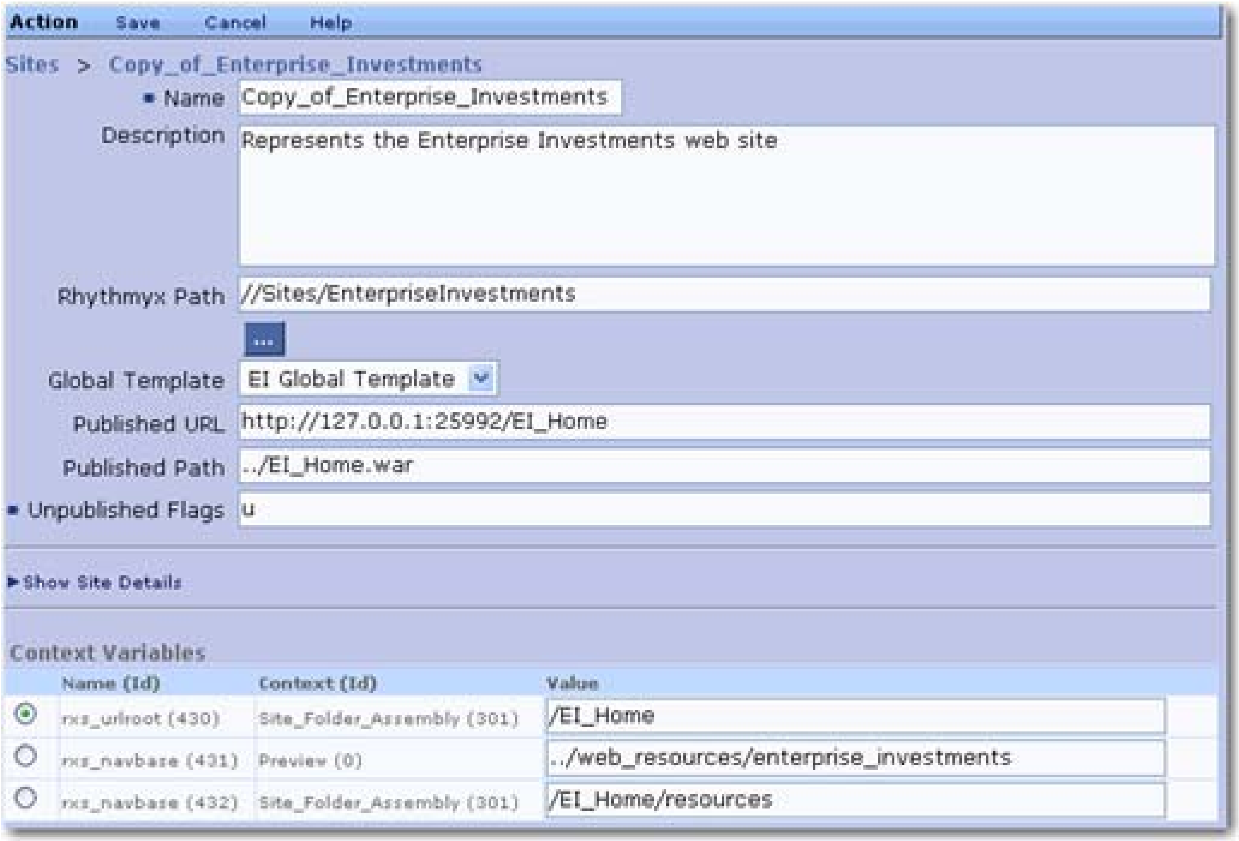
1 On the Publishing Design tab, click Sites.

Rhythmyx displays the Site List page.

2 Select the radio button in the EnterpriseInvestments row.

3 In the Menu bar, choose *Action* > *Copy Selected Site*.

4 Rhythmyx copies the Site registration and opens the Site Editor with the copied Site registration.



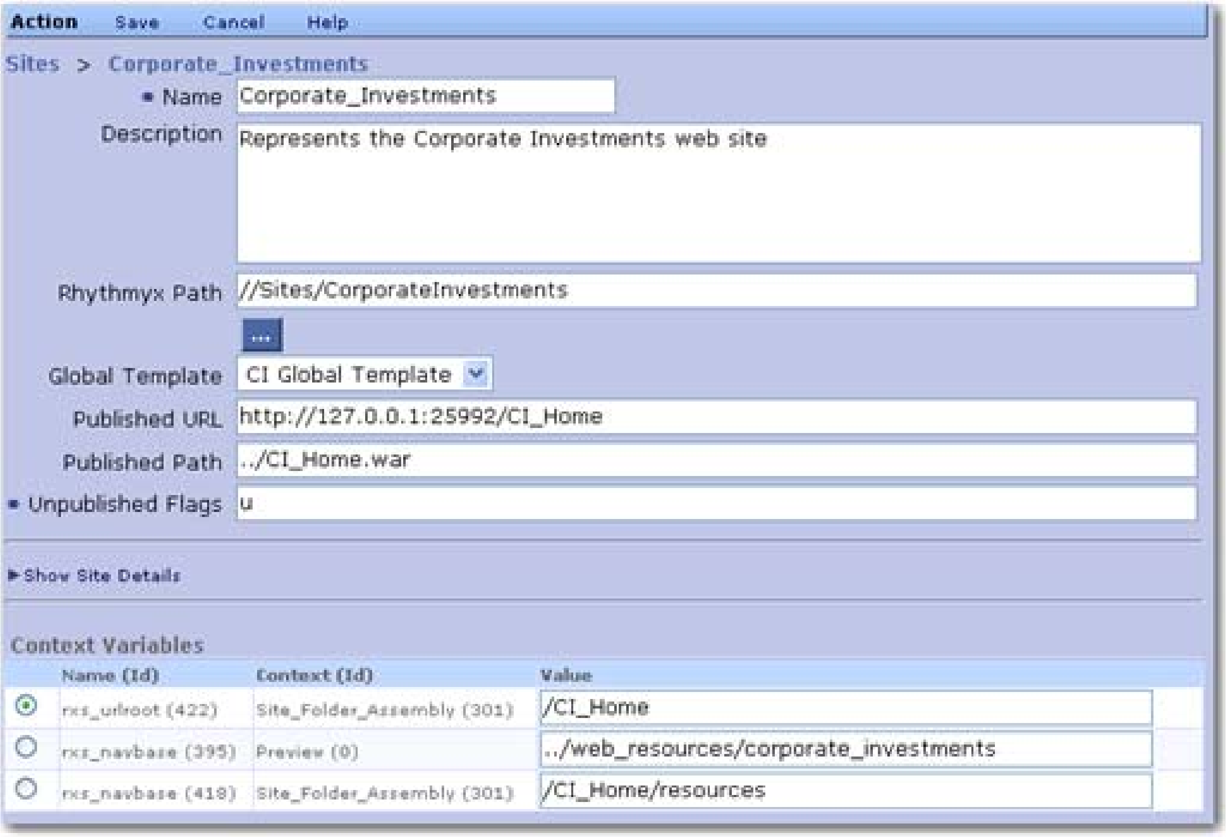
■ Name	Copy_of_Enterprise_Investments
Description	Represents the Enterprise Investments web site
Rhythmyx Path	//Sites/EnterpriseInvestments
Global Template	EI Global Template
Published URL	http://127.0.0.1:25992/EI_Home
Published Path	../EI_Home.war
■ Unpublished Flags	u

Context Variables

	Name (Id)	Context (Id)	Value
●	rxs_urlroot (430)	Site_Folder_Assembly (301)	/EI_Home
○	rxs_navbase (431)	Preview (0)	../web_resources/enterprise_investments
○	rxs_navbase (432)	Site_Folder_Assembly (301)	/EI_Home/resources

Figure 267: Copy of Enterprise Investments Site registration

5 Change the values to match the specifications of the Corporate Investments Site. Note that you can edit the value of a Context Variable within the Site Editor. The following graphic illustrations the Corporate Investments Site registration.



* Name: Corporate_Investments

Description: Represents the Corporate Investments web site

Rhythmyx Path: //Sites/CorporateInvestments

Global Template: CI Global Template

Published URL: http://127.0.0.1:25992/CI_Home

Published Path: ../CI_Home.war

* Unpublished Flags: u

Context Variables

	Name (Id)	Context (Id)	Value
●	rxs_urlroot (422)	Site_Folder_Assembly (301)	/CI_Home
○	rxs_navbase (395)	Preview (0)	../web_resources/corporate_investments
○	rxs_navbase (418)	Site_Folder_Assembly (301)	/CI_Home/resources

Figure 268: Corporate Investments Site Registration

6 In the Menu bar, click *Save*.

Copying Editions and Content Lists

Once you have created your new Site, you need to create the Editions and Content Lists used to publish the content of the Site. Editions and Content Lists can be copied from an existing Site to a new Site. In this exercise, we will copy the EI_Full Edition, with its Content Lists, to the CorporateInvestments Site that we created.

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

To copy the EI_Full Edition and its Content Lists:

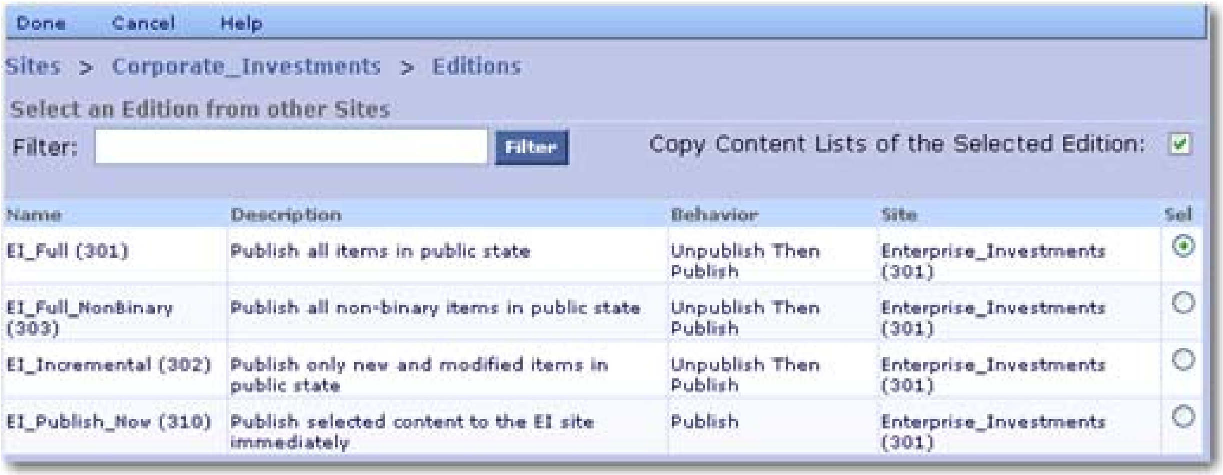
1 In the Publishing Design tab, expand the Sites node, expand the CorporateInvestments node and click Editions.

Rhythmyx displays the Edition List page. Since we have not added Editions to this Site yet, it will list no Editions.

2 In the Menu bar, choose *Action* > *Copy Edition From Other Site*.

Rhythmyx displays the Select Edition From Other Site page.

3 Select the radio button in the row of the EI_Full Edition. Check the Copy Content Lists of the Selected Edition box.



Select an Edition from other Sites

Filter:

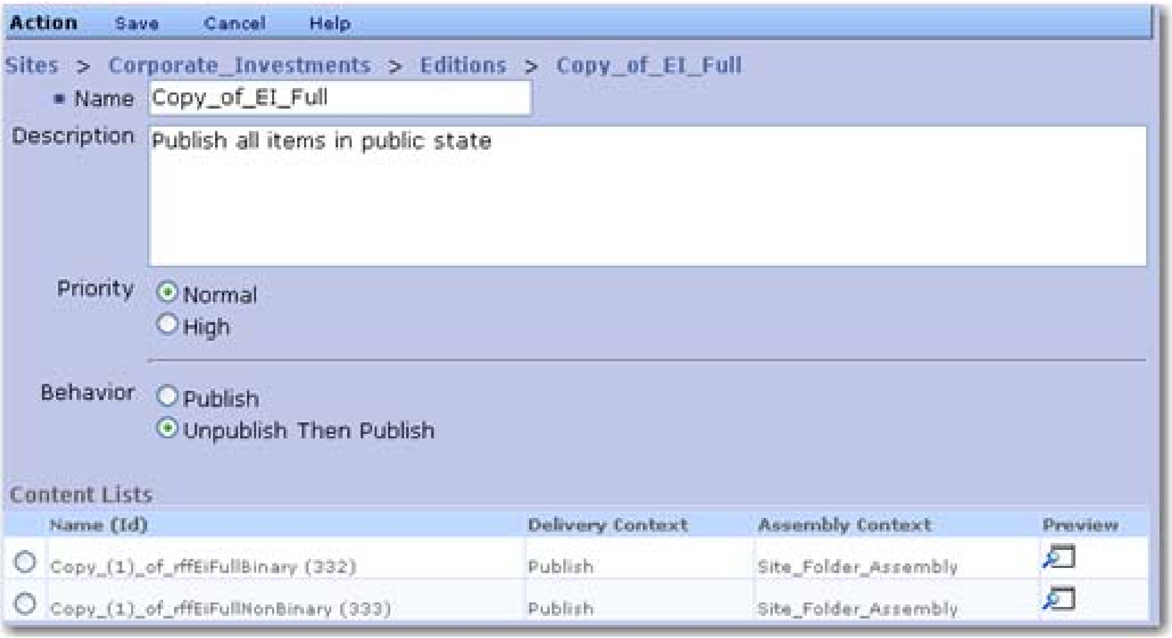
Copy Content Lists of the Selected Edition: ☑

Name	Description	Behavior	Site	Sel
EI_Full (301)	Publish all items in public state	Unpublish Then Publish	Enterprise_Investments (301)	●
EI_Full_NonBinary (303)	Publish all non-binary items in public state	Unpublish Then Publish	Enterprise_Investments (301)	○
EI_Incremental (302)	Publish only new and modified items in public state	Unpublish Then Publish	Enterprise_Investments (301)	○
EI_Publish_Now (310)	Publish selected content to the EI site immediately	Publish	Enterprise_Investments (301)	○

Figure 269: Copying the EI_Full Edition and it's Content Lists

4 In the Menu bar, click *Done*.

Rhythmyx copies the EI_Full Edition and its Content Lists.



* Name: Copy_of_EI_Full

Description: Publish all items in public state

Priority: ● Normal ○ High

Behavior: ○ Publish ● Unpublish Then Publish

Content Lists

	Name (Id)	Delivery Context	Assembly Context	Preview
○	Copy_(1)_of_rffEiFullBinary (332)	Publish	Site_Folder_Assembly	🔍
○	Copy_(1)_of_rffEiFullNonBinary (333)	Publish	Site_Folder_Assembly	🔍

Figure 270: EI_FullEdition and Content Lists after being copied to the CorporateInvestments Site

5 Modify the data for the Edition and the Content Lists to match the specifications required for the CorporateInvestments Site.

Setting Up Publishing to a Local Web Server

In the FastForward sample data we have used to illustrate implementation procedures, we have defined publishing to the FastForward Web applications installed with Rhythmyx. Percussion Software recommends that in your implementation environment you publish to a robust Web server. A common configuration is to publish to a Web server installed on the same machine as the Rhythmyx server. In this topic we will illustrate the modifications to the FastFoward configurations we have used so far to publish to the two most common production Web servers: Microsoft Internet Information Server (IIS) and the Apache Web Server. To publish to a local Web server, we must modify the Published URL and Published Path fields in the Site registration.

We will assume that these Web servers are installed using the defaults and that a virtual root named EI_Home, has been set up for the EnterpriseInvestments Site. We will also assume that you have copied the EI_Home/resources directory from the Rhythmyx installation to the EI_Home directory on the Web server and that the EI_Home /resources directory contains all of the CSS, JavaScript, and static images required to support the EnterpriseInvestments Site. Therefore we do not have to modify the Context Variables.

Assuming default installations, the modification to the Published URL field is the same in both Web servers. The default port for Web servers is port 80. Since we're assuming a virtual root that matches the root used in Rhythmyx (EI_Home), only the port needs to change in this field. Change the value from

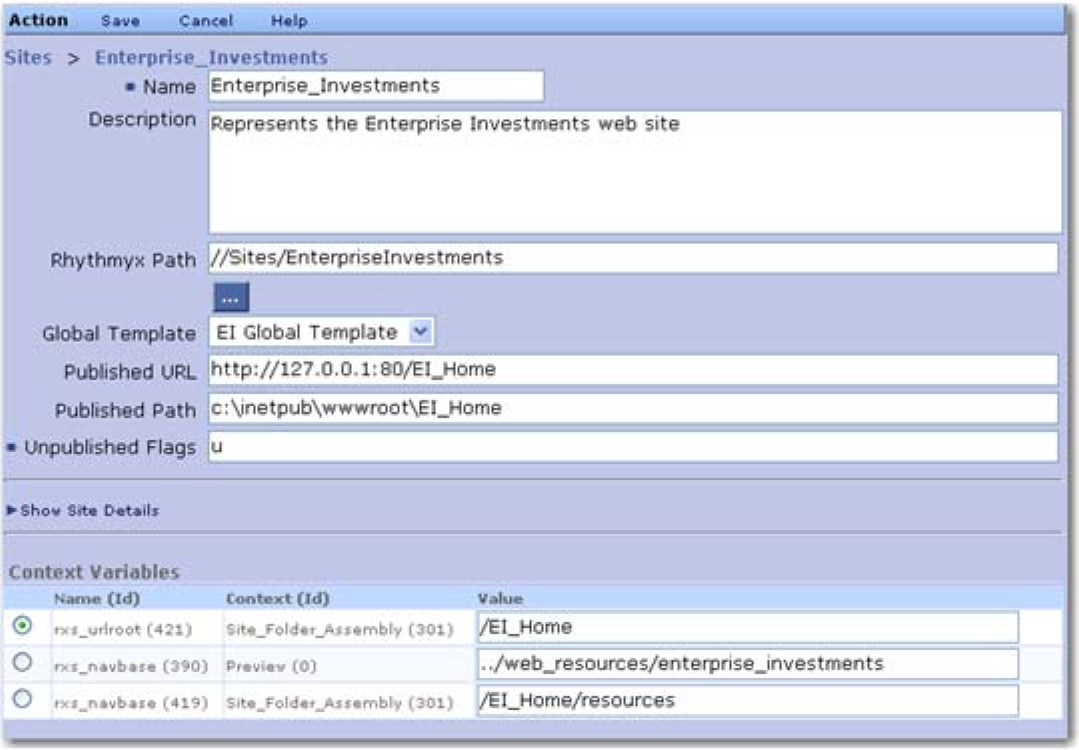
```
http://127.0.0.1:9992/EI_Home
```

to

```
http://127.0.0.1:80/EI_Home
```

The value of the **Published Path** will be different for each Web Server. The value in this field defines the path to the location where the assembled content will be delivered. When using IIS, the virtual root is a subdirectory of the wwwroot directory. Thus the value of the **Published Path** field of the EnterpriseInvestments Site registration when using IIS would be:

```
c:\inetpub\wwwroot\EI_Home
```



Sites > Enterprise_Investments

* Name	Enterprise_Investments
Description	Represents the Enterprise Investments web site
Rhythmyx Path	//Sites/EnterpriseInvestments
Global Template	EI Global Template
Published URL	http://127.0.0.1:80/EI_Home
Published Path	c:\inetpub\wwwroot\EI_Home
* Unpublished Flags	u

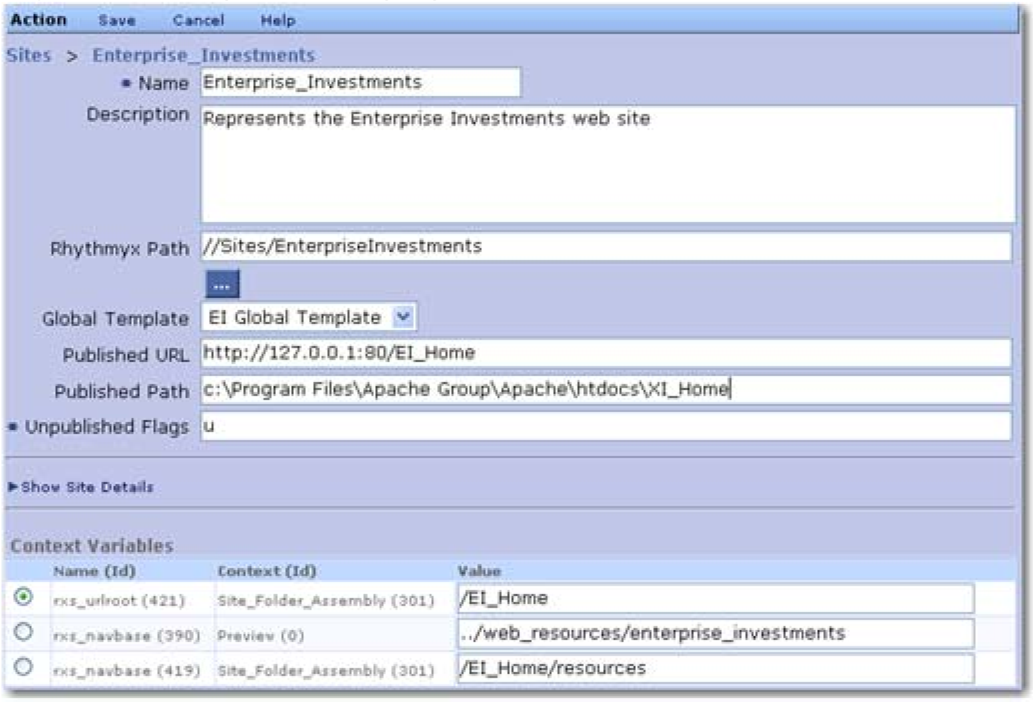
Context Variables

	Name (Id)	Context (Id)	Value
●	rxs_urlroot (421)	Site_Folder_Assembly (301)	/EI_Home
○	rxs_navbase (390)	Preview (0)	../web_resources/enterprise_investments
○	rxs_navbase (419)	Site_Folder_Assembly (301)	/EI_Home/resources

Figure 271: Site registration when publishing to IIS

When using an Apache Web Server, the value of the Published Path field must specify a subdirectory of Apache's default publishing root, `<Apacheroot>\htdocs`. Thus, the value of the Published Path field in this case would be

```
c:\Program Files\Apache Group\Apache\htdocs\XI_Home
```



* Name	Enterprise_Investments
Description	Represents the Enterprise Investments web site
Rhythmyx Path	//Sites/EnterpriseInvestments
Global Template	EI Global Template
Published URL	http://127.0.0.1:80/EI_Home
Published Path	c:\Program Files\Apache Group\Apache\htdocs\XI_Home
* Unpublished Flags	u

Context Variables

	Name (Id)	Context (Id)	Value
●	rxs_urlroot (421)	Site_Folder_Assembly (301)	/EI_Home
○	rxs_navbase (390)	Preview (0)	../web_resources/enterprise_investments
○	rxs_navbase (419)	Site_Folder_Assembly (301)	/EI_Home/resources

Figure 272: Site regsitration when publishing to Apache Web Server

Implementing FTP Delivery

Common practice is to run the Rhythmyx server on a separate machine from the production Web server. Since production Web servers are typically deployed in the demilitarized zone of your environment, file transfer protocol (FTP) is commonly used to deliver the published content to them. Rhythmyx can support both standard FTP and secure FTP.

In the following topics, we will set up delivery using both standard and secure FTP. Rhythmyx comes with standard Delivery Types for both forms of FTP delivery. These standard Delivery Types should meet the needs of most customers, and we will use them in our examples. If the standard Delivery Types do not meet your needs, you can define custom Delivery Types. For details, see the *Rhythmyx Technical Reference Manual*.

We will assume that a virtual root has been defined on the Web server for the Site, and that the contents of the web_resources directory have been copied to the virtual root directory. You must also create a virtual directory on the FTP server that points to the Site's virtual root on the Web server.

For the purposes of this exercise, we will assume that the virtual directory the virtual root for Enterprise Investments has been created at wwwroot/EnterpriseInvestments and that a virtual directory called FastForwardFTP has been created on the FTP server pointing at that virtual root. We will assume that the user account for access to the FTP site is FastForwardFTP with a password of *FastForward*.

Setting up Standard FTP Delivery

Rhythmyx delivers assembled output to the location defined in your FTP server configuration. You need to create the virtual root on the target Web server and copy all static resources to this location. You must also create the virtual directory on the FTP server that points to the virtual root on the Web server.

For the purposes of this exercise, we will assume that the virtual directory for the EnterpriseInvestments Site has been created and that a virtual directory called FastForwardFTP has been created on the FTP server pointing to that virtual root. We will assume that the user account for access to the FTP server is FastForwardFTP with a password of FastForward.

To set up standard FTP Publishing,

1. Create a Site definition that specifies the FTP connection data.
2. Define Content Lists that use the FTP Delivery Type.
3. Define Editions that use the Content Lists defined in Step 2.

The following topics illustrate an example implementation based on a copy of the EnterpriseInvestments Site.

Defining a Site Registration for FTP Publishing

This first step in setting up delivery using FTP is to define a Site registration with FTP data. In this exercise, we will copy the EnterpriseInvestments Site registration and add the FTP data. We will rename the Site registration to "EnterpriseInvestmentsFTP".

We will assume the following data for the FTP connection:

- The IP address of the production Web server is 255.255.255.112.
- The FTP port of the production Web server is 81.
- The FTP user on the production Web server is FastForwardFTP.
- The FastForwardFTP user's password is FastForward.

We will assume that the remaining data will remain the same for the Enterprise Investments FTP Site.

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

To create the Site registration for FTP publishing:

1. In Content Explorer, select the Publishing Design tab.
2. In the Navigation bar, select the Sites node.

   Content Explorer displays the Sites List.
3. Select the Enterprise Investments Site.
4. In the Menu bar, choose *Action > Copy Selected Site*.

   Rhythmyx creates a copy of the EnterpriseInvestments Site and displays it in the Site editor.
5. The default name of the copied site is *Copy of EnterpriseInvestments*. Change the name to *EnterpriseInvestmentsFTP*.
6. For the purposes of this exercise, we will assume that the rest of the Site registration data remains the same.
7. To add FTP data:

   a) Click the Show Site details link.

   Content Explorer displays several additional fields on the Site editor, including FTP Server IP address, FTP server port, User name, and Password.

   b) In the FTP server IP address field, enter *255.255.255.112*.

   c) In the FTP port field, enter *81*.

   d) In the User name field, enter *FastForwardFTP*.

   e) In the Password field enter *FastForward*.
8. To save the Site registration, in the Menu bar, click [**Save**].

Defining FTP Editions and Content Lists

The second and third steps in setting up FTP delivery is to define Content Lists to use the FTP Delivery Type and Editions that include these Content Lists. The following procedure combines these two steps by copying the EI_Full Edition and its Content Lists to the EnterpriseInvestmentsFTP Site. We will change the name of the Edition to EI_Full_FTP, but will not change any other data. We will change the change the **Delivery Type** to *ftp* and rename the Content Lists to indicate that they use FTP, but we will not modify any other data.

To create the EI_Full_FTP Edition:

1. In Content Explorer, select the Publishing Design tab.
2. In the Navigation bar, expand the Sites, Node. Expand the EnterpriseInvestmentsFTP node. Click on the Editions link.

   Rhythmyx displays the Edition List.
3. In the Menu bar, choose *Action > Copy Edition From Other Site*.
4. Select the radio button in the row of the EI_Full Edition. Check the **Copy Content Lists of the Selected Edition** box.
5. In the Menu bar, click *Done*.

   Rhythmyx copies the Edition and Content Lists. When the processing is complete, Rhythmyx displays the Edition Editor.
6. Change the **Name** to *EI_Full_FTP*.
7. In the Menu bar, click *Save*.

   Rhythmyx saves the Edition and displays the Edition List.
8. Under the EnterpriseInvestmenstFTP node, click Content Lists.

   Rhythymyx displays the List Content Lists page.
9. Click Copy_of_rffEIFullNonbinary.

   Rhythmyx displays the Content List Editor for the Copy_of_rffEIFullNonbinary Content List.
10. Change the **Name** to *rffEIFullNonbinaryFTP*.
11. In the **Delivery Type** drop list, choose *ftp*.
12. In the Menu bar, click *Save*.
13. Repeat Steps 9 to 12 for the Copy_of_rffEIFullBinary Content List.

Setting Up Secure FTP (SFTP) Delivery

Rhythmyx delivers assembled output to the location defined in your SFTP server configuration. You need to create the virtual root on the target Web server and copy all static resources to this location. You must also create the virtual directory on the SFPT server that points to the virtual root on the Web server.

Rhythmyx is certified against the following SSH servers:

- Open SSH
  - Linux
  - Solaris
  - Windows
- freeSSHd
  - Windows only

Files delivered to a Windows environment are created with the default permissions of that environment. When delivered to a Unix environment, files are created with default permissions of *644*.

NOTE: If you want to specify different permissions for files delivered to a Unix environment, in the file `<Rhythmyxroot>/AppServer/server/rx/deploy/rxapp.ear/rxapp.war/WEB-INF/config/user/spring/publisher-beans.xml` modify the SFTP Delivery Handler configuration (`bean id="sys_sftpDeliveryHandler"`) by adding the bean property `umask`, with a value of the umask of the permissions you want to set. The umask is the seven's complement of the desired file permissions; in other words, the value of each digit in the umask is the integer value between 0 and 7 that, when added to the same digit in the file permissions, results in a sum of 7. (File permissions and the umask are both specified in octal.) For example, if you want file permissions of 740, specify the value of the umask property as *037*. If this property is not specified, Rhythmyx assumes a default umask of 133, resulting in the default file permissions of 644 noted above.

For the purposes of this exercise, we will assume that a virtual directory for the EnterpriseInvestments Site has been created and that a virtual directory called FastForwardSFTP has been created on the SFTP server pointing to that virtual root.

To set up secure FTP Publishing,

1. Create a Site definition that specifies the SFTP connection data.
2. Define Content Lists that use the SFTP Delivery Type.
3. Define Editions that use the Content Lists defined in Step 2.

The following topics illustrate an example implementation based on a copy of the EnterpriseInvestments Site.

Defining a Site Registration for SFTP Publishing

This first step in setting up delivery using SFTP is to define a Site registration with FTP data. In this exercise, we will copy the EnterpriseInvestments Site registration and add the FTP data. We will rename the Site registration to "EnterpriseInvestmentsSFTP".

We will assume the following data for the FTP connection:

- The IP address of the production Web server is 255.255.255.112.
- The SFTP port of the production Web server is 81.
- The SFTP user on the production Web server is FastForwardSFTP.
- The FastForwardFTP user's password is FastForward.

We will assume that the remaining data will remain the same for the Enterprise Investments FTP Site.

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

To create the Site registration for FTP publishing:

1 In Content Explorer, select the Publishing Design tab.

2 In the Navigation bar, select the Sites node.

Content Explorer displays the Sites List.

3 Select the Enterprise Investments Site.

4 In the Menu bar, choose *Action* > *Copy Selected Site*.

Rhythmyx creates a copy of the EnterpriseInvestments Site and displays it in the Site editor.

5 The default name of the copied site is *Copy of EnterpriseInvestments*. Change the name to *EnterpriseInvestmentsSFTP*.

6 For the pusposes of this exercise, we will assume that the rest of the Site registration data remains the same.

7 To add FTP data:

a) Click the Show Site details link.

Content Explorer displays several additional fields on the Site editor, including FTP Server IP address, FTP server port, User name, and Password.

b) In the FTP server IP address field, enter *255.255.255.112*.

c) In the FTP port field, enter *81*.

d) In the User name field, enter *FastForwardSFTP*.

e) In the Password field enter *FastForward*.

8 To save the Site registration, in the Menu bar, click [**Save**].

Defining SFTP Editions and Content Lists

The second and third steps in setting up SFTP delivery is to define Content Lists to use the SFTP Delivery Type and Editions that include these Content LIsts. The following procedure combines these two steps by copying the EI_Full Edition and its Content Lists to the EnterpriseInvestmentsSFTP Site. We will change the name of the Edition to EI_Full_SFTP, but will not change any other data. We will modify the Content Lists by changing the Delivery Type to *sftp* and renaming them to indicate that they use SFTP, but we will not modify any other data.

To create the EI_Full_FTP Edition:

1. In Content Explorer, select the Publishing Design tab.
2. In the Navigation bar, expand the Sites, Node. Expand the EnterpriseInvestmentsSFTP node. Click on the Editions link.

   Rhythmyx displays the Edition List.
3. In the Menu bar, choose *Action > Copy Edition From Other Site*.
4. Select the radio button in the row of the EI_Full Edition. Check the Copy Content Lists of the Selected Edition box.
5. In the Menu bar, click *Done*.

   Rhythmyx copies the Edition and Content Lists. When the processing is complete, Rhythmyx displays the Edition Editor.
6. Change the Name to *EI_Full_SFTP*.
7. In the Menu bar, click *Save*.

   Rhythmyx saves the Edition and displays the Edition List.
8. Under the EnterpriseInvestmenstSFTP node, click Content Lists.

   Rhythymyx displays the List Content Lists page.
9. Click Copy_of_rffEIFullNonbinary.

   Rhythmyx displays the Content List Editor for the Copy_of_rffEIFullNonbinary Content List.
10. Change the Name to *rffEIFullNonbinarySFTP*.
11. In the Delivery Type drop list, choose *sftp*.
12. In the Menu bar, click *Save*.
13. Repeat Steps 9 to 12 for the Copy_of_rffEIFullBinary Content List.

CHAPTER 10

Database Publishing in Rhythmyx

Database publishing allows you to deliver Rhythmyx content to external target database repositories where it can be consumed by web-driven applications, such as portals. Rhythmyx only delivers the content to the target repository. Rhythmyx does not interact directly with the application consuming the content.

The following RDBMSs are supported as targets for database publishing:

- Microsoft SQL Server 2000, 2005, and 2008
- Oracle 9, 10, and 11

  Note: The database publishing Delivery Type assumes that Rhythmyx generates the primary keys. In some Oracle environments, Oracle defines the primary keys. These environments require a custom Delivery Type. For details about developing a custom Delivery Type, see the *Rhythmyx Technical Reference Manual*. Assistance from Percussion Professional Services Organization is recommended for this implementation.
- MySQL 5.1

  NOTE: MySQL is supported only as a target for database publishing. This RDBMS is not supported as the Rhythmyx Repository.

  NOTE: While Rhythmyx is shipped with a driver configuration for MySQL, the .jar files for the driver are not included due to the limitations of the GPL used for MySQL. You can download the drivers from ***www.mysql.com*** (http://www.mysql.com). The drivers should be added to the directory <Rhythmyxroot>/Appserver/Server/rx/lib.

Rhythmyx is shipped with a standard database publishing Delivery Type which should meet the requirements of most customers. If the standard database Publishing Delivery Type does not meet your needs, you can develop a custom Delivery Type. For details, see the *Rhythmyx Technical Reference Manual*.

The key element of database publishing is the database publishing Template. The Template is used to retrieve the Content Item data from the Rhythmyx Repository, and it maps the Content Item data fields to the tables and columns in the target database.

As always, planning before the implementation is important. In *Modelling and Design of a Rhythmyx Content Management System*, we examined some of the issues that should be addressed when planning a database publishing implementation.

Database Publishing Implementation Process

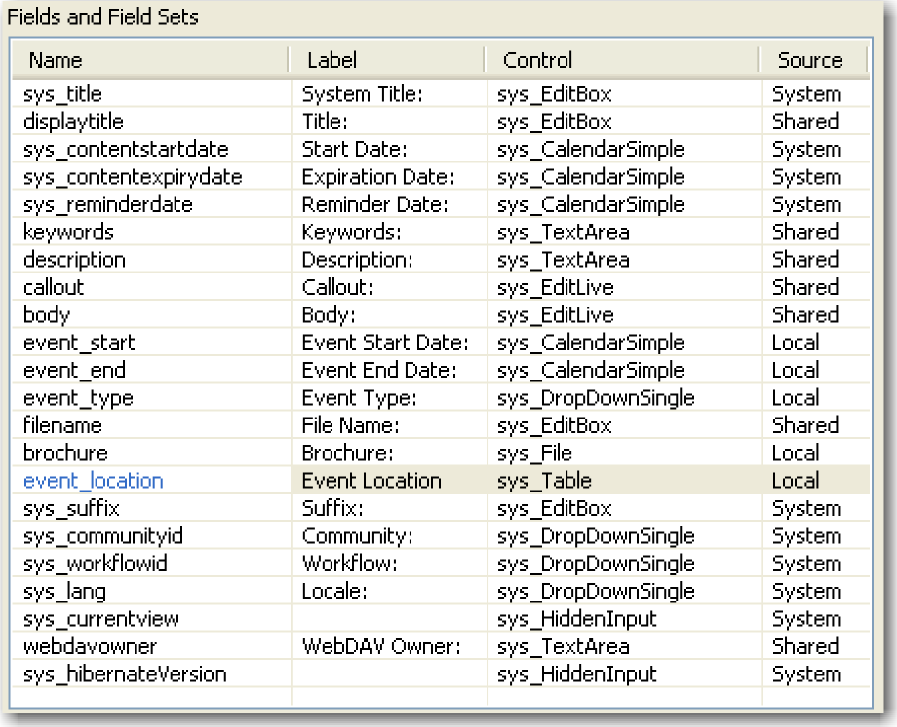
To implement database publishing:

1. Model and design your database publishing implementation. For details about modeling and design of database Templates, see *Modelling and Design of a Rhythmyx Content Management System*.
2. In the Rhythmyx Server Administrator, create a Connection to the target database or schema.
3. Create the database publishing Templates.
4. Create publishing configurations (Sites, Editions, Content Lists, and Contexts) to publish to the database.

Database Publishing Specifications

In our example implementation, we will publish Event Content Items to a database repository that supports a web applications users query to find Events. In our example, we will publish to Microsoft SQL Server. (In fact, for convenience, we will publish to the same instance as the Rhythmyx Repository; in you implementation, the target database is likely to be in a different instance.)

We will assume that an additional binary field, brochure, has been added to the Event Content Type to store a printable file about the event that customers can download. The following screenshot illustrates the definition of the Event Content Type we will be publishing:

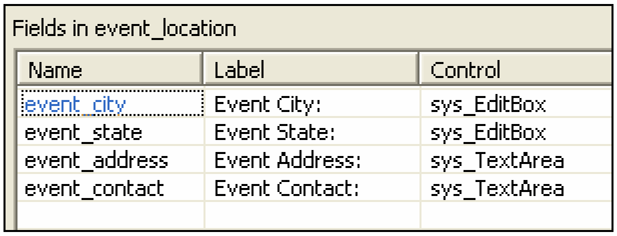


Name	Label	Control	Source
sys_title	System Title:	sys_EditBox	System
displaytitle	Title:	sys_EditBox	Shared
sys_contentstartdate	Start Date:	sys_CalendarSimple	System
sys_contentexpirydate	Expiration Date:	sys_CalendarSimple	System
sys_reminderdate	Reminder Date:	sys_CalendarSimple	System
keywords	Keywords:	sys_TextArea	Shared
description	Description:	sys_TextArea	Shared
callout	Callout:	sys_EditLive	Shared
body	Body:	sys_EditLive	Shared
event_start	Event Start Date:	sys_CalendarSimple	Local
event_end	Event End Date:	sys_CalendarSimple	Local
event_type	Event Type:	sys_DropDownSingle	Local
filename	File Name:	sys_EditBox	Shared
brochure	Brochure:	sys_File	Local
event_location	Event Location	sys_Table	Local
sys_suffix	Suffix:	sys_EditBox	System
sys_communityid	Community:	sys_DropDownSingle	System
sys_workflowid	Workflow:	sys_DropDownSingle	System
sys_lang	Locale:	sys_DropDownSingle	System
sys_currentview		sys_HiddenInput	System
webdavowner	WebDAV Owner:	sys_TextArea	Shared
sys_hibernateVersion		sys_HiddenInput	System

Figure 273: Event Content Type definition for the Database Publishing Examples

While many customers publish Content Item data to a single target table, it is also common to publish data to a parent table and one or more child tables. We will illustrate two options for publishing child data:

- publishing data from a child field set in a Rhythmyx Content Item to a child table in the target database. We will assume the following child editor has been defined in Rhythmyx:



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 274: Event Location field set

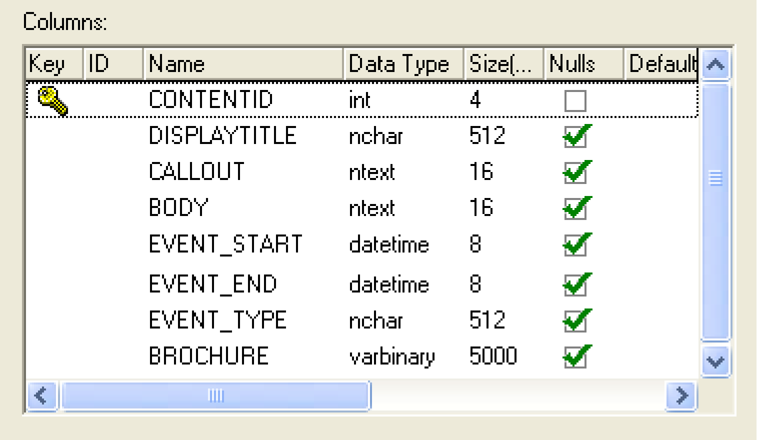
- publishing data drawn from an external database repository to a child table in the target database. We will use the Northwind database available for Microsoft SQL Server, and draw the data from the Employees table.

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 stores basic location and contact information: address, city state, and contact.
- Both the parent and the child tables are required to have primary/foreign keys. Standard practice for Database Publishing in Rhythmyx is the include a column for the Content ID. That column should be the primary key for the parent table, and the foreign key for relationships between the parent tables and the child tables. (Note: The database publishing Delivery Type assumes that Rhythmyx generates the primary key in the table. In some Oracle environments, Oracle defines the primary key for the table. These environments require a custom Delivery Type. For details about developing a custom Delivery Type, see the *Rhythmyx Technical Reference Manual*. Assistance from Percussion Professional Services Organization is recommended for this implementation.)
- The child table also uses the seq column as a primary key. This is necessary for enabling 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.

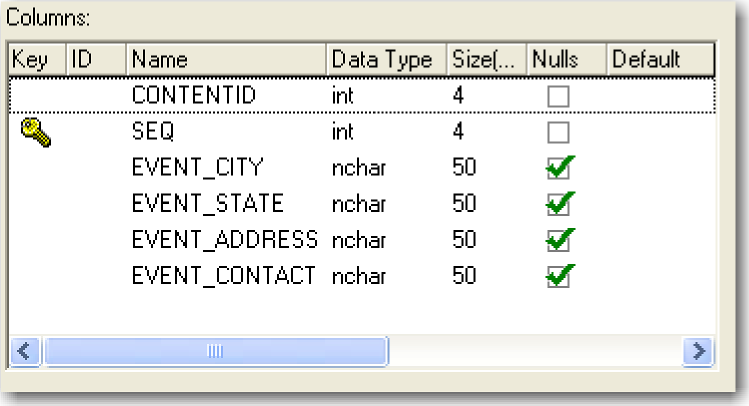
The target tables should resemble:



Columns:

Key	ID	Name	Data Type	Size(...	Nulls	Default
🔑		CONTENTID	int	4	□	
		DISPLAYTITLE	nchar	512	☑	
		CALLOUT	ntext	16	☑	
		BODY	ntext	16	☑	
		EVENT_START	datetime	8	☑	
		EVENT_END	datetime	8	☑	
		EVENT_TYPE	nchar	512	☑	
		BROCHURE	varbinary	5000	☑	

Figure 275: TARGET_CONTENT table



Columns:

Key	ID	Name	Data Type	Size(...	Nulls	Default
		CONTENTID	int	4	□	
🔑		SEQ	int	4	□	
		EVENT_CITY	nchar	50	☑	
		EVENT_STATE	nchar	50	☑	
		EVENT_ADDRESS	nchar	50	☑	
		EVENT_CONTACT	nchar	50	☑	

Figure 276: TARGET_LOCATION table

NOTE: Your database tables may already exist. If not, you should define them at this point. For this example, we have created a new database, rx_Events, in the same Microsoft SQL Server instance as the Rhythmyx Repository. If you want to follow the example implementation, create the tables as defined above in a different SQL Server database than your Rhythmyx Repository.

Creating a JNDI Datasource Configuration and a Database Connection

The first task in implementing database publishing is to define the connection to the target database. Usually, the target database is on a different machine or at least a different instance than your Rhythmyx Repository, so you will have to create the JNDI datasource configuration first. These configurations are created in the Rhythmyx Server Administrator.

In the following procedure, we will create an example JNDI datasource configuration to a database on an Oracle server. We will assume the following properties:

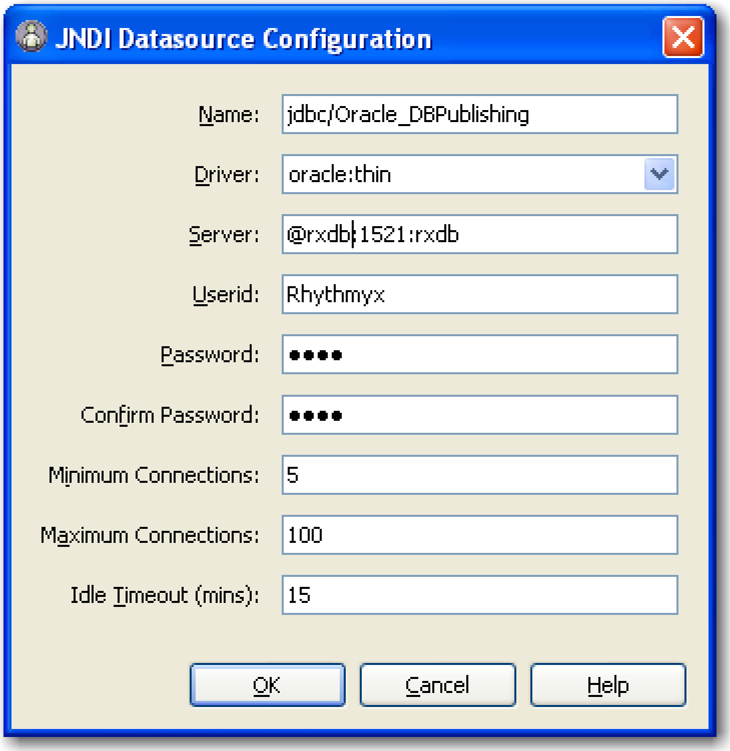
- The Oracle server runs on a machine called *rxdb* on the default Oracle port (1521). The system identifier is also *rxdb*.
- A user called *Rhythmyx* has been created on the Oracle server, with a password of *demo*.

We will name the JNDI datasource configuration *Oracle_DBPublishing*.

To create the Oracle_DBPublishing datasource configuration:

1. Start and log in to the Rhythmyx Server Administrator.
2. Along the top, click the Datasources tab. Along the bottom, click the JNDI tab.
3. Click the [**Add**] button.
4. The Server Administrator displays the JNDI Datasource Configuration dialog.
5. The Name field defaults to jdbc/. After the "/" character, enter *Oracle_DBPublishing*.
6. In the Driver drop list, choose *oracle:thin*.
7. In the Server field, enter the connect string: *@rxdb:1521:rxdb*.
8. In the Password and Confirm Password fields, enter *demo*.
9. Do not change the values in any of the other fields.

When complete, the JDNI datasource configuration should resemble the following screenshot:



Name: jdbc/Oracle_DBPublishing

Driver: oracle:thin

Server: @rxdb:1521:rxdb

Userid: Rhythmyx

Password: ●●●●

Confirm Password: ●●●●

Minimum Connections: 5

Maximum Connections: 100

Idle Timeout (mins): 15

Figure 277: Example Datasource Configuration for an Oracle Database

10. Click the [OK] button to save the configuration.
11. On the Server Administrator, click the [**OK**] button.

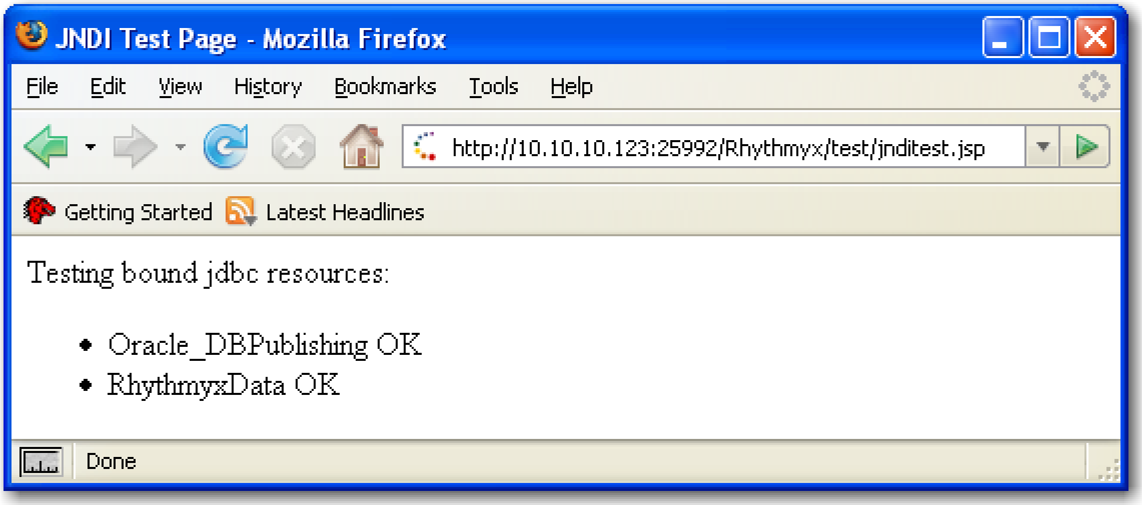
You must shut down and restart the Rhythmyx server before your changes will take effect. One you have restarted the Rhythmyx server, you can test the datasource configuration. To test the configuration:

1. Start a browser and enter *http://localhost:9992*, where *localhost* is the name or IP address of the machine where Rhythmyx is installed and *9992* is the Rhythmxy port.

   Rhythmyx returns the Application Server Home page.
2. Click on the Testing and Debugging Tools for Implementers link.
3. Rhythmyx returns a login page. Log in to Rhythmyx.

   Rhythmyx returns the Debugging and Testing page.
4. Click on the Test Bound JDBC Resources link.

Rhythmyx returns the JNDI Test Page showing the status of your JDNI datasource configurations:


JNDI Test Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://10.10.10.123:25992/Rhythmyx/test/jnditest.jsp
Getting Started
Latest Headlines
Done

Testing bound jdbc resources:

- Oracle_DBPublishing OK
- RhythmyxData OK

Figure 278: JNDI Test page showing a successful connection

Common problems with datasource configurations include:

- jTDS:
  - invalid connect string:

```
SQL_Connection Exception: Could not create connection; -
nested throwable: (java.sql.SQLException: The syntax of the
connection URL 'jdbc:jtds:sqlserver:sqlserver' is
invalid.); - nested throwable:
(org.jboss.resource.JBossResourceException: Could not
create connection; - nested throwable:
(java.sql.SQLException: The syntax of the connection URL
'jdbc:jtds:sqlserver:sqlserver' is invalid.)) getting
connection
```

In this case, the connect string is missing the leading double-slashes ("//"). The correct format of the connect string is *//servername*.

```
SQLAlt Exception: Could not create connection; - nested
throwable: (java.sql.SQLException: Unknown server host name
'sqserver'.); - nested throwable:
(org.jboss.resource.JBossResourceException: Could not
create connection; - nested throwable:
(java.sql.SQLException: Unknown server host name
'sqserver'.)) getting connection
```

In this case the machine name was specified incorrectly. Correct the name or IP address of the machine.

  - invalid Userid or Password:

    ```
    SQLAlt Exception: Could not create connection; - nested
    throwable: (java.sql.SQLException: Login failed for user
    'sb'.); - nested throwable:
    (org.jboss.resource.JBossResourceException: Could not
    create connection; - nested throwable:
    (java.sql.SQLException: Login failed for user 'sb'.))
    getting connection
    ```

- mySQL
  - invalid connect string:

    ```
    mysql_DBPublishing Exception: Could not create connection;
    - nested throwable:
    (org.jboss.resource.JBossResourceException: Apparently
    wrong driver class specified for URL: class:
    com.mysql.jdbc.Driver, url: jdbc:mysql:qadb); - nested
    throwable: (org.jboss.resource.JBossResourceException:
    Could not create connection; - nested throwable:
    (org.jboss.resource.JBossResourceException: Apparently
    wrong driver class specified for URL: class:
    com.mysql.jdbc.Driver, url: jdbc:mysql:qadb)) getting
    connection
    ```

    In this case, the connect string is missing the leading double-slashes ("//"). The correct format of the connect string is *//servername*.

    ```
    mysql_DBPublishing Exception: Could not create connection;
    - nested throwable:
    (com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
    Communications link failure Last packet sent to the server
    was 0 ms ago.); - nested throwable:
    (org.jboss.resource.JBossResourceException: Could not
    create connection; - nested throwable:
    (com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
    Communications link failure Last packet sent to the server
    was 0 ms ago.)) getting connection
    ```

    In this case the machine name was specified incorrectly. Correct the name or IP address of the machine.

  - invalid Userid or Password:

    ```
    mySQL Exception: Could not create connection; - nested
    throwable: (java.sql.SQLException: Access denied for user
    'rxserver'@'rxserver' (using password: YES)); - nested
    throwable: (org.jboss.resource.JBossResourceException:
    Could not create connection; - nested throwable:
    (java.sql.SQLException: Access denied for user
    'rxserver'@'rxserver' (using password: YES))) getting
    connection
    ```

- Oracle
    - invalid connect string:

      ```
      Oracle_DBPublishing Exception: Could not create connection;
      - nested throwable: (java.sql.SQLException: Io exception:
      The Network Adapter could not establish the connection); -
      nested throwable:
      (org.jboss.resource.JBossResourceException: Could not
      create connection; - nested throwable:
      (java.sql.SQLException: Io exception: The Network Adapter
      could not establish the connection)) getting connection
      ```
    - invalid Userid or Password:

      ```
      Oracle_DBPublishing Exception: Could not create connection;
      - nested throwable: (java.sql.SQLException: ORA-01017:
      invalid username/password; logon denied ); - nested
      throwable: (org.jboss.resource.JBossResourceException:
      Could not create connection; - nested throwable:
      (java.sql.SQLException: ORA-01017: invalid
      username/password; logon denied )) getting connection
      ```

Now you can create the database connection. For details, see ***Creating a Connection to an External Repository*** (on page 258)

Creating a Database Publishing Template

Once you have created the connection, you can create the database publishing Template. In this example, we will create a Template called rffEventDB to publish Event Content Items to the target database tables (TARGET_LOCATION and TARGET_CONTACT) that we described in the database publishing specification. We will assume that a database connection, named *Event*, has been created to connect to the database where these tables reside.

We will make the Template available to both the member and admin communities of both the EnterpriseInvestments and CorporateInvestments Sites.

We will also demonstrate two options for populating a child table: populating the child table with data defined in a child editor of the Content Type, and populating the child table with data derived from an external data repository.

To create the rffDbEvent Template:

1 Log in to the Rhythmyx Workbench and open the Assembly view.

2 In the Menu bar, choose *File* > *New* > *Template*.

The Rhythmyx Workbench displays the first dialog of the New Template Wizard.

3 Select the Database publishing XML radio button. Click the [**Next**] button.

The Rhythmyx Workbench displays the General Properties dialog of the New Template Wizard.

4 In the Template name field, enter *rffDbEvent*. This value also defaults to the Label field. Optionally, enter the description *Event Database Publishing Template*.

5 By default, new Templates are available to all Communities. Optionally, move the Default Community to the Available Communities list.

NOTE: Users upgrading from earlier versions of Rhythmyx may recall that you also had to add the XML definition of the target tables to the Template. That functionality is now included in the Template on another dialog as described in the next step of this procedure.

6 Click the [**Next**] button.

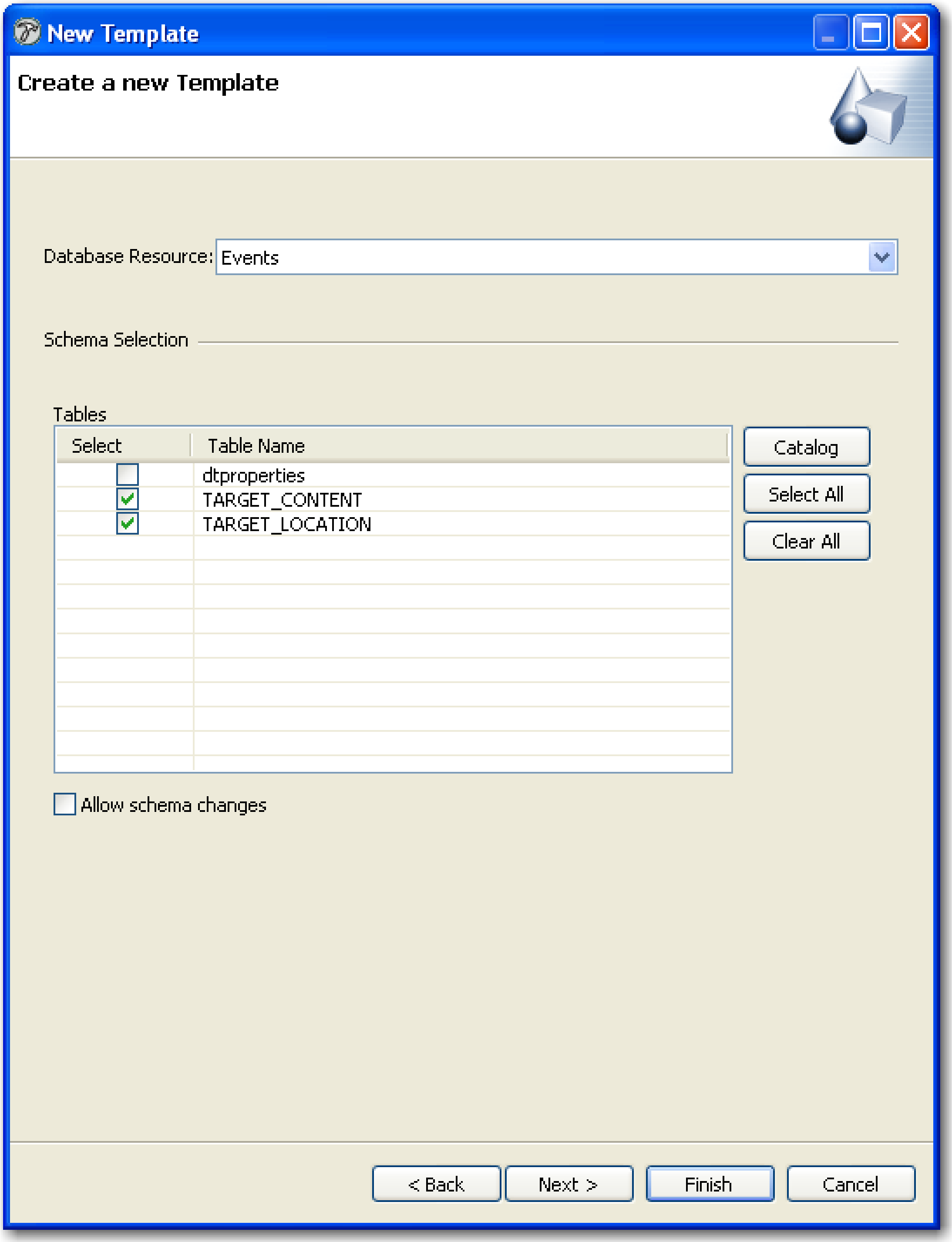
The Rhythmyx Workbench displays the Target Tables dialog.

7 The Data Resource drop list defaults to the first Connection defined on the Connections subtab of the Datasources tab of the Rhythmyx Server Administrator. (Typically, this will be the connection to the Rhythmyx Repository.) Select the *Event* connection.

8 Click the [**Catalog**] button.

Rhythmyx catalogs the database specified by the connection and populates the Tables table with a list of the tables found in that database.

9 In the Select column, check the boxes for the TARGET_LOCATION and TARGET_CONTACT tables.



New Template

Create a new Template

Database Resource: Events

Schema Selection

Tables

Select	Table Name
☐	dtproperties
☑	TARGET_CONTENT
☑	TARGET_LOCATION

☐ Allow schema changes

Figure 279: Table Definitions for the rffEventDB Template

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

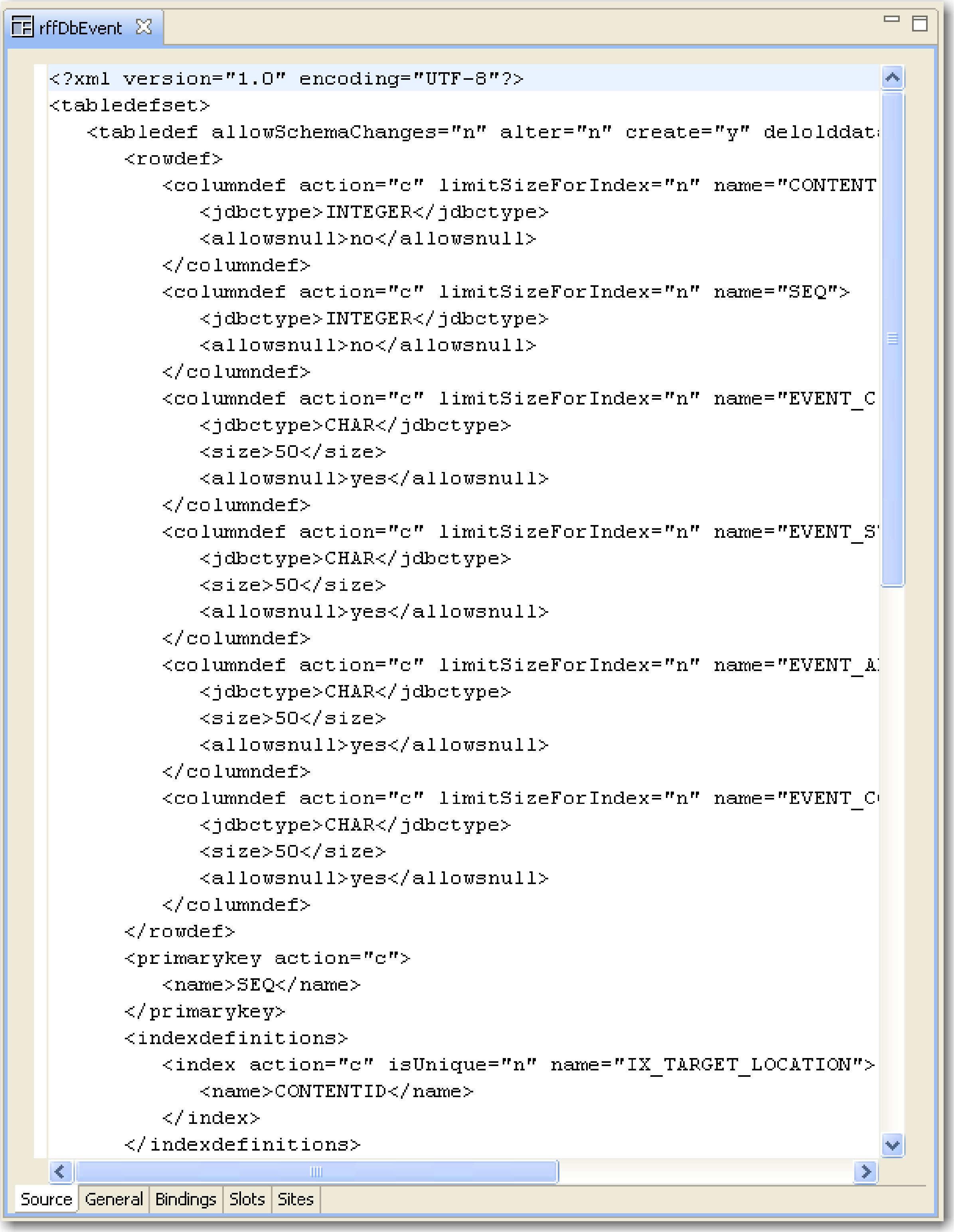
11 The Rhythmyx Workbench displays the Slots dialog. A Database Publishing Template does not include any Slots, so do not add any to the Template. Click the [**Next**] button.

The Rhythmyx Workbench displays the Content Types dialog.

12 In the Available Content Types list, select the rffEvent Content Type, then click the > button to associate the Content Type with the Template.

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

Rhythmyx creates the Template and displays it in the Template Editor. The Source tab includes the table definition XML used to map the Content Item data to the output tables:

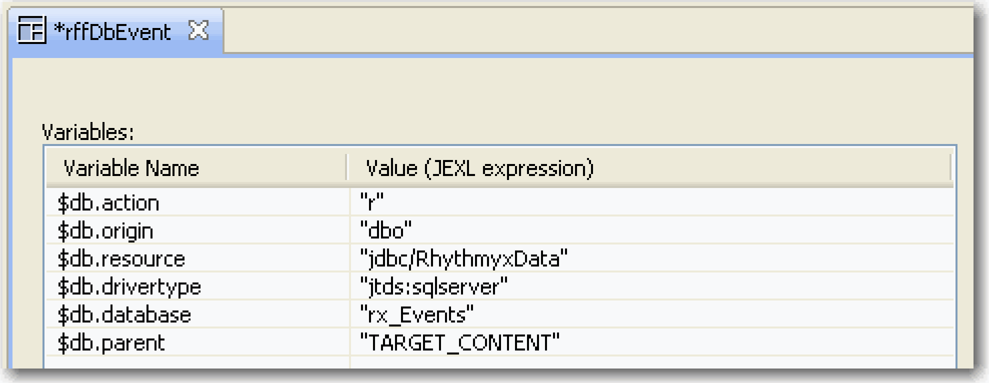


```
<?xml version="1.0" encoding="UTF-8"?>
<tabledefset>
   <tabledef allowSchemaChanges="n" alter="n" create="y" delolddat
      <rowdef>
         <columndef action="c" limitSizeForIndex="n" name="CONTENT
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>no</allowsnull>
         </columndef>
         <columndef action="c" limitSizeForIndex="n" name="SEQ">
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>no</allowsnull>
         </columndef>
         <columndef action="c" limitSizeForIndex="n" name="EVENT_C
            <jdbctype>CHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" limitSizeForIndex="n" name="EVENT_S
            <jdbctype>CHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" limitSizeForIndex="n" name="EVENT_A
            <jdbctype>CHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" limitSizeForIndex="n" name="EVENT_C
            <jdbctype>CHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
      </rowdef>
      <primarykey action="c">
         <name>SEQ</name>
      </primarykey>
      <indexdefinitions>
         <index action="c" isUnique="n" name="IX_TARGET_LOCATION">
            <name>CONTENTID</name>
         </index>
      </indexdefinitions>
```

Figure 280: Source tab of the rffEventDB Template showing the table definition XML

The XML conforms to the sys_Tabledef.dtd. For details about the contents of the XML, see the DTD.

While the Bindings tab includes the set of standard bindings required by a database Template:



Variables:

Variable Name	Value (JEXL expression)
$db.action	"r"
$db.origin	"dbo"
$db.resource	"jdbc/RhythmyxData"
$db.drivertype	"jtds:sqlserver"
$db.database	"rx_Events"
$db.parent	"TARGET_CONTENT"

Figure 281: Default bindings generated when creating a database publishng Template

These bindings define the connection information for the database publishing Delivery Type to use when connecting to the target database and delivering the data, or to specify the action to be performed when delivering the Content Item data.

Binding	**Function**	**Value**
$db.action	Action to perform on the database with the content.	"r" - "Replace" Inserts a row for the content. Deletes it first if it already exists. This is the default value when the binding is created, and is the assumed value when the value is missing or invalid. "n" - "New" Inserts a row for the content if it does not already exist. "u" - "Update" Updates the row for the content if it already exists. "d" - "Delete" Deletes the row for the content if it exists. NOTE: The "u" option should not be used if a Content Type published using this Template includes complex child data. The "r" option should be used instead.
$db.origin	Name of the target database schema.	"dbo"
$db.resource	Name of the datasource	"jdbc/[datasource name]"; in this case, "jdbc/RhythmyxData"
$db.drivertype	Database driver type.	May be one of the following values, depending on your dbms: ▪ jtds:sqlserver ▪ oracle:thin ▪ MySQL ▪ db2

Binding	Function	Value
$db.database	Used for MS SQL Server if the database name is not specified in the datasource. Name of the database.	In this example: "rx_Events". Optional in this example since it is specified in the datasource.
$db.parent	Parent target database table. or only target database table if no child data is being published	In this example: "TARGET_CONTENT"

Defining Bindings to Publish Content Item Data

A database publishing Template must include a set of bindings that maps Content Item fields to columns in the target database table. Each column you want to publish is defined as a $row.COLUMNNAME binding, where COLUMNNAME is the name of the target database column to which the Content Item field data will be published. The value of each of these bindings is $sys.item.getProperty to retrieve the data from the specified field. Bindings returning data from fields with a data type of *string* should use the .String (or .getString) function to ensure that the data is returned in the correct format.

```
$row.COLUMNNAME=$sys.item.getProperty('rx:fieldname').String
```

For example:

```
$row.DISPLAYTITLE=$sys.item.getProperty('rx:displaytitle').String
```

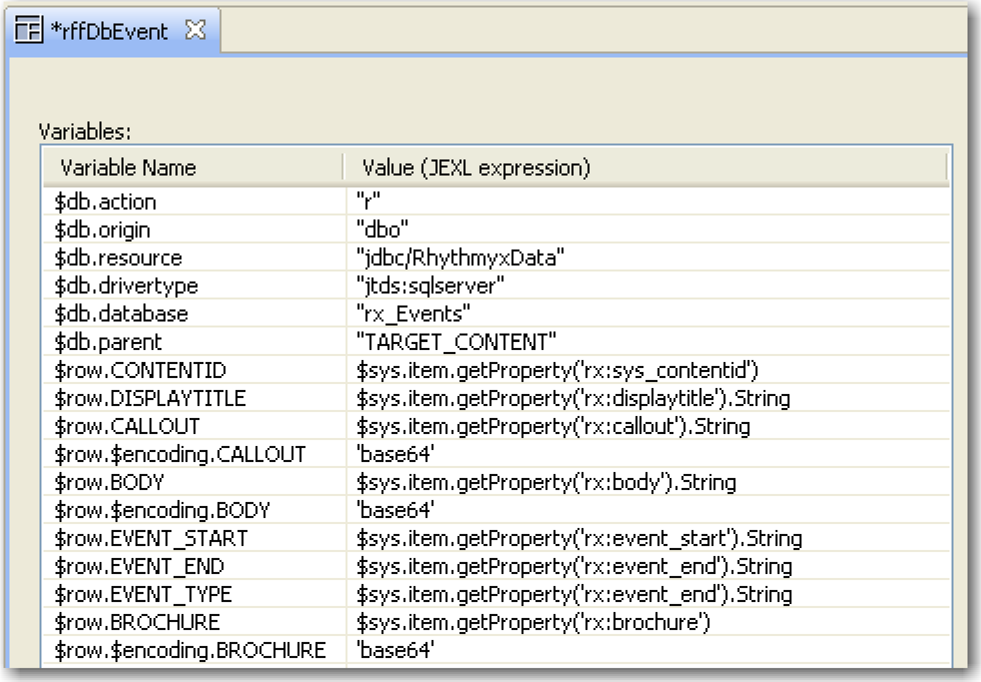
In some cases, the bindings also specify encoding for specially coded values, such as binary data or rich text data with HTML markup. The variable name for these bindings is $row.$encoding.COLUMNNAME. The value of the binding is either *base64* or null:

```
$row.$encoding.COLUMNNAME=base64
```

We are assuming that a binary field, brochure, has been added to the rffEvent Content Type to store a downloadable binary file. The bindings for this field would be:

```
$row.BROCHURE=$sys.item.getProperty('rx:brochure')
$row.encoding.BROCHURE.='base64'
```

The following graphic illustrates the bindings of the rffDbEvent Template to publish the data from the rffEvent Content Type, without any child data:



Variable Name	Value (JEXL expression)
$db.action	"r"
$db.origin	"dbo"
$db.resource	"jdbc/RhythmyxData"
$db.drivertype	"jtds:sqlserver"
$db.database	"rx_Events"
$db.parent	"TARGET_CONTENT"
$row.CONTENTID	$sys.item.getProperty('rx:sys_contentid')
$row.DISPLAYTITLE	$sys.item.getProperty('rx:displaytitle').String
$row.CALLOUT	$sys.item.getProperty('rx:callout').String
$row.$encoding.CALLOUT	'base64'
$row.BODY	$sys.item.getProperty('rx:body').String
$row.$encoding.BODY	'base64'
$row.EVENT_START	$sys.item.getProperty('rx:event_start').String
$row.EVENT_END	$sys.item.getProperty('rx:event_end').String
$row.EVENT_TYPE	$sys.item.getProperty('rx:event_end').String
$row.BROCHURE	$sys.item.getProperty('rx:brochure')
$row.$encoding.BROCHURE	'base64'

Figure 282: rffEventDB Template bindings to publish the Event Content Type without children

Binding	**Function**	**Value**
$db.action	Action to perform on the database with the content.	"r" - "Replace" Inserts a row for the content. Deletes it first if it already exists. This is the default value when the binding is created, and is the assumed value when the value is missing or invalid. "n" - "New" Inserts a row for the content if it does not already exist. "u" - "Update" Updates the row for the content if it already exists. "d" - "Delete" Deletes the row for the content if it exists. NOTE: The "u" option should not be used if a Content Type published using this Template includes complex child data. The "r" option should be used instead.
$db.origin	Name of the target database schema.	"dbo"
$db.resource	Name of the datasource	"jdbc/[datasource name]"; in this case, "jdbc/RhythmyxData"

Binding	Function	Value
$db.drivertype	Database driver type.	May be one of the following values, depending on your dbms: ▪ jtds:sqlserver ▪ oracle:thin ▪ MySQL ▪ db2
$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"
$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 $sys.item.getProperty("rx:sys_contentid") gets the value of sys_contentid from the Content Item. For more information about JEXL expressions, see the topic ***Bindings*** (see page 141).
$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 $sys.item.getProperty("rx:displaytitle").String gets the value of displaytitle 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 $sys.item.getProperty("rx:callout").String gets the value of callout from the Content Item.
$row.$encoding.CALLOUT	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.BROCHURE column below. In this case, the value is *'base64'*.
$row.BODY	Mapping for the specified column in the parent table. In this example, the mapping is for the BODY column.	A literal value or JEXL expression. In this example: the JEXL expression $sys.item.getProperty("rx:body").String gets the value of body from the Content Item.

Binding	Function	Value
$row.$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.BROCHURE column below. In this case, the value is *'base64'*.
$row.EVENT_START	Mapping for the specified column in the parent table. In this example, the mapping is for the EVENT_START column.	A literal value or JEXL expression. In this example: the JEXL expression $sys.item.getProperty("rx:event_start") gets the value of event_start from the Content Item.
$row.EVENT_END	Mapping for the specified column in the parent table. In this example, the mapping is for the EVENT_END column.	A literal value or JEXL expression. In this example: the JEXL expression $sys.item.getProperty("rx:event_end") gets the value of event_end from the Content Item.
$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 $sys.item.getProperty("rx:event_type").String gets the value of event_type from the Content Item.
$row.BROCHURE	Mapping for the specified column in the parent table. In this example, the mapping is for the BROCHURE column.	A literal value or JEXL expression. In this example: the JEXL expression $sys.item.getProperty("rx:event_type").String gets the value of event_type from the Content Item.

Binding	Function	Value
$row.$encoding.BROCHURE	The type of encoding for the specified column.	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. Currently, the only encoding type for Database Publishing is `base64`. ▪ use `base64` for: ▪ binary data; ▪ text data that includes HTML, XML, or SGML markup; ▪ Data from a column in the source table that uses rich-text formatting. In other cases, do not use an encoding parameter. Since the brochure field stores binary data, this binding has a value of *'base64'*. (Be sure to include the quotation marks. If the value is specified without quotation marks, the JEXL processor will return the error "can't overwrite cause" when assembling the Template.

Defining Bindings to Publish Local Child Data

To publishing child data managed within a Detail Editor for the Content Type:

- Add a $db.child[0] binding for each target child table; increment the counter for each target child table.
- Add a $child[0].COLUMNNAME binding for each child column you want to publish. The value of the binding uses the $rx.asmhelper.childValues function. This function takes requires the following parameters:
  - parentNode

    The Content Item whose child Fieldset children to return. Use $sys.item.
  - childName

    Name of the child Fieldset whose values you want to return.
  - propertyName

    Name of the field to return from the child Fieldset.

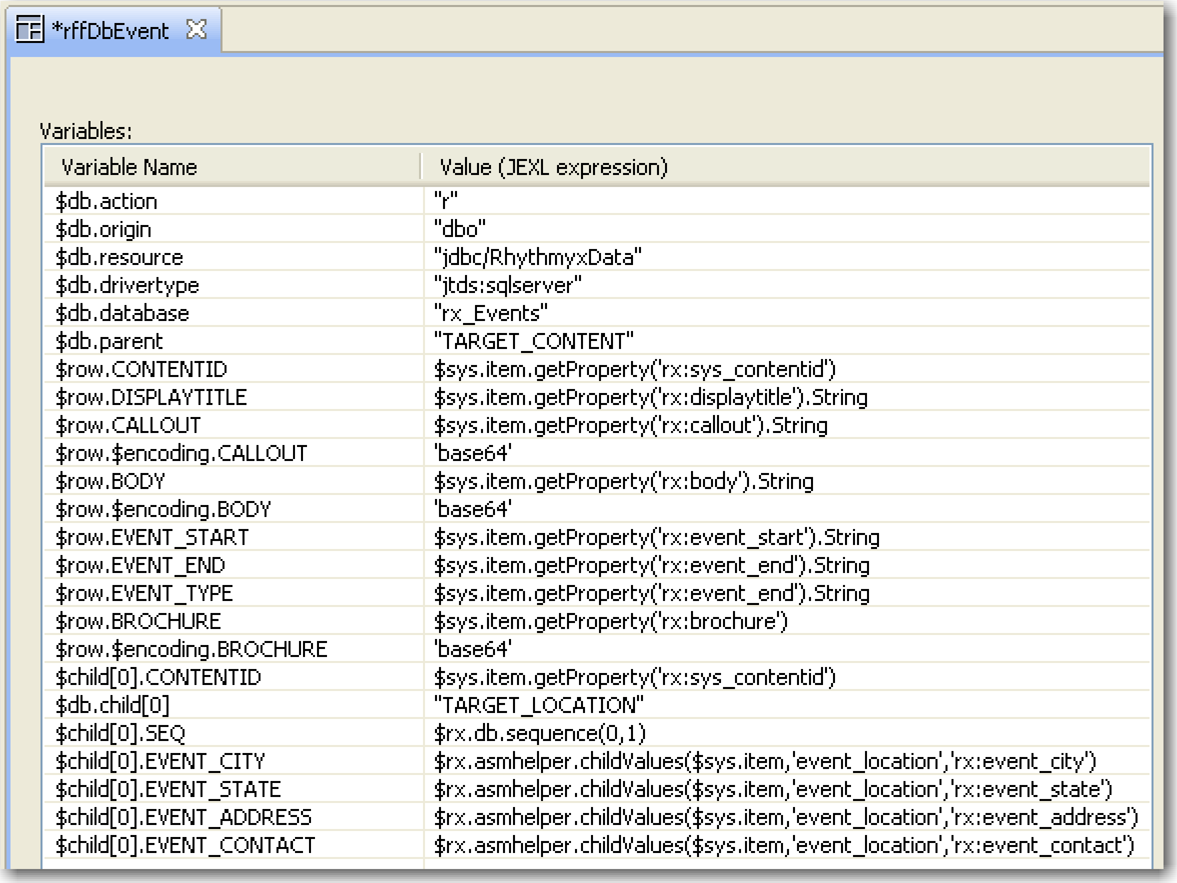
For example:

```
$child[0].COLUMNNAME=$rx.asmhelper.childValues($sys.item,'childfie
ldset','childfield')
```

Bindings returning data from fields with a data type of *string* should use the .String (or .getString) function to ensure that the data is returned in the correct format.

```
$child[0].COLUMNNAME=$rx.asmhelper.childValues($sys.item,'childfie
ldset','childfield').String
```

In this example, we will define bindings to publish data from the event_location child table we defined for the rffEvent Content Type. We will assume that the value of the event_contact field is defined locally as string data.



Variable Name	Value (JEXL expression)
$db.action	"r"
$db.origin	"dbo"
$db.resource	"jdbc/RhythmyxData"
$db.drivertype	"jtds:sqlserver"
$db.database	"rx_Events"
$db.parent	"TARGET_CONTENT"
$row.CONTENTID	$sys.item.getProperty('rx:sys_contentid')
$row.DISPLAYTITLE	$sys.item.getProperty('rx:displaytitle').String
$row.CALLOUT	$sys.item.getProperty('rx:callout').String
$row.$encoding.CALLOUT	'base64'
$row.BODY	$sys.item.getProperty('rx:body').String
$row.$encoding.BODY	'base64'
$row.EVENT_START	$sys.item.getProperty('rx:event_start').String
$row.EVENT_END	$sys.item.getProperty('rx:event_end').String
$row.EVENT_TYPE	$sys.item.getProperty('rx:event_end').String
$row.BROCHURE	$sys.item.getProperty('rx:brochure')
$row.$encoding.BROCHURE	'base64'
$child[0].CONTENTID	$sys.item.getProperty('rx:sys_contentid')
$db.child[0]	"TARGET_LOCATION"
$child[0].SEQ	$rx.db.sequence(0,1)
$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 283: Database publishing Template bindings when publishing child data managed in a child editor of the Content Type

For details of the parent bindings, see ***Defining Bindings to Publish Content Item Data*** (on page 378). (NOTE: The "u" option for the $db.action binding should not be used if a Content Type published using this Template includes complex child data. The "r" option should be used instead.)

Binding	**Function**	**Value**
$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".
$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 $sys.item.getProperty("rx:sys_contentid") 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 $rx.db.sequence(start value, increment value) function returns the start value the first time it is used, then adds the increment for each additional value. So $rx.db.sequence(1,1) 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: $rx.asmhelper.childValues($sys.item,"child field set name","rx:child field set field") where: $sys.item = 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 $rx.asmhelper.childValues($sys.item,"event_location", "rx:event_city") gets the value of event_city from the event_location child field set.

Binding	Function	Value
$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 $rx.asmhelper.childValues($sys.item,"event_location", "rx:event_state") gets the value of event_state from the event_location child field set.
$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 $rx.asmhelper.childValues($sys.item,"event_location", "rx:event_address") 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 $rx.asmhelper.childValues($sys.item,"event_location", "rx:event_contact") gets the value of event_contact from the event_location child field set.

Defining Bindings to Publish Child Data from an External Repository

Instead of being managed locally as child content, child data may be stored and managed in a remote database repository. To retrieve child data stored in a remote repository:

- Define a database connection to the remote repository. For details about creating a database connection, see ***Creating a Connection to an External Repository*** (on page 258).
- Define a binding that uses the binding function $rx.db.get to execute a SQL query in the remote repository to retrieve the data you want to publish.

  ```
  $soucedb=$rx.db.get("datasource","selectStatement")
  ```

  where `datasource` is the name of the database connection used to connect to the source database or schema, and `selectStatement` is the SQL query used to select data from that repository. For example:

  ```
  $sourcedb=$rx.db.get("NorthwindData","select
  City,Region,Address,Contact from Employees where Country='USA'")
  ```

- Define column bindings using the binding function $rx.asmhelper.mapvalues() to map the column data.

  ```
  $child[0].TARGETCOLUMN=$rx.asmhelper.mapValues ($sourcedb,'key')
  ```

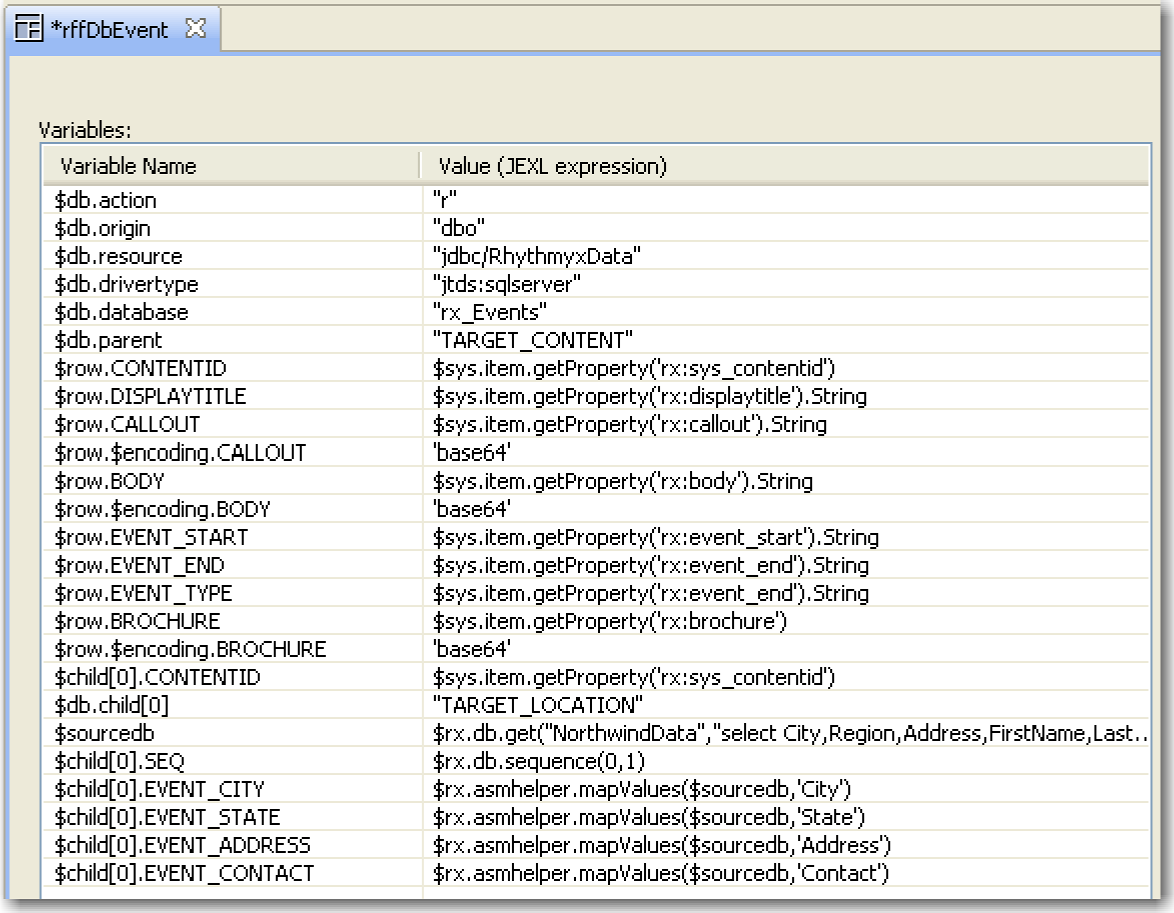
  where `$sourcedb` is the name of the name of the binding variable used to query the data from the remote repository and `key` is name of the remote repository column whose data you want to output to the column specified in the name of the binding variable being defined. For example:

  ```
  $child[0].EVENT_CITY=$rx.asmhelper.mapValues($sourcedb,'City')
  ```

The data is selected from the remote repository as a block and inserted into the target child table. In the following example, we are querying on the value in the Country column of the Employees table in the Northwind database. You might want to define values for the "where" clause in the Content Items. For example, if Events could take place in different States, the Content Type might include include a State drop list and store the value selected by the user in that field. You could then build the query in the following manner:

```
$state=$sys.item.getProperty('rx:state').String
$sourcedb=$rx.db.get("NorthwindData","select City,Region,Address,Contact
from Employees where Country="+$state);
```

In the example below, we assume that a Contact column is defined in the Employees table of the Northwind database that stores the name and phone number.



Variables:

Variable Name	Value (JEXL expression)
$db.action	"r"
$db.origin	"dbo"
$db.resource	"jdbc/RhythmyxData"
$db.drivertype	"jtds:sqlserver"
$db.database	"rx_Events"
$db.parent	"TARGET_CONTENT"
$row.CONTENTID	$sys.item.getProperty('rx:sys_contentid')
$row.DISPLAYTITLE	$sys.item.getProperty('rx:displaytitle').String
$row.CALLOUT	$sys.item.getProperty('rx:callout').String
$row.$encoding.CALLOUT	'base64'
$row.BODY	$sys.item.getProperty('rx:body').String
$row.$encoding.BODY	'base64'
$row.EVENT_START	$sys.item.getProperty('rx:event_start').String
$row.EVENT_END	$sys.item.getProperty('rx:event_end').String
$row.EVENT_TYPE	$sys.item.getProperty('rx:event_end').String
$row.BROCHURE	$sys.item.getProperty('rx:brochure')
$row.$encoding.BROCHURE	'base64'
$child[0].CONTENTID	$sys.item.getProperty('rx:sys_contentid')
$db.child[0]	"TARGET_LOCATION"
$sourcedb	$rx.db.get("NorthwindData","select City,Region,Address,FirstName,Last..
$child[0].SEQ	$rx.db.sequence(0,1)
$child[0].EVENT_CITY	$rx.asmhelper.mapValues($sourcedb,'City')
$child[0].EVENT_STATE	$rx.asmhelper.mapValues($sourcedb,'State')
$child[0].EVENT_ADDRESS	$rx.asmhelper.mapValues($sourcedb,'Address')
$child[0].EVENT_CONTACT	$rx.asmhelper.mapValues($sourcedb,'Contact')

Figure 284: Bindings to publish child data stored in a remote repository

For details of the parent bindings, see ***Defining Bindings to Publish Content Item Data*** (on page 378). (NOTE: The "u" option for the $db.action binding should not be used if a Content Type published using this Template includes complex child data. The "r" option should be used instead.)

Binding	**Function**	**Value**
$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".
$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 $sys.item.getProperty("rx:sys_contentid") 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 $rx.db.sequence(start value, increment value) function returns the start value the first time it is used, then adds the increment for each additional value. So $rx.db.sequence(1,1) returns 1, 2, 3, 4, etc. as values.
$sourcedb	Query to the remote repository where the data is stored.	The $rx.db.get(datasource,selectStatement) function, where datasource is the name of the database connection used to connect to the remote repository and selectStatement is the SQL query to run against that repository to return the data. Data is returned as a List of Maps In this example, the JEXL expression $rx.db.get("NorthwindData","select City,Region,Address,FirstName,LastName,HomePhone from Employees where Country='USA'") retrieves the values of the specified columns in the Employees table.

Binding	**Function**	**Value**
$child[0].EVENT_CITY	Mapping for the specified column in the child table. In this example, the data is derived from the City column of the Employees table in the Northwind database.	A literal value or JEXL expression. For data derived from a remote repository, use the function $rx.asmhelper.mapValues(maplist,'key') where maplist = the name of the binding variable used to query the remote repository key = the name of the column from the remote repository whose data you want to insert into the specified child column In this example, the JEXL expression $rx.asmhelper.mapValues($sourcedb,'City') gets the value of the City column from the data returned by the $sourcedb binding.
$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 $rx.asmhelper.mapValues($sourcedb,State') gets the value of the State column from the data returned by the $sourcedb binding.
$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 $rx.asmhelper.mapValues($sourcedb,'Address') gets the value of the Address column from the data returned by the $sourcedb binding.
$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 $rx.asmhelper.mapValues($sourcedb,'Contact) gets the value of the Contact column from the data returned by the $sourcedb binding.

Testing and Debugging a Database Publishing Template

Templates for database publishing can be previewed just as Templates to publish files can be previewed. Due to the different nature of the delivery target, the preview is different:

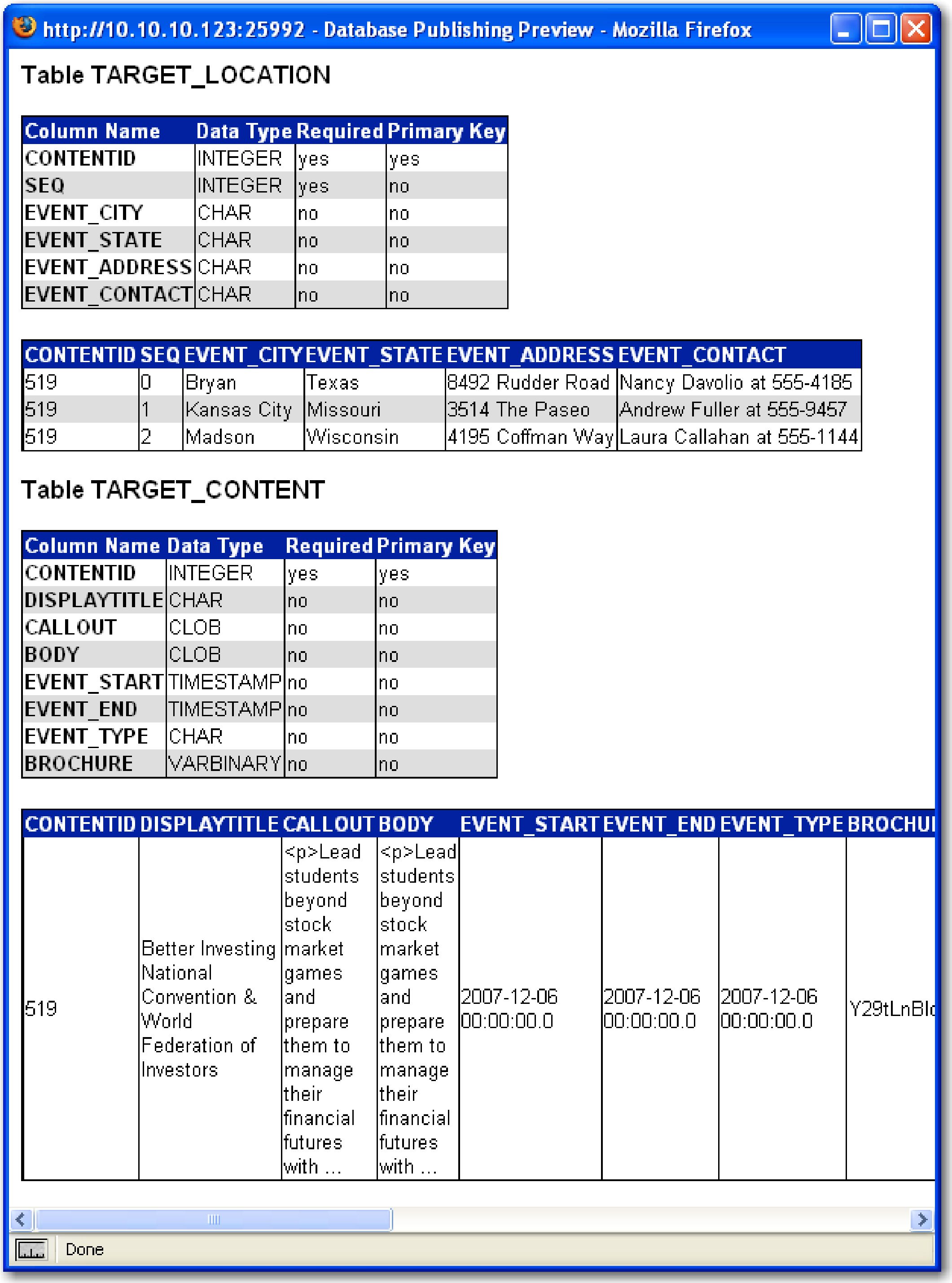

http://10.10.10.123:25992 - Database Publishing Preview - Mozilla Firefox
Done

Table TARGET_LOCATION

Column Name	Data Type	Required	Primary Key
CONTENTID	INTEGER	yes	yes
SEQ	INTEGER	yes	no
EVENT_CITY	CHAR	no	no
EVENT_STATE	CHAR	no	no
EVENT_ADDRESS	CHAR	no	no
EVENT_CONTACT	CHAR	no	no

CONTENTID	SEQ	EVENT_CITY	EVENT_STATE	EVENT_ADDRESS	EVENT_CONTACT
519	0	Bryan	Texas	8492 Rudder Road	Nancy Davolio at 555-4185
519	1	Kansas City	Missouri	3514 The Paseo	Andrew Fuller at 555-9457
519	2	Madson	Wisconsin	4195 Coffman Way	Laura Callahan at 555-1144

Table TARGET_CONTENT

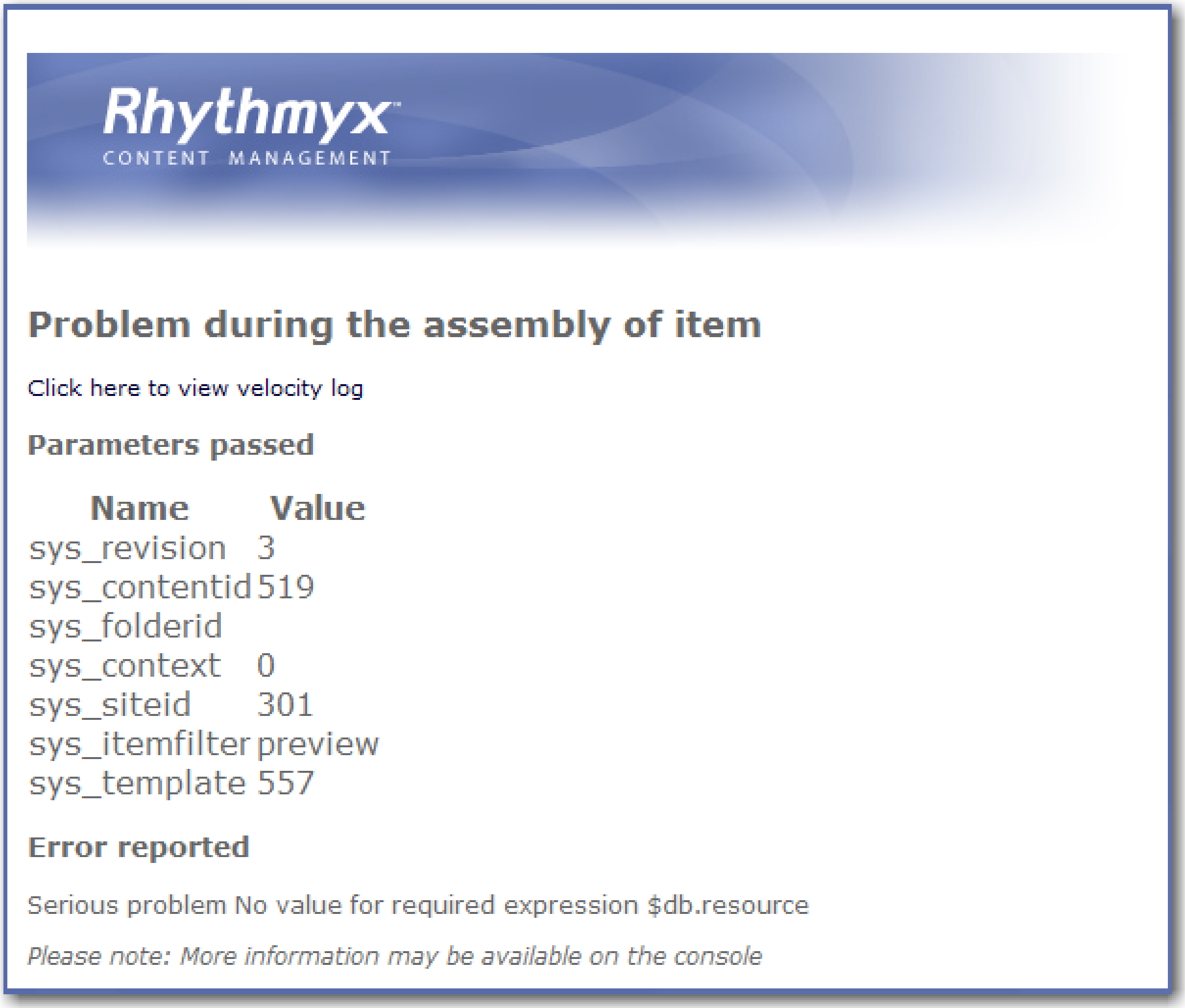
Column Name	Data Type	Required	Primary Key
CONTENTID	INTEGER	yes	yes
DISPLAYTITLE	CHAR	no	no
CALLOUT	CLOB	no	no
BODY	CLOB	no	no
EVENT_START	TIMESTAMP	no	no
EVENT_END	TIMESTAMP	no	no
EVENT_TYPE	CHAR	no	no
BROCHURE	VARBINARY	no	no

CONTENTID	DISPLAYTITLE	CALLOUT	BODY	EVENT_START	EVENT_END	EVENT_TYPE	BROCHU
519	Better Investing National Convention & World Federation of Investors	<p>Lead students beyond stock market games and prepare them to manage their financial futures with ...	<p>Lead students beyond stock market games and prepare them to manage their financial futures with ...	2007-12-06 00:00:00.0	2007-12-06 00:00:00.0	2007-12-06 00:00:00.0	Y29tLnBl

Figure 285: Previewing the rffEventDB Template

The preview displays the definition of the target tables and the output that will be published to each column based on the bindings defined in the Template.

If any of the bindings are incorrect, the following error message will be returned:



Problem during the assembly of item

Click here to view velocity log

Parameters passed

Name	Value
sys_revision	3
sys_contentid	519
sys_folderid	
sys_context	0
sys_siteid	301
sys_itemfilter	preview
sys_template	557

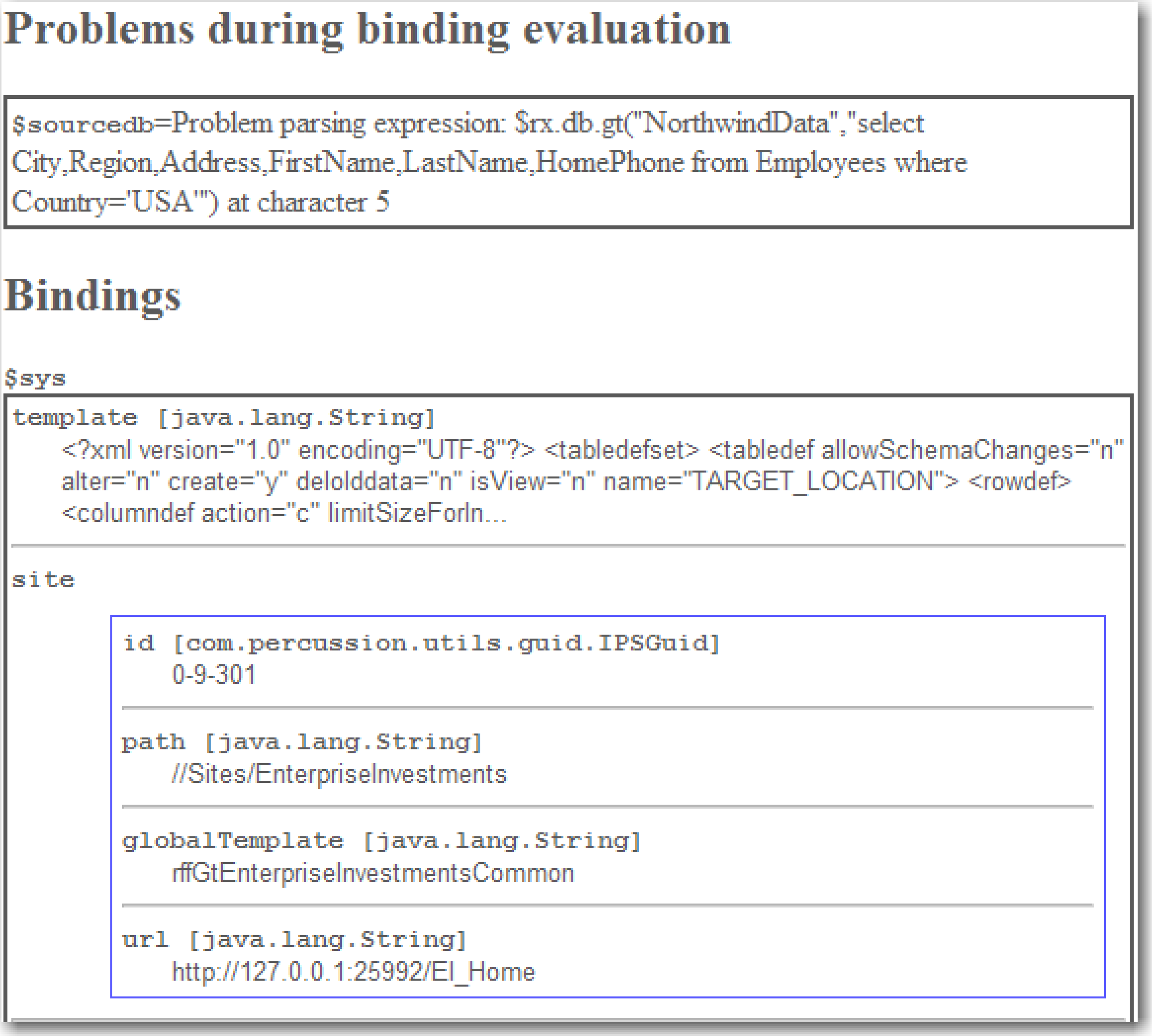
Error reported

Serious problem No value for required expression $db.resource

Please note: More information may be available on the console

Figure 286: Database Template preview showing error results

To find the specific error, view the debug output (in the preview URL, change /assembler/render to /assembler/debug). The bindings that are generating errors will be listed at the top of the debug output.



Problems during binding evaluation

$sourcedb=Problem parsing expression: $rx.db.gt("NorthwindData","select City,Region,Address,FirstName,LastName,HomePhone from Employees where Country='USA'") at character 5

Bindings

$sys

template [java.lang.String]
<?xml version="1.0" encoding="UTF-8"?> <tabledefset> <tabledef allowSchemaChanges="n" alter="n" create="y" delolddata="n" isView="n" name="TARGET_LOCATION"> <rowdef> <columndef action="c" limitSizeForIn...

site

id [com.percussion.utils.guid.IPSGuid]
0-9-301

path [java.lang.String]
//Sites/EnterpriseInvestments

globalTemplate [java.lang.String]
rffGtEnterpriseInvestmentsCommon

url [java.lang.String]
http://127.0.0.1:25992/EI_Home

Figure 287: Database Template debug output showing a binding error.

In this example, the binding function $rx.db.get was specified incorrectly.

Defining the Publishing Configurations for Database Publishing

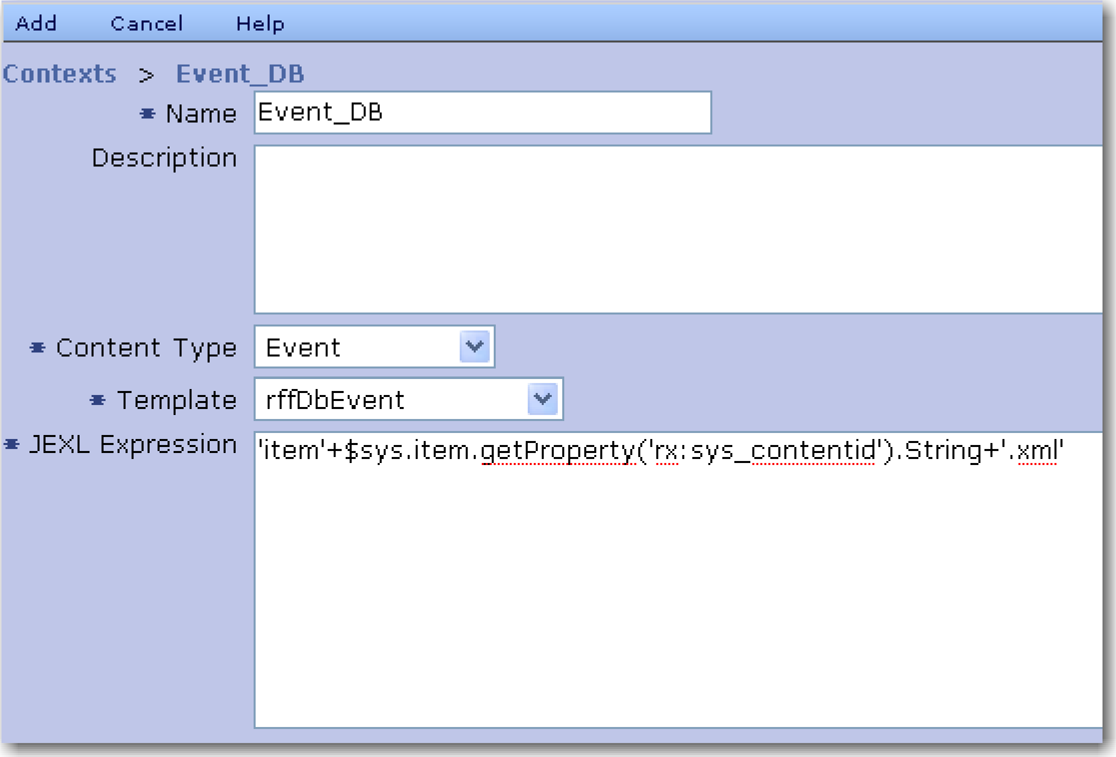
Database publishing requires the standard publishing configurations:

- a Site

  A Site for database publishing requires a minimum of data: a Name and a Rhythmyx Path. None of the other data in a Site configuration is necessary for database publishing.

- a Context with a Location Scheme

  The association of a Content List with an Edition requires a Location Scheme, so you must define a Context and a Location Scheme. The Location Scheme is not used to define a publishing location (the database Template defines the target location for the Content Item data), but is used in the publishing log to define the value of the Location on the Published Item Details dialog. A simple Locations Scheme similar to the following is all that is required:



Contexts > Event_DB

* Name: Event_DB

Description:

* Content Type: Event

* Template: rffDbEvent

* JEXL Expression: 'item'+$sys.item.getProperty('rx:sys_contentid').String+'.xml'

Figure 288: Simple Location Scheme for Database Publishing

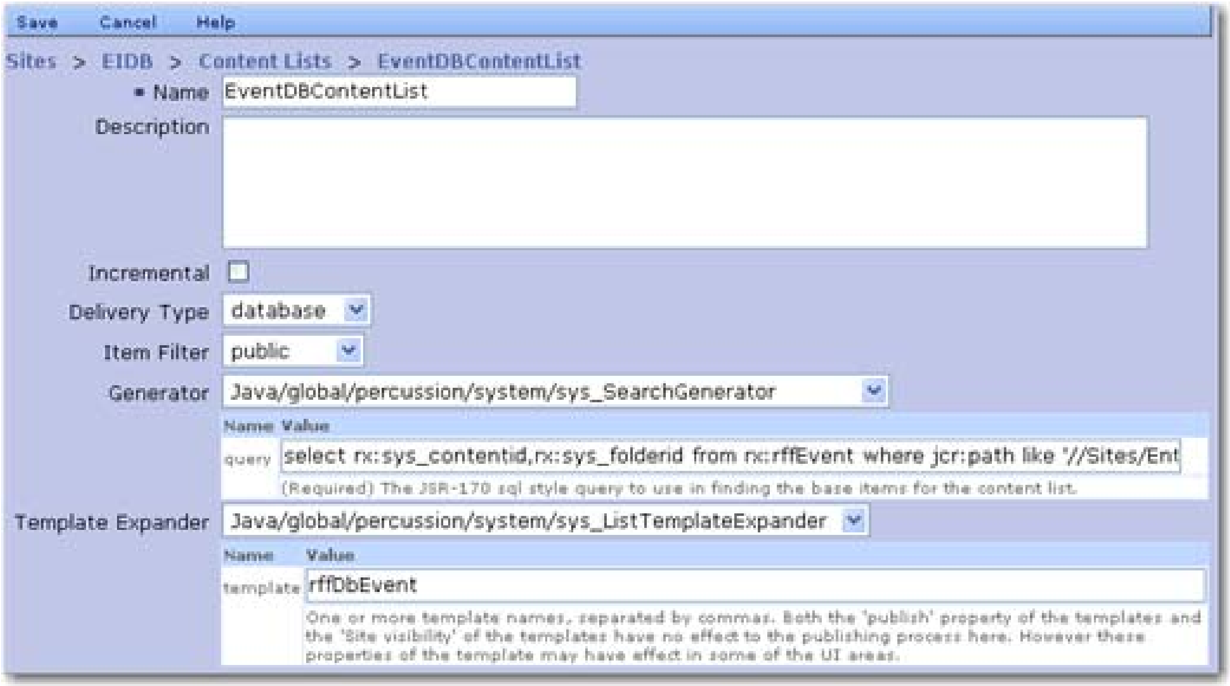
- a Content List

  A Content List for database publishing differs from a Content List for file publishing in two ways:

  - The value selected for the **Delivery Type** drop list must be *database*.

- The value selected for the Template Expander should be *sys_ListTemplateExpander*.

The following graphic illustrates the Content List defined to publish Content Items of the Event Content Type using the rffDbEvent Template we developed earlier in this chapter.



▪ Name: EventDBContentList

Description:

Incremental: □

Delivery Type: database

Item Filter: public

Generator: Java/global/percussion/system/sys_SearchGenerator

Name	Value
query	select rx:sys_contentid,rx:sys_folderid from rx:rffEvent where jcr:path like '//Sites/Ent (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_ListTemplateExpander

Name	Value
template	rffDbEvent One or more template names, separated by commas. Both the 'publish' property of the templates and the 'Site visibility' of the templates have no effect to the publishing process here. However these properties of the template may have effect in some of the UI areas.

Figure 289: Content List defined to publish Event Content Items using the rffDbEvent Template

- an Edition

  Assign the database publishing Content List to the Edition. As the Delivery Context for the association, specify the Context you defined for database publishing.

CHAPTER 11

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

Setting Up SSL

The Secure Socket Layer (SSL) is a protocol that ensures the authenticity of information exchanged over HTTP. It uses a digital certificate, which identifies the sender, and public and private keys for signing messages and encrypting/decrypting data. Enabling SSL ensures secure communication between the Rhythmyx server and the Workbench or Content Explorer.

You can obtain a digital certificate to enable SSL publishing from a recognized certificate authority (such as VeriSign or Thawte) or use a self-signed certificate. Communication will be equally secure in both cases. Certificates from a certificate authority are rather expensive, but will be immediately recognized by your users browsers. If you use a self-signed certificate, the browser will not recognize it and users will be prompted to indicate whether they trust the certificate.

For details about obtaining a digital certificate from a certificate authority, see the Web site of the certificate authority you want to use.

Enabling SSL on the Rhythmyx Server on Windows

To enable SSL on the Rhythmyx server in Microsoft Windows:

1. Obtain a digital certificate from a certificate authority or create your own self-signed certificate.

NOTE: Keystore files must be stored in the directory <Rhythmyxroot>/AppServer/server/rx/conf (or in a subdirectory of that directory).

2. Start the Rhythmyx Server Properties Editor.

   You can only the run the Server Properties Editor from the local Rhythmyx server. Go to your Rhythmyx installation root directory and run RhythmyxServerPropertiesEditor.exe or RhythmyxServerPropertiesEditor.sh.

3. In the SSL KeyStore File field of the Rhythmyx Server Properties Editor, enter the path to the certificate keystore file.
4. In the SSL Key Password, enter the password of the keystore file. Enter the same value in the Confirm SSL Key Password field.
5. In the SSL Port field, specify the Rhythmyx server will use for SSL communication. The default SSL port for Rhythmyx is *9443*.
6. You should not modify the Allowed SSL Ciphers field.
7. Click the [**Save**] button to save your changes.

You must restart the Rhythmyx server for your changes to take effect.

To confirm that you have configured SSL correctly, start a browser and log in to your Rhythmyx server via SSL; for example:

```
https://10.10.10.100:9443/Rhythmyx
```

Remember to use *https* (secure http) as the protocol rather than the unsecure http protocol. If SSL is configured correctly and you entered the address correctly, you should connect to Rhythmyx normally, but via https instead of http. If you use a self-signed certificate, you may be prompted to trust the certificate.

Enabling SSL on the Rhythmyx Server on Solaris and Linux

To enable SSL on the Rhythmyx server on Solaris or Linux,

1. Obtain a digital certificate from a certificate authority or create your own self-signed certificate.

NOTE: Keystore files must be stored in the directory <Rhythmyxroot>/AppServer/server/rx/conf (or in a subdirectory of that directory).

2. Using a simple text editor, open the file <Rhythmyxroot>AppServer/server/rx/deploy/jbossweb-tomcat55.sar/server.xml.
3. Add an SSL <Connector> with the other <Connector> elements:

   ```
   <Connector URIEncoding="UTF-8" acceptCount="100"
   address="${jboss.bind.address}" clientAuth="false"
   connectionTimeout="20000" disableUploadTimeout="true"
   enableLookups="false"
   keystoreFile="${jboss.server.home.dir}/conf/Rhythmyx.keystore"
   keystorePass="mypass" maxHttpHeaderSize="8192"
   maxSpareThreads="75" maxThreads="250" minSpareThreads="25"
   port="9443" scheme="https" secure="true" sslProtocol="TLS"/>
   ```

   where:

   - `keystoreFile="${jboss.server.home.dir}/conf/Rhythmyx.keystore"` specifies the path to your keystore file; use the `${jboss.server.home.dir}` replacement variable to specify the path to the Rhythmyx deployment directory.
   - `keystorePass="mypass"` specifies the keystore password.
   - `port="9443"` specifies the SSL port of the Rhythmyx server. Port 9443 is commonly used as the Rhythmyx port, but you can use any port not already in use.
4. Save server.xml. You must restart the Rhythmyx server for your changes to take effect.

To confirm that you have configured SSL correctly, start a browser and log in to your Rhythmyx server via SSL; for example:

```
https://10.10.10.100:9443/Rhythmyx
```

Remember to use *https* (secure http) as the protocol rather than the unsecure http protocol. If SSL is configured correctly and you entered the address correctly, you should connect to Rhythmyx normally, but via https instead of http. If you use a self-signed certificate, you may be prompted to trust the certificate.

Implementing a Self-signed Certificate

To implement SSL, you can use self-signed certificates, rather than certificates from a recognized certificate authority. This option is less expensive, but your user's browsers will not recognize the self-signed certificates and they will display prompts to trust the certificate.

Use the keytool included with the Rhythmyx JRE to create and manage the certificates.

To generate a self-signed certificate on the Rhythmyx server:

1 Create a certificate file (.csr file) to store your certificate. To create the .csr file, start a simple text editor, such as Notepad or vi, and save an empty file. Use the name of your Rhythmyx system alias. For example, if you name your Rhythmyx server “Rhythmyx”, you would save your file with the name Rhythmyx.csr. Save this file to the directory `<Rhythmyxroot>/JRE/bin`.

2 Open a terminal window and change directories to `<Rhythmyxroot>/JRE/bin`.

Note: Solaris and Linux environments that have Java installed may run keytool commands against the keytool in the installed Java Runtime Engine, which may be different than the Java Runtime Engine included with Rhythmyx. As a result, invalid keystore entries may be created. To resolve this issue, enter the commands as /keytool to ensure that the keytool in the Rhythmyx JRE is run.

3 Create a public/private key pair:

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

Where:

- <keystoreentryalias> is the keystore entry alias;
- <rhythmyxhostname> is the name of the machine where the Rhythmyx server resides;
- <organizationalunit> is the name of the organization unit for which you are issuing the certificate (typically the department responsible for operating the Rhythmyx server);
- <organization> is the organization for which you are issuing the certificate (typically your company name);
- <location> is the name of your location, typically the town or citykey -keystor
- <state> is the name of the state or province;
- <country> is the name of your country;
- <keypassword> is the password for the key pair;
- <storepassword> is the password for the keystore file;
- <keystorefilename> is the name of the keystore file; you can give this file any name, but it must end in `.keystore`. You should also include the path to the Rhythmyx JRE when defining the keystore. If you do not specify the location, the keystore is created in the default location, which is the users home directory.

For example:

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

NOTE: The keystore file should be stored in the directory <Rhythmyxroot>/AppServer/server/rx/conf (or a subdirectory of that directory). You can move the keystore files to that location at any time during this process. To simplify the self-certification process, you may want to wait to move the keystore until you are ready to enable SSL on the Rhythmyx server.

4 Self-certify the key pair:

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

Where:

- <keystoreentryalias> is the keystore entry alias;
- <keypassword> is the password for the key pair;
- <storepassword> is the password for the keystore file;
- <keystorefilename> is the name of the keystore file (including the path if you did not create the keystore file in the default location).

For example:

```
keytool -selfcert -alias Rhythmyx -keypass mypass  -storepass
mypass -keystore <Rhythmyxroot>/JRE/bin/Rhythmyx.keystore
```

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

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

Where:

- <keystoreentryalias> is the keystore entry alias;
- <keypassword> is the password for the key pair;
- <storepassword> is the password for the keystore file;
- <keystorefilename> is the name of the keystore file (including the path if you did not created the keystore file in the default location).
- <certificatefile> is the name of the certificate file; you can give this file any name, but to make it easy to recognize, it is strongly recommended that you use ".csr" as the extension.

For example:

```
keytool -export -alias Rhythmyx -keypass mypass    -storepass
mypass -keystore Rhythmyx.keystore -file
<Rhythmyxroot>/JRE/bin/Rhythmyx.csr
```

6 Repeat Steps 2 through 5 on the remote Rhythmyx Publisher. Give the keystore and certificate files different names than you used for the Rhythmyx server.

7 Copy the Rhythmyx server certificate file (e.g, Rhythmyx.csr) to the Rhythmyx Publisher; copy the Rhythmyx Publisher certificate file to the Rhythmyx server.

8 Import the Publisher certificate into the Rhythmyx server's cacerts file:

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

Where

- <keystoreentryalias> is the keystore entry alias;
- <certificatefile> is the name of the certificate file;
- <cacertpath> is the path to the cacert file into which you want to import the certificate.

Note that the password for the cacert file is *changeit*. You cannot use a different value for this parameter or the command will fail.

For example, the following command would import a key pair with the alias RxPub, stored in a certificate file named RxPub.csr, into the Rhythmyx server's cacerts file:

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

9 Import the Rhythmyx server's keypair into the Rhythmyx Publisher using the command in Step 8.

10 Configure the ***Rhythmyx Server to use SSL*** (see "Enabling SSL on the Rhythmyx Server on Windows" on page 402) and restart. (NOTE: The keystore files should be stored in the directory <Rhythmyxroot>/AppServer/server/rx/conf before enabling SSL.)

APPENDIX II

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	***System Variables*** (see page 408)	General system variables
$rx	***System Functions*** (see page 412)	System methods implemented by system-defined extensions
$nav	***Navon Properties*** (see page 307)	Variables used in Managed Navigation Templates.
$db	***Database Publishing Variables*** (see page 433)	Variables used to implement Database Publishing.
$user	User-defined Variables and Functions	Prefix used for custom variables and functions developed by customers.
$tools	***Velocity Tools Extensions*** (see "Velocity Tool Extensions" on page 434)	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 (<RhythmyxRoot>/Docs.Rhythmyx/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	***IPSAssemblyItem*** (../Javadocs/com/percussion/services/assembly/IPSAssemblyItem.html)	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 *UTF-8*.] Used in: Templates
$sys.crossSiteLink	Boolean	*True* 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	Contains the fields and children of the current Content Item. Each field, and each child, is considered a property of the $sys.item variable. Use the getProperty method to refer to a field of the Content Item. Usually, you will use the getString method to return a string, although you can also return other Java data types. For example, $sys.item.getProperty(displaytitle).getString refers to the value of the displaytitle field of the current Content Item as a string value. For details see the JavaDoc for javax.jcr.Property in the JSR-170 JavaDoc. Used in: Templates and Location Schemes
$sys.mimetype	String	Value of the MIMEType field of the Template. The bindings can override the value specified in the Template. Required for Binary Templates. Default value is *text/html*. Used in: Templates

<table>
<tr><th>Variable</th><th>Type</th><th>Description</th></tr>
<tr><td>$sys.params</td><td>Map<String,String>[]</td><td>Provides access to HTML parameters passed from the Content Item being assembled to the Template. An assembly request includes the following parameters:
<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>sys_path</td><td>Path to the Content Item being assembled.
Required.</td></tr>
<tr><td>sys_siteid</td><td>The Site identifier.
Optional</td></tr>
<tr><td>sys_context</td><td>The name of the Context used to assemble the Content Item.
Required.</td></tr>
<tr><td>sys_itemfilter</td><td>The name of the Item Filter to use to filter Related Content when assembling the Content Item. Required.</td></tr>
<tr><td>sys_template</td><td>The name or ID of the Template to use to assemble the Content Item.
Required.</td></tr>
<tr><td>sys_command</td><td>If specified with the value <i>editrc</i>, the Content Item is assembled in Active Assembly mode, including Active Assembly decorations. If specified as null or any other value, the Content Item is assembled without Active Assembly decorations.
Required.</td></tr>
<tr><td>sys_reinit</td><td>If specified with the value <i>true</i>, the Velocity engine in re-initialized before assembling the Content Item, which causes all macros to be reloaded. If not included or specified with the value false, the Velocity engine is not re-initialized before assembling the Content Item and cached Templates are used.
Optional.</td></tr>
</table>
Used in: Templates and Location Schemes</td></tr>
<tr><td>$sys.path</td><td>String</td><td>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.
Used in: Location Schemes</td></tr>
</table>

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	***IPSGuid*** (../Javadocs/com/percussion/utils/guid/IPSGuid.html)	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*.

- $rx.asmhelper: provide data for use in assembly
- ***$rx.codec*** (on page 417): encode and decode data
- ***$rx.cond*** (on page 418): evaluate conditional statements (Note: This binding function is deprecated)
- ***$rx.db*** (on page 419): Database Publishing
- ***$rx.doc*** (on page 419): process XML and HTML documents
- ***$rx.ext*** (on page 420): call a Rhythmyx extension
- ***$rx.guid*** (on page 421): retrieve GUIDs
- ***$rx.i18n*** (on page 421): 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
- ***$rx.pagination*** (on page 428): provide pagination in assembled Content Items
- ***$rx.session*** (on page 430): returns session IDs
- ***$rx.string*** (on page 430): 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.getTidiedContent
- $rx.asmhelper.getTitle ($sys.item.guid)
- $rx.asmhelper.combine
- $rx.asmhelper.childValues
- $rx.asmhelper.mapValues

$rx.asmhelper.assemble

Returns List<IPSAssemblyResult>

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	***IPSAssemblyItem*** (../Javadocs/com/percussion/services/assembly/IPSAssemblyItem.html)	The Content Item whose related Content Items to assemble. Use $sys.item.
slot	***IPSTemplateSlot*** (Javadocs/com/percussion/services/assembly/IPSTemplateSlot.html)	The Slot whose related Content Items to assemble.
params	Map<String,Object>	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	***IPSGuid*** (../Javadocs/com/percussion/utils/guid/IPSGuid.html)	The GUID of the Content Item whose menus to retrieve. Use $sys.item.guid.
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 *Customizing Content Explorer*.
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 $sys.params.
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.getTidiedContent

Returns String

Used in Templates

Used to return a tidied field value when including an inline Snippet Template the value of a non-rich-text field that includes both HTML code and the ampersand character ("&").

Name	Type	Description
fieldValue	String	The Contents of the field to tidy and return.

Example:

```
$rx.asmhelper.getTidiedContent($sys.item.getProperty(“displaytitle
”).String()
```

returns the value in the sys_displaytitle field tidied to render both HTML and the ampersand character correctly in an inline Snippet.

$rx.asmhelper.getTitle ($sys.item.guid)

Returns String

Used in Templates

Returns the title of the specified Content Item.

$rx.asmhelper.combine

Returns Map<String, Object>

Used in Templates

Generates a new parameter map with 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<String, Object>

Used in Templates

Generates a new parameter map with 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<Object>

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 field to return from the child Fieldset whose values you want to return

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.

Name	Type	Description
mapList	List<Map<String, Object>>	List of maps. Usually derived from the output of the $rx.db.get function.
key	String	Name of the key in the mapList parameter whose values you want to return

$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 *true*.
falsevalue	Object	The value to return if the boolean expression evaluates to *false*.

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	***IPSAssemblyResult*** (Javadocs/com/percussion/services/assembly/IPSAssemblyResult.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.generateToPage
- $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	***IPSAssemblyItem*** (../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.

Name	Type	Description
context	Number	The publishing Context for which to generate the URL.

$rx.location.generateToPage

Returns String

Used in Templates and Location Schemes

Generates the target URL for a page of a paginated item. Used to support pagination.

Name	Type	Description
$assemblyItem	***IPSAssemblyItem*** (../Javadocs/com/percussion/services/assembly/IPSAssemblyItem.html)	The paginated target Content Item to which the URL will point.
$pageNo	String	The page number to which to generate the URL. A value of 0 or null indicates that the Content Item will not be paginated.

$rx.location.generateToPage

Returns String

Used in Templates and Location Schemes

Additional signature of the $rx.location.generateToPage function that allows the user to specify the Template to which to link.

Name	Type	Description
$assemblyItem	***IPSAssemblyItem*** (../Javadocs/com/percussion/services/assembly/IPSAssemblyItem.html)	The paginated target Content Item to which the URL will point.
$template	String	The name of the target Template of the URL.
$pageNo	String	The page number to which to generate the URL. A value of 0 or null indicates that the that the Content Item will not be paginated.

$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.

Name	Type	Description
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.pagination

The methods of this function are used when paginating assembled Content Items.

- $rx.paginate.fieldContentPageCount
- $rx.paginate.getFieldPage
- $rx.paginate.getSlotPage
- $rx.paginate.slotContentPageCount

$rx.paginate.fieldContentPageCount

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	***IPSAssemblyItem*** (../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.paginate.getFieldPage

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	***IPSAssemblyItem*** (../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.paginate.getSlotPage

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	***IPSAssemblyItem*** (../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.paginate.slotContentPageCount

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	***IPSAssemblyItem*** (../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.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.getJSessionID
- $rx.session.getSessionID

$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 `PSSESSIONID`.

$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: ▪ ANCESTOR: a node in the path of the Navon higher than the PARENT Navon node ▪ DESCENDANT: A node in the path after the Navon ▪ NONE: No other category applies ▪ PARENT: The immediate predecessor of the Navon in its path. ▪ SELF: The Navon itself. ▪ SIBLING: Another Navon that shares the PARENT of the Navon.
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) ▪ True: The Navon is a leaf (has no children). ▪ False: The Navon is not a leaf (has children).
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 (<RhythmyxRoot>/Docs.Rhythmyx/JavaDocs/index.html).

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.

<table>
<tr><th>Variable</th><th>Type</th><th>Description</th></tr>
<tr><td>$db.action</td><td>String</td><td>The action to perform. Always a single alphabetic character:
<table>
<tr><th>Value</th><th>Action</th></tr>
<tr><td>r</td><td>Inserts the row. If the row already exists, it is deleted first.</td></tr>
<tr><td>n</td><td>Inserts the row if it does not already exist.</td></tr>
<tr><td>u</td><td>Updates the row if it exists.</td></tr>
<tr><td>d</td><td>Deletes the row if it exists.</td></tr>
</table></td></tr>
<tr><td>$db.origin</td><td>String</td><td>The schema or origin of the target database.</td></tr>
<tr><td>$db.resource</td><td>String</td><td>The JNDI resource on the target server that contains the database.</td></tr>
<tr><td>$db.drivertype</td><td>String</td><td>The type of the driver. The driver type is the portion of the JDBC URL after <i>jdbc:</i> and before the database specification.</td></tr>
<tr><td>$db.parent</td><td>String</td><td>The name of the parent table.</td></tr>
<tr><td>$db.child [<i>n</i>]</td><td>String</td><td>The name of the child table.</td></tr>
<tr><td>$row.<i>columnname</i></td><td>String or Number</td><td>The name of a specific column in a row of the parent database.</td></tr>
<tr><td>$child[<i>n</i>].<i>columnname</i>[<i>i</i>]</td><td>String or Number</td><td>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.</td></tr>
<tr><td>$row.$encoding.<i>columnname</i></td><td>String</td><td>The encoding of the specified column. Options are empty or <i>base64</i>.</td></tr>
<tr><td>$child[n].$encoding.columnname</td><td>String</td><td>The encoding of the specified child column. Options are empty or <i>base64</i>.</td></tr>
</table>

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:

Tool	Class
$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.

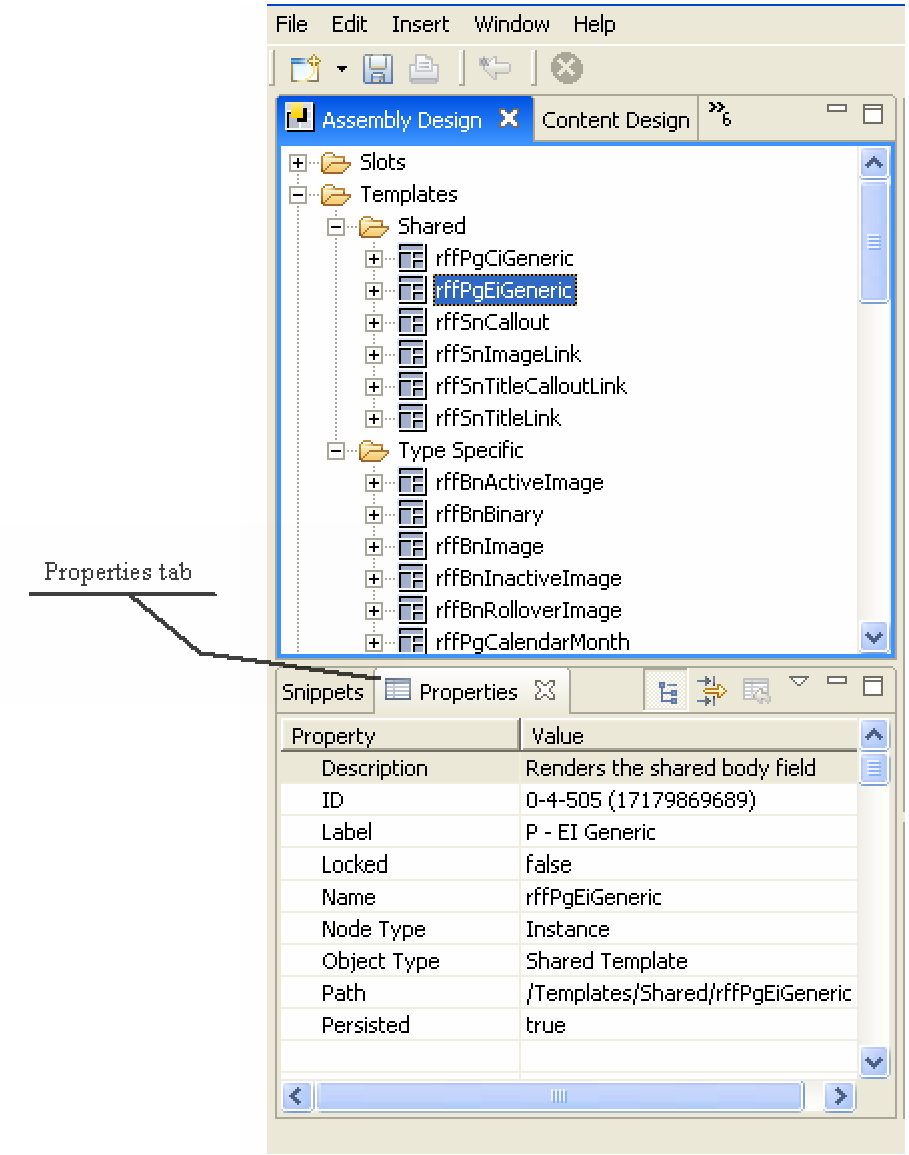
An Assembly Item is a container for all data required to assemble a Content Item output, such as the Site and the Template, including the Assembly Node.

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 III

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:



Properties tab

Property	Value
Description	Renders the shared body field
ID	0-4-505 (17179869689)
Label	P - EI Generic
Locked	false
Name	rffPgEiGeneric
Node Type	Instance
Object Type	Shared Template
Path	/Templates/Shared/rffPgEiGeneric
Persisted	true

Figure 290: 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 IV

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. Names 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 V

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 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	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

Name	Label	Hidden	Description	Control	Data Type/ StorageSize
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

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

Name	Label	Hidden	Description	Control	Data Type/ StorageSize
item_file_attachment_ size	File Size:		Field that stores the size of the binary file.	sys_EditBox	integer
item_file_attachment_t ype	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.

Fields

Source	Name	Label	Control Name	Occur	Data Type	Format
system	sys_title	System Title:	sys_EditBox	required	text	255
shared	shared/displaytitle	Title:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	required	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	16
system	sys_currentview		sys_HiddenInput	optional	text	
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
query	rffAutoPressRelease2005
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	255
shared	shared/displaytitle	Title:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	required	datetime	none
shared	shared/callout	Callout:	sys_EditLive	required	text	max
local	placeholder	Placeholder	sys_HiddenInput	required	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	16
system	sys_currentview		sys_HiddenInput	optional	text	
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)

callout	*Free form content goes here*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

Templates

- rffSnCallout

Workflows:

- Standard
- Simple (Default)

"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	255
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	calendar_start	Calendar Date:	sys_CalendarSimple	required	datetime	none
local	query	Query:	sys_DropDownSingle	optional	text	255

Source	Name	Label	Control Name	Occur	Data Type	Format
shared	shared/filename *Note that by default 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	16
system	sys_currentview		sys_HiddenInput	optional	text	
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
calendar_start	sys_DateFormat(yyyy-MM-dd, ,)
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	255
shared	shared/displaytitle	Title:	sys_EditBox	required	text	512
local	firstname	First Name:	sys_EditBox	optional	text	100
local	middlename	Middle Name:	sys_EditBox	optional	text	100
local	lastname	Last Name:	sys_EditBox	optional	text	100
local	dept	Dept:	sys_EditBox	optional	text	255
local	address1	Address 1:	sys_EditBox	optional	text	200
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	required	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	16
system	sys_currentview		sys_HiddenInput	optional	text	
shared	shared/webdavowner * hidden by default	WebDAV Owner:	sys_TextArea	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

Templates

- rffSnNameAndAddress
- rffSnNameAndEmail

Workflows:

- Standard
- Simple (Default)

"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	255
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

Source	Name	Label	Control Name	Occur	Data Type	Format
shared	sharedbody	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	optional	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	16
system	sys_currentview		sys_HiddenInput	optional	text	
shared	shared/webdavowner	WebDAV Owner	sys_TextArea	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
event_start	sys_DateFormat(yyyy-MM-dd, ,)
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	255
shared	shared/displaytitle	Title:	sys_EditBox	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	required	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	16
system	sys_currentview		sys_HiddenInput	optional	text	
shared	shared/webdavowner	WebDAV Owner:	sys_TextArea	optional	text	255
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
url	#
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

Templates

- rffSnLink

Workflows:

- Standard
- Simple (Default)

"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	Hid-den	Label	Control Name	Occur	Data Type	Size
system	sys_title		System Title:	sys_EditBox	required	text	255
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		Binary Filename:	sys_EditBox	required	text	512
shared	sharedbinary/item_file_attachment_type		File Type:	sys_EditBox	optional	text	256
local	file_category		File Category:	sys_DropDownSingle	optional	text	50
local	filename		File Name:	sys_EditBox	optional	text	512
shared	sharedbinary/item_file_attachment_ext	yes	File Extension:	sys_EditBox	required	text	50
system	sys_suffix	yes	Suffix:	sys_EditBox	optional	text	50
shared	sharedbinary/item_file_attachment_size	yes	File Size:	sys_EditBox	optional	integer	none

Source	Name	Hid-den	Label	Control Name	Occur	Data Type	Size
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	16
system	sys_currentview			sys_HiddenInput	optional	text	
shared	shared/ webdavowner	yes	WebDAV Owner:	sys_textArea	optional	text	255
system	sys_hibernateVersion			sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	255
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
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	16
system	sys_currentview		sys_HiddenInput	optional	text	
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none
local	usage	Usage:	sys_DropDownSingle	required	text	1

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community

Field	Value
sys_lang	PSXUserContext/User/SessionObject/sys_lang
usage	*N*

Templates

- rffDsEIGenericSelector
- rffPgCIGeneric
- rffPgEIGeneric
- rffPgEIGenericCategoryPage
- 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	255
shared	shared/displaytitle	Title:	sys_HiddenInput	required	text	512
system	sys_contentstartdate	Start Date:	sys_CalendarSimple	required	datetime	none

Source	Name	Label	Control Name	Occur	Data Type	Format
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_HiddenInput	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	File Name	sys_EditBox	optional	text	50
shared	shared/body	Body:	sys_HiddenInput	optional	text	max
shared	shared/filename *Note that by default a this field is hidden	File Name:	sys_HiddenInput	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	
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
displaytitle	*Default Display Title*
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

Templates

- rffDsEIGenericSelector

- rffPgCIGeneric
- rffPgCIGenericWord
- rffPgEIGeneric
- rffPgEIGenericCategoryPage
- rffSnCallout
- rffSnImageLink
- 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	255
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/body	Body:	sys_EditLive	optional	text	max

Source	Name	Label	Control Name	Occur	Data Type	Format
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	16
system	sys_currentview		sys_HiddenInput	optional	text	
local	placeholder	Placeholder	sys_HiddenInput	optional	text	1
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
filename	*index*
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

Templates

- rffPgCIHome
- rffPgEIHome
- 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	Hid-den	Label	Control Name	Occur	Data Type	Size
system	sys_title		System Title:	sys_EditBox	required	text	255
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	50
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/ img_alt		Image Alt Text:	sys_EditBox	required	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	yes	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

Source	Name	Hid-den	Label	Control Name	Occur	Data Type	Size
shared	sharedimage/ img2_ext	yes	Mini Extension:	sys_EditBox	optional	text	50
shared	sharedimage/ img2_type	yes	Mini Mime Type:	sys_EditBox	optional	text	256
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	yes	Community:	sys_DropDownSingle	required	integer	none
system	sys_workflowid	yes	Workflow:	sys_DropDownSingle	required	integer	none
system	sys_lang	yes	Locale:	sys_DropDownSingle	required	text	16
system	sys_currentview			sys_HiddenInput	optional	text	
shared	shared/ webdavowner	yes	WebDAV Owner:	sys_textArea	optional	text	255
system	sys_hibernateVersion			sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
sys_suffix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	Size
system	sys_title	System Title:	sys_EditBox	required	text	255
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	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	optional	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	16
system	sys_currentview		sys_HiddenInput	optional	text	

Source	Name	Label	Control Name	Occur	Data Type	Size
system	sys_hibernateVersion		sys_HiddenInput	optional	integer	none

Default Values

Field	Value
sys_contentstartdate	sys_DateFormat(yyyy-MM-dd,)
body	*Enter body here*
sys_sufix	*.html*
sys_communityid	PSXUserContext/User/SessionObject/sys_community
sys_lang	PSXUserContext/User/SessionObject/sys_lang

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	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	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."
Expire	Archive	Manual Transition, 1 Approval	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	No

Name	To State	Details	Notification
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

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
Return to Public	Public	Manual Transition, 1 Approval, Admin only	No

State: Archive

Publishable Value: Archive

Sort Order: 50

Assigned Roles

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

Transitions

Name	To State	Details	Notification
Republish	Public	Manual Transition, 1 Approval, Admin or Web Admin only	No
Revive	Draft	Manual Transition, 1 Approval	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

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

Transitions

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 or Web_Admin only	No

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	Yes	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

State: Pending

Publishable Value: Unpublish

Sort Order: 40

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
Editor	Reader	No	No	No
QA	Reader	No	No	No

Transitions

Name	To State	Details	Notification
Age to Public	Public	Aging Transition, Triggered by Publish Date of Content Item	No
Force to Public	Public	Manual Transition, 1 Approval	No

State: Public

Publishable Value: Publish

Sort Order: 50

Assigned Roles

Roles	Assignment Type	Ad Hoc	Show in Inbox	Receive Notifications?
Web Admin	Assignee	No	No	Yes
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."
Expire	Archive	Manual Transition, 1 Approval	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	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: 60

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
Editor	Reader	No	No	No
QA	Assignee	No	No	No

Transitions

Name	To State	Details	Notification
Return to Public	Public	Manual Transition, 1 Approval	No

State: Archive

Publishable Value: Archive

Sort Order: 70

Assigned Roles

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

Transitions

Name	To State	Details	Notification
Republish	Public	Manual Transition, 1 Approval, Admin or Web Admin only	No
Revive	Draft	Manual Transition, 1 Approval	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://127.0.0.1:9992/CI_Home	http://127.0.0.1:9992/EI_Home
Site Home Page URL	http://localhost:9992/Rhythmyx/assembler/render?sys_revision=8&sys_siteid=303&sys_itemfilter=preview&sys_template=rffPgCiHome&sys_contentid=551&sys_folderid=523&sys_context=0	http://localhost:9992/Rhythmyx/assembler/render?sys_revision=4&sys_siteid=301&sys_itemfilter=preview&sys_template=rffPgEiHome&sys_contentid=466&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	CI Global Template	EI Global Template

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

Parameter	Value
sys_FlagValues	u

FastForward Content Lists

FastForward is delivered with the following Content Lists

rffCiFullBinary

Site Root Full for Binary Content Types - Corporate Investments

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

Incremental: no

rffCiFullNonBinary

Site Root Full for Non-Binary Content Types - Corporate Investments

Generator: sys_SearchGenerator

Query: 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/CorporateInvestments%'

Template Expander: sys_SiteTemplateExpander

Site ID: 303

Item Filter: Public

Incremental: no

rffCiIncremental

Site Root Incremental - Corporate Investments

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

Incremental: yes

rffEiFullBinary

Site Root Full for Binary Content Types - Enterprise Investments

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

Incremental: no

rffEiFullNonBinary

Site Root Full for Non-Binary Content Types - Enterprise Investments

Generator: sys_SearchGenerator

Query: 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%'

Template Expander: sys_SiteTemplateExpander

Site ID: 301

Item Filter: Public

Incremental: no

rffEiIncremental

Site Root Incremental - EnterpriseInvestments

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: 301

Item Filter: Public

Incremental: yes

rffEiPublishNow

Site Publish Now - Enterprise Investments

Generator: sys_SelectedItemsGenerator

Template Expander: sys_SiteTemplateExpander

Site ID: 301

default_template: unspecified - include all default Templates

Item Filter: sitefolder

Incremental: no

APPENDIX VI

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: ▪ None ▪ Reader ▪ Assignee ▪ Admin
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.

Name	Label	Searchable	Comments
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 • 164
#displayfield • 131
#field • 131
#inner macro • 175
#slot macros • 156, 164

$

$rx Functions • 417, 418, 419, 420, 421, 422, 424, 427, 430

A

Absolute Aging • 44
Access Control Lists • 67, 76
Accessibility • 395
ACL • 76, See Access Control Lists
Active Assembly Tutorial • 395
Ad Hoc Assignment • 55
Administrator • 36
Aging • 44, 61, 64
Allowed Content • 120, 121
Ancestor • 281
Ancestor-sibling • 281
Approvals (on Transitions) • 62
Assembly Binding Variables • 435
Automated Index • See Automated Slots
Automated Slots • 192, 196, 198, 199
  and Incremental Publishing • 192
Axis • 281

B

Binary
  fields • 98
  files • 144
  Templates • 125, 144
  treat data as • 87
Binding Variables • 434
Bindings • 141, 142, 144, 147, 151, 198, 307
  variables • 307, 407, 408, 412, 433, 434, 435

C

Cascading Stylesheets • 303, 305
Cataloging • 28
Child Field Set • 250, 252
Child Table • 164
  and database publishing • 382, 385
Comments • 60
Communities • 33, 37
Community Visibility • 37, 231
Content Editor • 483, See Content Type
Content Finder • 120, 121
Content List • 314, 318, 319
  Database Publishing • 392
Content Type • 215, 217, 219, 220, 222, 223, 227, 235, 236, 239, 240, 241, 243, 246, 248, 250, 251, 252, 271, 272, 291
  child field set • 250, 252
  definition • 215
  including shared and system fields • 227
  overriding a shared field • 251
  properties • 243
Context Variables • 180, 182
Contexts • 321, 327, 328
Controls • 87, 95, 98
CSS • See Cascading Stylesheets

D

Data bindings • See Bindings
Data Type • 87, 98
Database Publishing • 363, 364, 368, 373, 378, 382, 389, 392
  Binding Variables • 373, 378, 382, 385, 433
  Content List • 392
  Context • 392
  datasource • 368
  Edition • 392
  modeling and design • 363, 365
  Oracle • 363
  Site • 392
  Templates • 125, 373, 378, 382, 385
  troubleshooting • 389
Datasource • 368
Default Security Provider • 21, 23
Default Value • 87, 98
Demand Publishing • 342
Descendant • 281
Development Infrastructure • 19
Display Formats • 395
DOCTYPE • 126

E

Editions • 309, 330, 334, 342, 392

Database Publishing • 392
Demand Publishing • 342
Escalating Notices • 64

F

FastForward Implementation Plan • 447, 448, 452, 471, 479
Fields • 81, 87, 98, 104, 105, 108, 227, 241, 251
binary • 98
Field and Field Set Editor • 85
read-only • 104
transformations • 104
validation • 104, 108
visibility • 105
Folders • 76
FROM Clause • 196

G

Global Template • 169, 173, 175, 178, 180, 182
adding Context Variables • 180, 182
adding Managed Navigation • 178

H

HTML
preparing for use in Templates • 126
search properties • 87
Hypertext Links • 147

I

Icons • 229, 243, 252
Image Links • 151
Implementation Guide • 11
Implementation Roadmap • 12, 14
Incremental Publishing • 311, 319, 330
and Automated Slots • 192, 334
Content List • 319
Incremental Publishing/Edition • 334
Initial State • 42
Inline scripting • 126
Internationalization • 395
Item Validation • 215, 248, 271

J

JSR-170 • 192, 196

K

Keywords • 291

L

Label, and internationalization • 87
Landing Page • 285
Lifecycle Analysis • 395
LIKE Operator • 196
Links • 147, 151
List Control • 95
Local Content • 175
Local Fields • 241
Local Template • 125, See Templates
Localization • 395
Location Schemes • 321, 327, 328

M

Macros • See Velocity
Managed Navigation • 74, 120, 125, 279, 281, 285, 286, 287, 290, 291, 295, 296, 297, 303, 305, 306, 307
adding to a Global Template • 178
adding to a Site • 74
bindings • 307
Cascading Stylesheets • 303, 305, 306
Customizing Look and Feel • 296, 297, 303, 305, 306, 307
maintaining • 285, 286, 287, 290, 291
Slots • 295
Templates • 296, 297, 307
Max Results • 196
Members • 28
Microsoft Word • 395
Migration • 275
Mime type • 87, 95, 98, 251
Mnemonic • 87, 98

N

Naming Conventions • 439, 440, 441, 442, 443, 444, 445
Navigation Sections
merging • 290
Merging Navigation Sections • 290
splitting • 287
Splitting Navigation Sections • 287
NavImage • 279, 281, 286
Navon • 74, 279, 285, 287, 290
Binding Variables • 307
landing page • 285
merging • 290
splitting • 287
NavTree • 67, 74, 281, 285
Notification • 46, 47, 55

associating with a Transition • 47
creating • 46
enabling for a Role • 55

O

Object Properties • 437
Oracle • 363
ORDER BY Clause • 196

P

Page Template • 125, 153, 156, 164
Post-processing • 271, 272
Pre-processing • 271, 272
Publish Now • See Demand Publishing
Publishing • 309, 311, 313, 314, 318, 319, 321, 327, 328, 330, 334, 337, 338, 340, 342, 344, 349, 351, 353, 356, 357, 358, 359, 360, 361, 363, 364, 365, 368, 373, 378, 382, 385, 389, 392
Database Publishing • See Database Publishing
FTP • 356, 357, 358, 359, 360, 361
Local Web Server • 353
setting up Site • 67, 69
SFTP • 356, 359, 360, 361
Publishing Root Location • 71

R

Read-only rules • 104
Repeating Transitions • 61
Rich Text Control • 87
Roadmap • 12, 14
Roles • 24, 28
Ad Hoc assignment • 55
adding users to • 28
assigning to a State • 42
Community • 24, 33
Notification • 55
Workflow • 24, 38
Root • 281

S

Scripting • 126
Search (field properties) • 87, 98
Searches • 395
Security Provider
Default Security Provider • 21, 23
SELECT Clause • 196
Self • 281
Server Administrator • 18
Shared Fields • 81, 83, 84, 85, 87, 98
creating Shared Field Sets • 87, 98
including in a Content Editor • 227
including shared/system • 227
overriding • 251
Shared Templates • 125
Show in Preview • 87, 98
Sibling • 281
Sites • 67, 69, 71, 73
adding Managed Navigation • 74
adding Subfolders • 73
creating Site Root • 69
Database Publishing • 392
registering • 71
Slots • 113, 120, 124, 192, 196, 198, 199, 295
Automated • 113, 192, 196, 198, 199
controlling Contents • 124
Creating • 120, 121, 124
Managed Navigation • 113, 295
queries • 196, 199
Standard • 120, 121, 124
Snippets
Complex • 147, 151
Snippet Drawer • 131
Snippet Template • 127, 128, 131
Standard Slots • 120, 121, 124
States • 39, 40, 42, 57
assigning Roles • 40
creating • 39
Initial State • 42
Static Files • 180, 182
Storage size • 87, 95, 98
System Fields • 227

T

Templates • 113, 125, 126, 127, 128, 131, 139, 144, 147, 151, 153, 156, 164, 169, 173, 175, 178, 180, 182, 187, 296, 297

binary • 144
debugging • 139
dispatch • 187, 188, 190
Global Templates • 169, 173, 175, 178, 180, 182
HTML, preparing for use in Templates • 126
Local • 125
Managed Navigation • 296, 297, 307
page • 153, 156, 164
Page without Global • 183, 184, 186
Snippet • 127, 128, 131, 147, 151
Troubleshooting • 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 211, 212, 213
Text Extraction • 275
Text, treating as binary • 87
Transformation • 104, 271
Transition • 42, 43, 44, 58, 60, See Aging
approvals • 62
comments • 60
creating • 42, 43, 44
Treat data as binary • 87
Troubleshooting • 200
Tutorial • 395
Type-specific Template • 125

U

Unpublishing • 39, 311, 314, 330, 444, 479
USERLOGIN • 21, 23
Users
adding to a Role • 28
adding to default Security Provider • 21, 23

V

Validation • 104, 108, 271
Variable Parameters • 198, 199
Variable Selectors • 305, 306
Variables • 407
Velocity • 125, 131, 156, 434
Views • 395
Visibility • 104, 105

W

Web Services • 395
WebDAV • 275, 395
WebImageFX • 249
WHERE Clause • 196
Word • See Microsoft Word
Workbench • 16
Workflow • 35, 36, 37, 38, 39, 40, 42, 43, 44, 46, 53, 54, 55, 57, 58, 60, 61, 62, 64, 471, 472, 475, See Transition, See States
associating with a Community • 37
copying • 54
creating • 36
defined • 35
Roles • 38