Rhythmyx

Implementing Database Publishing

Version 5.7

Printed on 17 October, 2005

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

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

Contents

CHAPTER 1

Prerequisites

It is recommended that you:

- Install the archive file:
  <Rhythmyx root>/Samples/DeprecatedSamples.pda

  This file includes sample Rhythmyx components and applications which facilitate database publishing and are referenced in this document. For help installing the archive file, see the *Installing Rhythmyx* document.

  NOTE: Do not install FastForward if you plan to install the archive file; FastForward components and archive components may conflict with one another.
- Implement filesystem publishing before you attempt to implement database publishing. The procedure for database publishing builds on the procedure for filesystem publishing.
- Read all chapters in this document. To successfully perform database publishing, you must understand the steps and methods described in Chapters One and Two and how to apply them to an example as described in Chapter Three.

CHAPTER 2

Publishing to a Database

The Database Publisher lets you to publish both raw content and formatted content into any database schema. This allows Rhythmyx to support any delivery environment. For example, a Rhythmyx environment could simultaneously support content delivery through Net/IIS, BEA Weblogic Portal, or iPlanet/Apache Web Server, and custom built database driven delivery applications. The Database Publisher simplifies this by letting you map raw data or formatted content snippets to any database schema using the Rhythmyx mapper. Mapping between content fields and target database columns does not have to be one to one; after retrieving content from the Rhythmyx repository, the Database Publisher can assemble content fields into formatted variants and insert them into columns in the target database.

For example, portals are common in today's Web delivery environment, either custom built or from companies such as BEA, Plumtree or Yahoo! Each portal has its own unique schema to which the Database Publisher can publish. Each schema is a mixture of raw content (for formatting by the portal), metadata (for maintaining structure and indexing in the portal), formatted content snippets that the portal developer can simply place within a page, or a finished document (such as a PDF) that can then be downloaded from the site. The delivery application can then act upon and process the content as it normally would using its own local datastore without having to integrate with another database for content it needs.

Install the Database Publisher by selecting a Custom Publisher setup type and the Database Publisher feature when you install the Rhythmyx suite. (See the *Installing Rhythmyx* document for detailed installation instructions.) Rhythmyx installs the Database Publisher plugin into `<Rhythmyxroot>/AppServer/webapps/RxServices/WEB-INF/lib/rxdbpublisher.jar`. The Database Publisher plugin publishes all content lists that have a **deliverytype** parameter of `database`.

Overview of the Database Publisher Plugin

Database publishing runs as part of Rhythmyx's Publishing system, using the same building blocks, including Sites, Editions, and Content Lists. To use the Database Publisher, you must create a special Data Publishing XML variant in the Assembly system for each type of content to be published to a database. Unlike standard file-based Publishing, which can publish any type of Assembly variant, the Database Publisher can only publish these data variants, which are XML only, include RDBMS schema information, and conform to sys_Database Publisher.dtd. To publish a formatted content variant, you must insert it within the proper nodes of the database publishing XML variant. A typical XML variant for database publishing defines a parent table and one or more child tables. It includes a <datapublisher> node that defines the structure of the tables in <tabledefset> nodes. The child tables include <foreignkey> elements that define the fields for matching them with the parent table.

Within the <datapublisher> node, the XML variant includes a <tabledefset> node defining the table schema we are publishing to and a <tabledataset> node defining the table data to be published. The <tabledataset> contains a <table> node for the parent table data. The <table> node includes one or more <row> nodes that contain a <column> element for each column in the parent table. The Database Publisher inserts or updates these rows into the target database. The <column> element can have an @encoding attribute that tells the Database Publisher in what encoding format it receives the content. The Database Publisher decodes the encoded content before storing it in the database.

For each child table, the <row> node also includes a <childtable> element with <row> and <column> elements. The Database Publisher updates, inserts or deletes rows in the child table according to the <row> specifications.

Steps for Database Publishing

Before you configure database publishing, you must know which parent and child tables in your target database you will be using and their structure, including their primary and foreign keys. If you are publishing to parent-child tables, you must define a primary key relationship between the tables. You must also know whether you want to publish the data in each column as raw content or a formatted variant. If you want to publish the content as a formatted variant, you must know whether you want to set the @encoding attribute to *escaped* or *base64* (see ***Encoding Data*** (on page 16) for more information). Most often, non text data is base64 encoded.

Once you have identified the data you want to publish to your database and its format, complete the following steps to set up database publishing. These steps include creating the HTML file for each table that Rhythmyx's XSpLit will transform into an XML file and an XSL stylesheet, and using the Table Definition Builder to build your table definition XML with these tables. See ***Example: Publishing the News Content Type to a Database*** (on page 33) for a detailed example that follows these steps. The steps below reference specific topics in this example as well as other topics in this document and other Rhythmyx documents that provide additional reference and information.

To set up Rhythmyx to publish to a database, include the following steps when you set up Publishing in Rhythmyx:

1. In addition to following the standard procedure in "Registering a Publisher" in the *Implementing Publishing in Rhythmyx* document:
   - Define any of the optional parameters that your Database Publisher requires. See "Descriptions of Publisher Parameters" in the *Implementing Publishing in Rhythmyx* document for descriptions of optional parameters.
   - Follow the steps in "Implementing a User-Created Plug-in" in the *Implementing Publishing in Rhythmyx* document to register the plugin. Enter the *Name database* and the *Value com.percussion.publisher.client.PSDatabasePublisherHandler*.
2. In addition to following the standard procedure for creating a Rhythmyx content assembler application (see "Creating Assembly Applications" in the *Implementing Content Editors and Content Assembly* document), create a Database Assembler application:

   a) Create an HTML file for each parent table and child table to be published.

   See Parent Table HTML Markup Rules and ***Child Table HTML Markup*** (on page 13) for more information.

   See ***Marking Up HTML Files for the News Target Database*** (on page 42) for an example.

   b) Drag and drop the files on the Rhythmyx Workbench to create query resources. Add the sys_DatabasePublisher exit to the parent table resource. This exit looks up the table definition and produces an output document in the correct format for the database.

   See ***sys_DatabasePublisher*** (on page 14) and the topics "To Create a Query from an HTML Page" in the Rhythmyx CMS Online Help for information.

See ***Creating the News Database Application*** (see "Creating the News Database Assembler Application" on page 44) for an example.

c) For each resource, attach tables, define the Selector, and complete the Mapper.

See the topic "Defining How Data Maps to the Resource" in the Rhythmyx CMS Online Help for information.

See ***Mapping the News content and contentContact Resources*** (on page 45) for an example.

d) If the database has child tables that return multiple rows but does not specify <rowid> elements, specify the repetition of XML columns.

See ***Specifying the Repetition of XML Columns*** (on page 15) for more information.

See ***Specifying Column Repetition in News Resources*** (on page 51) for an example.

e) If your assembler will publish any content that should be Base64 encoded, add a Base64 encoding UDF to the mapper in your XML resource. The UDF converts formatted content to Base64 strings to prevent parsing errors.

For help determining which fields should be Base64 encoded and which Base64 encoding UDF to use, see ***Encoding Data*** (on page 16).

f) Use the Table Definition Builder to create a table definition file for the database. You will add the table definition XML file to the assembler application as a static file (copy it into the database assembler application folder). To see the definitions of attributes and elements in the Table Definition file that you create, see <Rhythmyx Root>/DTD/sys_Tabledef.dtd.

See Table Definition Builder for information.

See ***Creating the News Database Table Definition*** (on page 52) for an example.

g) Register the Database Assembly Variant.

See Registering Database Assembly Variant for an example.

3 If you are publishing Variants (rather than raw data) to the body fields(s) in your target database table:

a) Develop resource files, Variants, and Slots if they do not already exist.
See Developing Variants, Resource, and Slots for an example.

b) Create Variant assembler applications in the Rhythmyx Workbench if they do not already exist.
See ***Variant Assembler Applications*** (on page 58) for an example.

4 Build a content list application. Map the *deliverytype* parameter to *database*, the Name you assigned to the plugin parameter when registering the Database Publisher plugin.

See the section "Creating a Content List Application" in the document *Implementing Publishing in Rhythmy*x for more information.

See ***Building the News Content List Application*** (on page 60) for an example.

5 Register the Publisher, Site, Edition and Content Lists in the Content Explorer.
See Registering News Publishing Components for an example.

6 Set up Data Sources in your server.

CHAPTER 3

Database Publishing Reference and Guide

This chapter describes configuration options and procedures for some of the steps listed in ***Steps for Database Publishing*** (on page 7). Read the topics in this chapter to understand how you must configure the components of Database Publishing and the choices that you want to make to create the optimal setup for your system.

To see a step by step example that guides you through the procedure outlined in ***Steps for Database Publishing*** (on page 7), see the chapter ***Example: Publishing the News Content Type to Database*** (see "Example: Publishing the News Content Type to a Database" on page 33).

Parent Table HTML Markup Rules

You must create an HTML file for the main table (or parent table) in your target database (and all sub-tables associated with it, or ***child tables*** (see "Child Table HTML Markup" on page 13). Drag and drop the HTML files onto the Rhythmyx Workbench to create query resources that you can map to the correct backend information for content assembly

The following is a sample parent table HTML markup.

```
<html>
   <head>
      <title>CONTENT: parent table markup</title>
   </head>
   <body>
      <!-- The complete table definition, required in parent tables. -->
      <span id="psx-tabledefset/@lookup" />
      <!-- The database definitions. @drivertype is always required.
           See the explanation below for the requirements regarding the
other attributes -->
      <span id="psx-database/@dbname" />
      <span id="psx-database/@drivertype" />
       <span id="psx-database/@resourceName" />
       <span id="psx-database/@origin" />
      <!-- The table definition. -->
      <span id="psx-table/@name" />
      <span id="psx-table/row/ID" />
      <span id="psx-table/row/VERSION" />
      <span id="psx-table/row/AUTHOR" />
       <span id="psx-table/row/LOCKER" />
      <span id="psx-table/row/CREATED" />
      <span id="psx-table/row/MODIFIED" />
      <span id="psx-table/row/ABSTRACT" />
      <span id="psx-table/row/ABSTRACT/@encoding" />
      <span id="psx-table/row/BODY_HTML" />
      <span id="psx-table/row/BODY_HTML/@encoding" />
      <span id="psx-table/row/BODY_PDF" />
      <span id="psx-table/row/BODY_PDF/@encoding" />
      <!-- Include all child tables, optional -->
      <span id="psx-table/row/children/contentContact/@lookup" />
   </body>
</html>
```

Each group of span tags in the markup includes HTML elements that form part of the table definition:

- psx-tabledefset - You must include the psx-tabledefset element to define the table definition. The @lookup attribute is the location of the table definition file, usually created by the table definition builder. Normally this file is stored in the application directory of the content assembler.
- psx-database - You must include the psx-database element to provide database information. The @drivertype attribute holds the driver type used (for example: jtds:sqlserver or oracle:thin). The other three attributes differ depending on the database type and whether you are publishing to a single instance or multiple instances of the database.

Oracle and DB2, publishing to single or multiple instances of the database:

- The @resourceName attribute holds the name of the resource (It must match the Resource name and the ResourceParams name in the Publisher server configuration file, server.xml.
- The @origin attribute holds the name of the target database schema. If @origin exists, the @dbname attribute is optional and Rhythmyx ignores it.
- If @origin does not exist, @dbname is required and Rhythmyx uses its value for the database schema.
- If the user or username parameter in the Publisher server configuration file, server.xml, is not the same as the schema name, use the @origin attribute.

The attributes take the values of attributes of the Resource element in the context.xml file.

Example Resource element for Oracle:

```
<Resource name="jdbc/TEST2ORACLE" auth="Container"
type="javax.sql.DataSource" username="TEST2ORACLE"
password="PASSWORD"
driverClassName="oracle.jdbc.driver.OracleDriver="oracle:thin"
 url="jdbc:oracle:thin:@localhost:1521:UTF8" DatabaseName=""/>
```

Example Resource element for DB2:

```
<Resource name="jdbc/dbpubtarget" auth="Container"
type="javax.sql.DataSource"
driver="db2" driverClassName="com.ibm.db2.jcc.DB2Driver"
url="jdbc:db2://localhost:50000/GBRX56FF;user=db2admin;
password=db2admin"
maxActive="8" maxIdle="4"/>
```

In the following context.xml Resource element for SQL Server or Sybase, Rhythmyx is publishing to multiple instances of the database:

- The @dbname attribute is required and holds the target database name. The @resourceName attribute holds the resource name from server.xml. The @origin attribute remains blank.

The following is the Resource element in the sample context.xml for SQL Server. The Resource name is “jdbc/test1”.

```
<Resource name="jdbc/test1" auth="Container"
type="javax.sql.DataSource" username="sa" password="demo"
driverClassName="com.inet.tds.TdsDriver" driver="TdsDriver"
url="jdbc:inetdae7:parmenion.percussion.com?database=test1"/>
```

The following is the Resource element in the sample context.xml for Sybase. The Resource name is "jdbc/dbpubtarget".

```
<Resource name="jdbc/dbpubtarget" auth="Container"
type="javax.sql.DataSource"
driverClassName="com.sybase.jdbc2.jdbc.SybDriver"
driver="sybase" url="jdbc:sybase:Tds:162.138.53.222:5010"/>
```

- psx-table - You must include a psx-table element to define the table in which the data is published. The @name attribute is the table name or the table alias.
- row sub-element - The row sub-element defines the table columns and child tables.
- 1 to n columnName elements - The names of the columnName elements must be the actual table column names or aliases. The @isEmptyNull attribute indicates whether nulls are allowed. Set it to yes (default) or no. The @encoding attribute specifies the type of encoding. Its possible values are:
  - `text (default)`
  - `escaped`
  - `base64`

  Leave the value as `text` if the content you are publishing is raw data (metadata or any other information that is left as an XML fragment). Set the value to `base64` or `escaped` if the content you are publishing is a formatted variant (this prevents the Publisher from confusing format characters). For more information on deciding the type of encoding, ***see Encoding*** Data (on page 16).
- 0 to n child elements - Each child element includes 0 to n child table Name elements. The names of the child table Name elements do not have to be the actual child table names, but they must be unique within the table definition. The @lookup attribute specifies the location of the child table.

NOTE: For any aliases that you provide, you must include the real name in an aliasmap in the sys_DatabasePublisher exit. You need to provide aliases only when the column names are not legal XML names. For more information, see the topic "sys_DatabasePublisher" in the *Rhythmyx CMS Online Help*.

Child Table HTML Markup

You must create an HTML file for each child table (sub-table) to be published along with the parent table (main table) in your target database. Drag and drop the HTML files onto the Rhythmyx Workbench to create query resources.

The markup rules for child tables are the same as those for parent tables; however the table definition psx-tabledefset and the database definition psx-database are not required. If these elements are included, the assembler ignores them.

The following is a sample child table HTML markup.

```
<html>
   <head>
      <title>CONTENT_CONTACT: child table markup</title>
   </head>
   <body>
      <!-- The table definition. -->
      <span id="psx-table/@name" />
      <span id="psx-table/row/CONTACTID" />
      <span id="psx-table/row/NAME" />
      <span id="psx-table/row/PHONE" />
      <span id="psx-table/row/EMAIL" />
   </body>
</html>
```

sys_DatabasePublisher

Context:
Java/global/percussion/contentassembler/

Description:
This exit is required on each database publisher parent table resource. This exit looks up the table definition specified in the parent table mapper and produces the XML file that conforms to the sys_DatabasePublisher.dtd.

Class name:
com.percussion.cas.PSDatabasePublisher

Interface:
com.percussion.extension.IPSResultDocumentProcessor

Parameters:

Name	**Data Type**	**Description**
action	java.lang.String	Action performed on the database: r - (default) Inserts the row. Deletes it first if it already exists. n - Inserts the row if it does not already exist. u - Updates the row if it already exists. d - Deletes the row if it already exists. Use d for unpublishing.
aliasmap	java.lang.String	Optional. Static XML file in the assembler application that contains table and column name mappings to be used if the table or column names contain characters that are not allowed in XML elements. The exit creates XML files that use the aliases, and then reinserts the real names in the output. Must conform to: ../dtd/aliasmap.dtd

Specifying the Repetition of XML Columns

In child tables that return multiple rows, columns may repeat within rows instead of appearing in separate rows. To prevent this, in these child tables, change the definition of each column in the Page Datatank.

NOTE: This step is not necessary if the XML encloses column names within <rowid> elements.

To change the definition of each column:

1. Open the Page Datatank for the XML resource.
2. Right-click on each column name and select *1 Element can appear exactly once* in the drop list.
3. Click [**OK**].

Encoding Data

The Database Publisher requires formatted content to be encoded after it retrieves it from the source database so that it does not incorrectly interpret formatting characters. It decodes the content before inserting it into the target database. The @encoding attribute in the columnName element specifies how the Database Publisher should encode each column of data from the source table.

The possible encoding types for Database Publishing are `text` (default), `escaped` (this represents "no-escaping" encoding), and `base64`.

Use `text` for unformatted text, and `escaped` or `base64` for formatted variants. The encoding attribute tells the Database Publisher how it must decode the received content before storing it into the target database. `escaped` formatting is easier to implement and allows you to view the variant during processing to determine if it is correct. `base64` requires more processing steps and does not allow you to view the variant during processing. However, in cases where the formatting fields and characters on the document would require very detailed xsl, it is simpler for you to use the base64 encoding UDF. When you specify `base64` encoding, you must map the target database body field to the output of a base64 encoding UDF. Rhythmyx provides several UDFs for this purpose:

- sys_Base64Encoder - Use this UDF if you are copying raw (unformatted) body data from the source database into the target database. Also use this to copy binary data, such as an image or PDF.
- sys_GetBase64EncodedBody - Use this UDF if you are copying the information between the <BODY> tags of a formatted Variant from the source database to the target database.
- sys_GetBase64Encoded - Use this UDF if you are copying an entire Variant (including <HTML>, <HEAD> and <BODY> tags) from the source database to a body field in the target database.

For more information, see the topics "sys_Base64Encoder," "sys_GetBase64EncodedBody," and "sys_GetBase64Encoded" in the *Rhythmyx CMS Online Help*.

Use the following guidelines to determine which type of encoding to use:

- use `text` for non-formatted text, such as metadata;
- use `escaped` for a formatted variant that comes from a single column in the source table and includes text with some inline markup;
- use `base64` for:
  - Variants assembled from several columns in the source table;
  - Variants that include binary files (including PDFs);
  - Variants that may or may not be formatted (that is, if you are uncertain whether to use `escaped` or `base64`, use `base64`).

Table Definition Builder

The Database Publisher refers to a table definition file, which supplies the schema of the target database. You can create a Table Definition file using the Table Definition Builder supplied by Rhythmyx. Open the Table Definition Builder dialog by running `<Rhythmyxroot>/AppServer/bin/runtd.bat` (for Windows) or `<Rhythmyxroot>/AppServer/bin/runTd.sh` (for Unix). The runtd script is installed with the Database Publisher.

If you are using the Microsoft SQLServer 2000 JDBC driver or the Sprinta JDBC driver:

The runtd.bat makes references to the following CLASSPATH entries, for use of the Microsoft SQLServer 2000 JDBC driver:

- `../common/lib/msbase.jar`
- `../common/lib/mssqlserver.jar`
- `../common/lib/msutil.jar`

Since this JDBC driver is not redistributable, you must download it from:

`http://download.microsoft.com/download/SQLSVR2000/Install/2.2.0022/NT5XP/EN-US/setup.exe` for Windows

or:

`http://download.microsoft.com/download/SQLSVR2000/Install/2.2.0022/UNIX/EN-US/mssqlserver.tar` for Unix

and then manually copy the 3 JAR files to the `<Rhythmyxroot>/AppServer/common/lib`.

Similarly, to use the Sprinta JDBC driver, you must copy the sprinta2000.jar file from the `/<Rhythmyxroot>/jdbc/sprinta` folder to the `/<Rhythmyxroot>/AppServer/common/lib` folder (this requires a license code). In addition, add the following CLASSPATH entry: ../common/lib/Sprinta2000.jar
to the `runTd.bat` file immediately in front of the class definition entry: com.percussion.tablefactory.tools.PSTDToolDialog.

You must insert a space between the CLASSPATH entry and the class definition. For example:

```
%EXECJAVA% -classpath ../webapps/RxServices/WEB-
INF/lib/rxtablefactory.jar;../webapps/RxServices/WEB-
INF/lib/rxclient.jar;../common/lib/xmlParserAPIs.jar;../webapps/RxServi
ces/WEB-INF/lib/xercesImpl.jar;../common/endorsed/xmlParserAPIs.jar;
../common/endorsed/xercesImpl.jar;../common/lib/jtds.jar;../common/lib/
classes12.jar;../common/lib/sybase.jar;../common/lib/msbase.jar;../comm
on/lib/mssqlserver.jar;../common/lib/msutil.jar;../webapps/RxServices/W
EB-INF/lib/saxon.jar;../common/lib/Sprinta2000.jar
com.percussion.tablefactory.tools.PSTDToolDialog
```

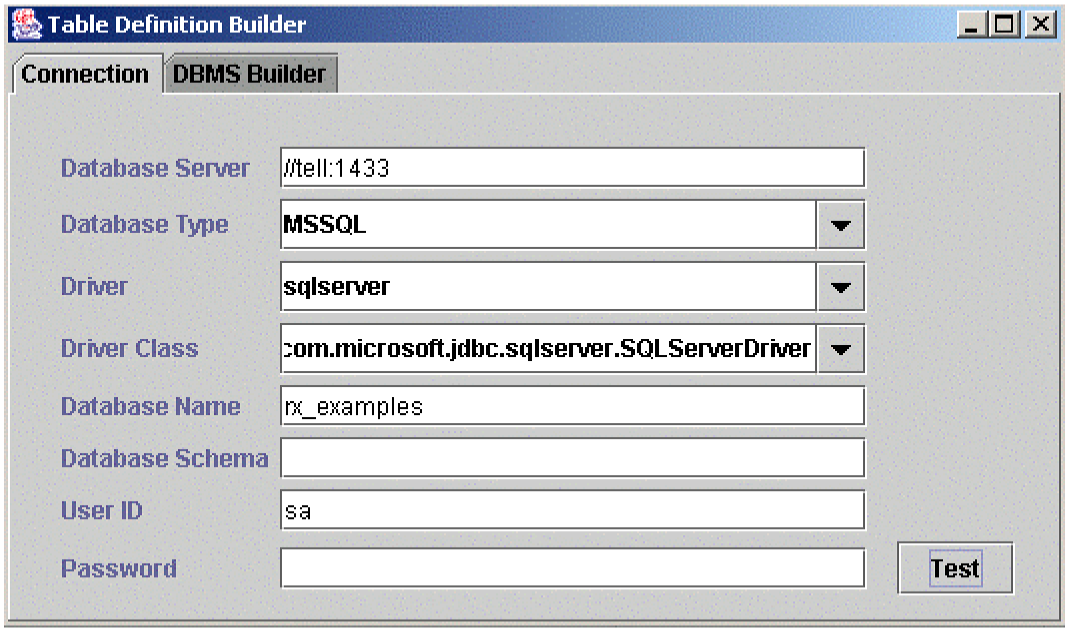
NOTE for Sprinta users: The Sprinta license is only available and licensed for Rhythmyx Servers licensed for SQL Server access. If you want to run the Database Publisher on another machine, either use the Microsoft Driver or obtain another Sprinta license.

Defining Target Database Connectivity Properties

The Table Definition Builder must connect to the target database before you can define the target schema. To connect to the target database, the Table Definition Builder uses connectivity properties that you provide.

To define target database connectivity properties:

1 Open the Table Definition Builder. Click the Connection tab if it is not selected.



Database Server	//tell:1433
Database Type	MSSQL
Driver	sqlserver
Driver Class	:om.microsoft.jdbc.sqlserver.SQLServerDriver
Database Name	rx_examples
Database Schema	
User ID	sa
Password	

Figure 1: Table Definition Builder, Connection Tab

2 In **Database Server**, enter the name or URL of the target database server. Required.

3 In **Database Type** enter a database type or choose one from the drop list. The drop list includes all currently supported database types.

4 In **Driver**, enter a driver or choose one from the drop list. The drop list includes all currently supported drivers. Required.

5 In **Driver Class**, enter a driver class or choose one from the drop list. The drop list includes all currently supported driver classes. Required.

6 In **Database Name**, enter the name of the target database.

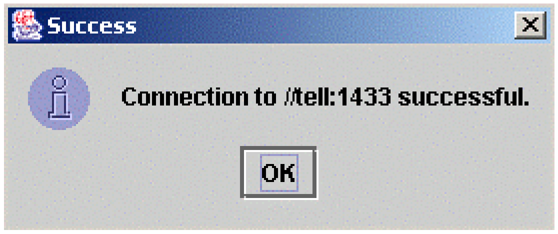
7 If the database type supports schemas, in **Database Schema**, enter the target database schema.

8 In **User Id**, enter the name of the user who has access to the target database.

9 In **Password**, enter the password associated with the **User Id**.

10 To test the connection information, click [**Test**].

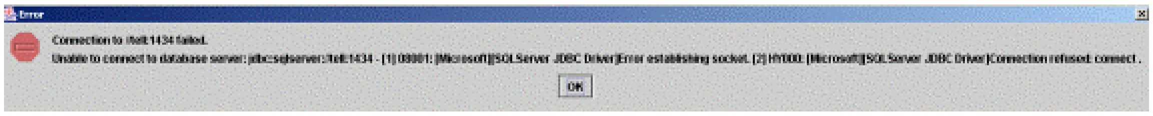
- If the connection works, the tool displays the dialog:



Connection to //tell:1433 successful.

Figure 2: Successful Database Connection

- If the connection fails, the tool displays an Error dialog. (You cannot save a connection that fails.):



```
Connection to //tell:1434 failed.
Unable to connect to database server: jdbc:sqlserver://tell:1434 - [1] 08001: [Microsoft][SQLServer JDBC Driver]Error establishing socket. [2] HY000: [Microsoft][SQLServer JDBC Driver]Connection refused: connect .
```

Figure 3: Database Connection Failure

11 After you confirm that the connection works, click the DBMS Builder tab to select target database tables and create the target table definition file.

The following tables provide examples of the information required to connect to different database types.

for Tomcat 4.1.27 and later

Database Server	DB Type	Driver	Driver Class	Database Name	Database Schema	User ID	Password
//localhost:1433	jTDS	jtds:sqlserver	net.sourceforge.jtds.jdbc.Driver	dbpubtarget	N/A	sa	demo
//localhost:1433	MSSQL	sqlserver	com.microsoft.jdbc.sqlserver.SQLServerDriver	dbpubtarget	N/A	sa	demo
@localhost:1521:UTF8	Oracle:thin	oracle:thin	oracle.jdbc.driver.OracleDriver	dbpubtarget	RX_EXAMPLES	sa	demo
Tds:localhost:5000	SYBASE	sybase	com.sybase.jdbc2.jdbc.SybDriver	dbpubtarget	N/A	sa	demo
SAMPLE (database name as specified in the Client Configuration Assistant)	DB2	db2	COM.ibm.db2.jdbc.app.DB2Driver	N/A	N/A	sa	demo

for versions of Tomcat prior to 4.1.27:

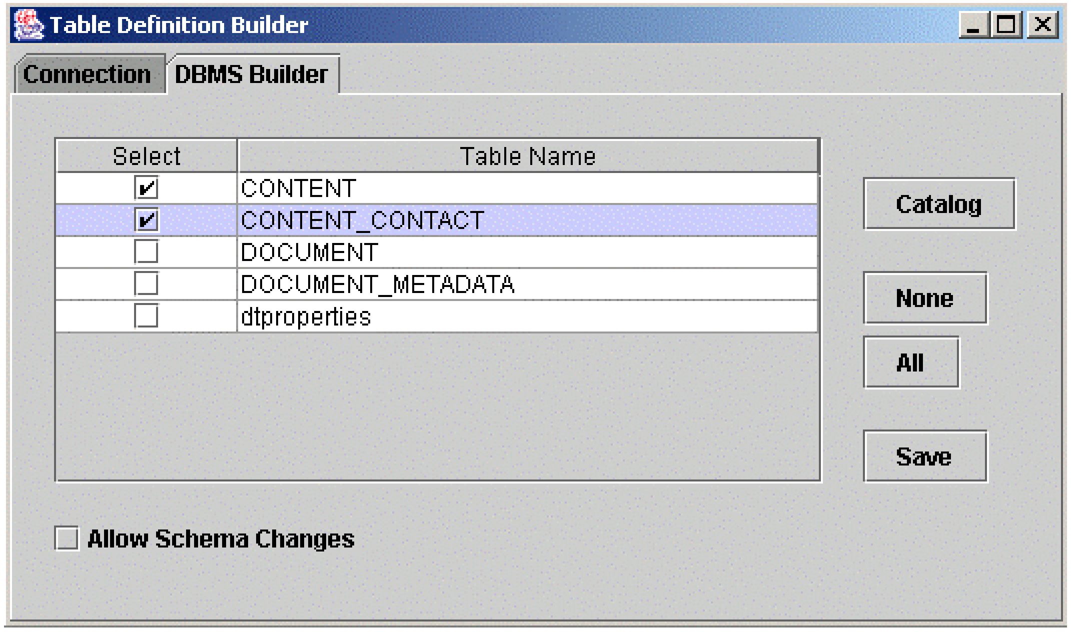
Database Server	DB Type	Driver	Driver Class	Database Name	Database Schema	User ID	Password
//localhost:1433	MSSQL	sqlserver	com.microsoft.jdbc.sqlserver. SQLServerDriver	rx_examples	N/A	sa	<blank>
@localhost:1521: orcl	ORACLE	Oracle:thin	oracle.jdbc.driver.OracleDriver	N/A	RX_EXAMPLES	scott	tiger
Tds:dbserver:5000	SYBASE	sybase	com.sybase.jdbc2.jdbc.SybDriver	rx_examples	N/A	sa	<blank>
SAMPLE (database name as specified in the Client Configuration Assistant)	DB2	db2	COM.ibm.db2.jdbc.app. DB2Driver	N/A	N/A	sample	user

Selecting Tables and Creating the Table Definition

You select target database tables from the tables cataloged using the connection specified in the Connection tab. Therefore, you can only select database tables if your target database connection is valid.

To select target database tables:

1. ***Define target database connectivity properties*** (see "Defining Target Database Connectivity Properties" on page 18). Click the DBMS Builder tab.
2. To list all tables cataloged with the database connection specified in the Connection tab, click [**Catalog**]. If your connection is not valid, the tool displays an error. If your connection is valid the dialog lists all tables cataloged with the database connection.



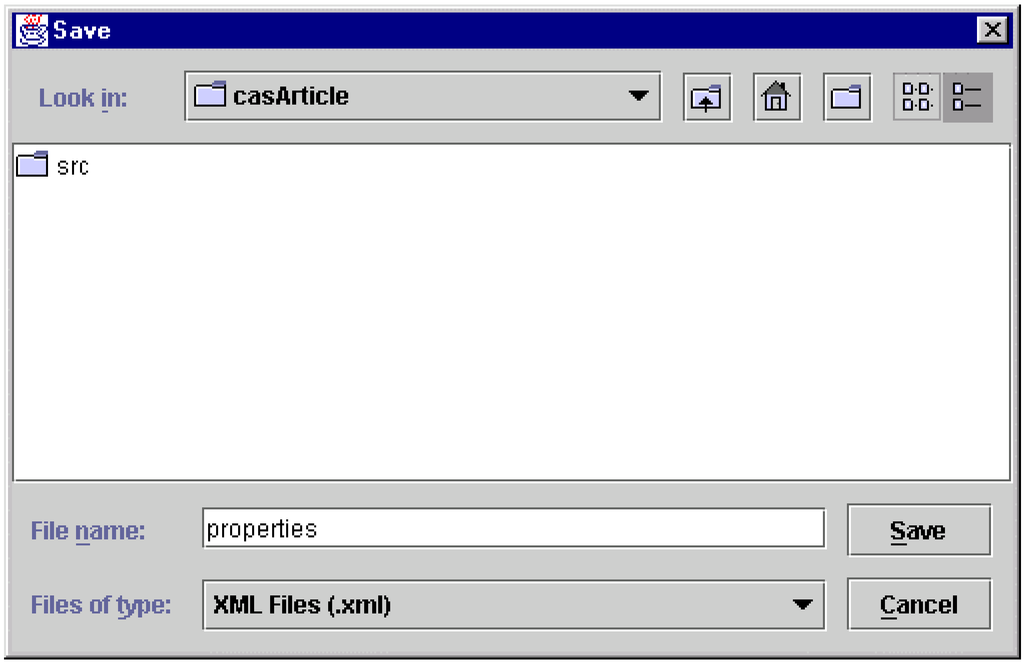
Select	Table Name
✓	CONTENT
✓	CONTENT_CONTACT
☐	DOCUMENT
☐	DOCUMENT_METADATA
☐	dtproperties

☐ Allow Schema Changes

Figure 4: Table Definition Builder, DBMS Builder Tab

3. Click the checkboxes beside the table names to include the tables in your output schema. Click [**All**] to check all of the tables. Click [**None**] to uncheck all of the tables. To select specific tables, check them individually.
4. To save your output table schema to a table definition file, click [**Save**]. If you have not selected any tables, the tool returns an error dialog.

If you have selected tables, the tool displays a Save dialog.



File name: properties

Files of type: XML Files (.xml)

Figure 5: Save Dialog

5 Navigate to any directory and enter a File name. If you have not yet created your assembler application, move the table definition file into its directory after you create it.

6 Leave **Files of type** as XML Files. The tool always saves to an .xml file that conforms to sys_Tabledef.dtd in <Rhythmyx>/DTD/sys_DatabasePublisher.dtd.

7 Click [**Save**]. If your connection is valid, the tool saves your file.

Modifying the Table Definition File for Sequential Columns

If the table(s) defined in your table definition file has sequential columns, you must add a `<sequence>` tag inside the `<columndef>` tag to avoid a key violation error.

In Oracle, the value in the `<sequence>` tag must be the name of the sequence used to obtain the value of the column. In other databases (SQL Server, Solaris, DB2) the value in the `<sequence>` tag can be any string.

If the row action is replace (`"r"`), the database server performs an update if the row exists, and an insert if it does not exist. An update requires a valid key value. The server cannot obtain the value of sequential columns; if your primary key contains sequential columns, you must add an update key (if an update key does not already exist). The update key should not contain any sequential columns.

NOTE: The `sys_Tabledef.dtd` specifies that keys and index definitions must be defined in your table definition file in the order: Primary Key, Foreign Key, Update Key, Index Definitions.

Example:

If a table has the primary key column `ID`, the Table Definition Builder creates the following column definition for `ID` in the table definition file:

```
<columndef action="c" name="ID">
   <jdbctype>INTEGER</jdbctype>
   <allowsnull>no</allowsnull>
</columndef>
```

If `ID` is a sequential column, you must manually edit its column definition in the table definition file and add a `<sequence>` tag:

```
<columndef action="c" name="ID">
   <jdbctype>INTEGER</jdbctype>
   <allowsnull>no</allowsnull>
   <sequence>ID</sequence>
</columndef>
```

If the table already exists and you want to change the action to replace `"r"`, you must have an update key. If the table does not already have an update key and you cannot use an existing field as an update key, you must add one to the table:

```
<primarykey action="r">
   <name>ID</name>
</primarykey>
<updatekey action="r">
   <name>CONTENTID</name>
</updatekey>
```

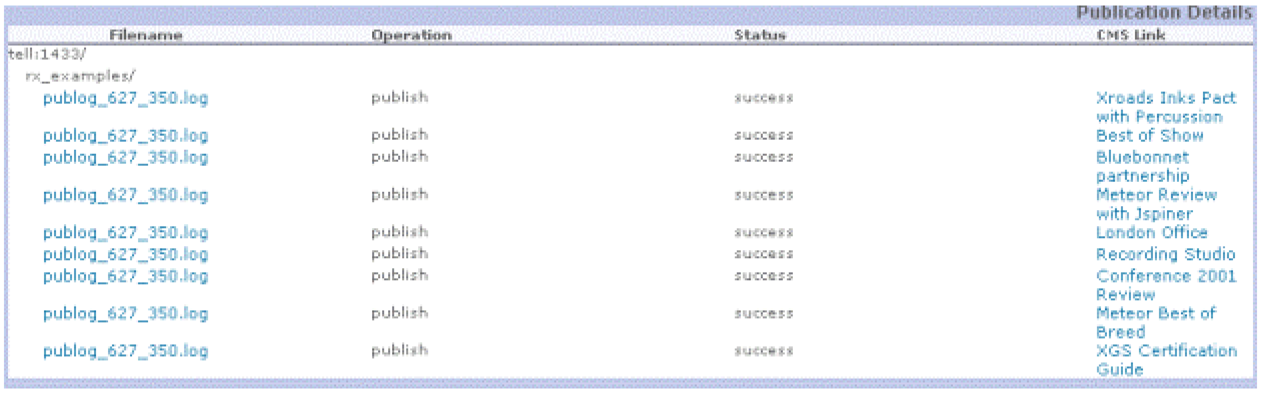
See ***tabledefset.xml example*** (see "Creating the News Database Table Definition" on page 52) for an example of a complete table definition file created by the Table Definition Builder.

Displaying a Database Logging View

After filesystem Publishing in Rhythmyx, the Publication Detail Map displays links to each page on your Web site and links to the corresponding content items as assembled by Rhythmyx. Since the Database Publisher does not create a separate file for each published item, the links in the Filename column in the Publication Details Map link to the publication log, which contains all item data that was published to the target database for one content list. The `CMS Link` column in the Publication Detail Map displays links to assembled items as it does for filesystem Publishing. Your entry in the **Publishing Root Location** field in the Edit Site Properties page specifies what the Publication Detail Map displays as a header above the log file names.

To specify the location that the Publication Details Map displays above log file names:

1 Set the **Publishing Root Location** in the Edit Site Properties page to `<server>:<port>/<database>/` or another appropriate description for the location of the publishing tree. You must end the **Publishing Root Location** with a forward slash. In the example below, the **Publishing Root Location** is `tell:1433/rx_examples/`.



Publication Details

Filename	Operation	Status	CMS Link
tell:1433/			
rx_examples/			
publog_627_350.log	publish	success	Xroads Inks Pact with Percussion
publog_627_350.log	publish	success	Best of Show
publog_627_350.log	publish	success	Bluebonnet partnership
publog_627_350.log	publish	success	Meteor Review with Jspiner
publog_627_350.log	publish	success	London Office
publog_627_350.log	publish	success	Recording Studio
publog_627_350.log	publish	success	Conference 2001 Review
publog_627_350.log	publish	success	Meteor Best of Breed
publog_627_350.log	publish	success	XGS Certification Guide

Figure 6: Publication Details Map

NOTE: There is one log file per content list, so the same log file name may display repeatedly in the `Filename` column when a single content list includes several items in the `CMS Link` column.

Database Publisher DTDs

The sys_DatabasePublisher.dtd and the aliasmap.dtd for the Database Publisher are installed to <Rhythmyx>/DTD if your installation code includes Database Publishing.

- sys_DatabasePublisher.dtd - specifies the format for content items delivered to the Database Publisher plugin; its main elements are:
  - <tabledefset> - defines the table schema used for publishing items; its structure is specified in sys_Tabledef.dtd, which is included in sys_DatabasePublisher.dtd;
  - <tabledataset> - contains the table data that is published; its structure is specified in sys_Tabledata.dtd, which is included in sys_DatabasePublisher.dtd;
- aliasmap.dtd - specifies the format for aliasmaps provided to the sys_DatabasePublisher exit.

Setting Up Tomcat Data Sources

Each content item delivered from the Publisher for database publishing is an XML file conforming to the ***sys_DatabasePublisher.dtd*** (see "Database Publisher DTDs" on page 26). This XML file contains content as well as information for the publishing connection and target database. The Database Publisher looks up connections through JNDI. Therefore you must register all Data Sources in the server you are using.

To set up Tomcat Data Sources:

1. Shut down Tomcat (if it is running).
2. Open `<Rhythmyxroot>\AppServer\webapps\RxServices\META-INF\context.xml` in a simple text editor.
3. Configure the <Resource> element to point to your RDBMS.
   a) Uncomment the <Resource> element for your RDMS (Commented <Resource> elements are provided for both the JTDS drive for Microsoft SQL Server and for Oracle.)
   b) In the `url` attribute of the <Resource> element, modify the connect string to point to your RDBMS.
   c) Modify the username and password to the credentials required to access your RDBMS. If you use the JTDS driver, the username and password are part of the connect string in the `url` attribute. If you use any other RDBMS, the username and password are separate attributes.
   d) You can add other attributes to this element. For details about the attributes you can add, see `http://jakarta.apache.org/commons/dbcp/configuration.html`.
4. Save your changes to `context.xml`.
5. Restart your Tomcat server.

To test your configuration, open a browser and enter the following URL:

```
http://host:port/RxServices/jnditest.jsp
```

Where

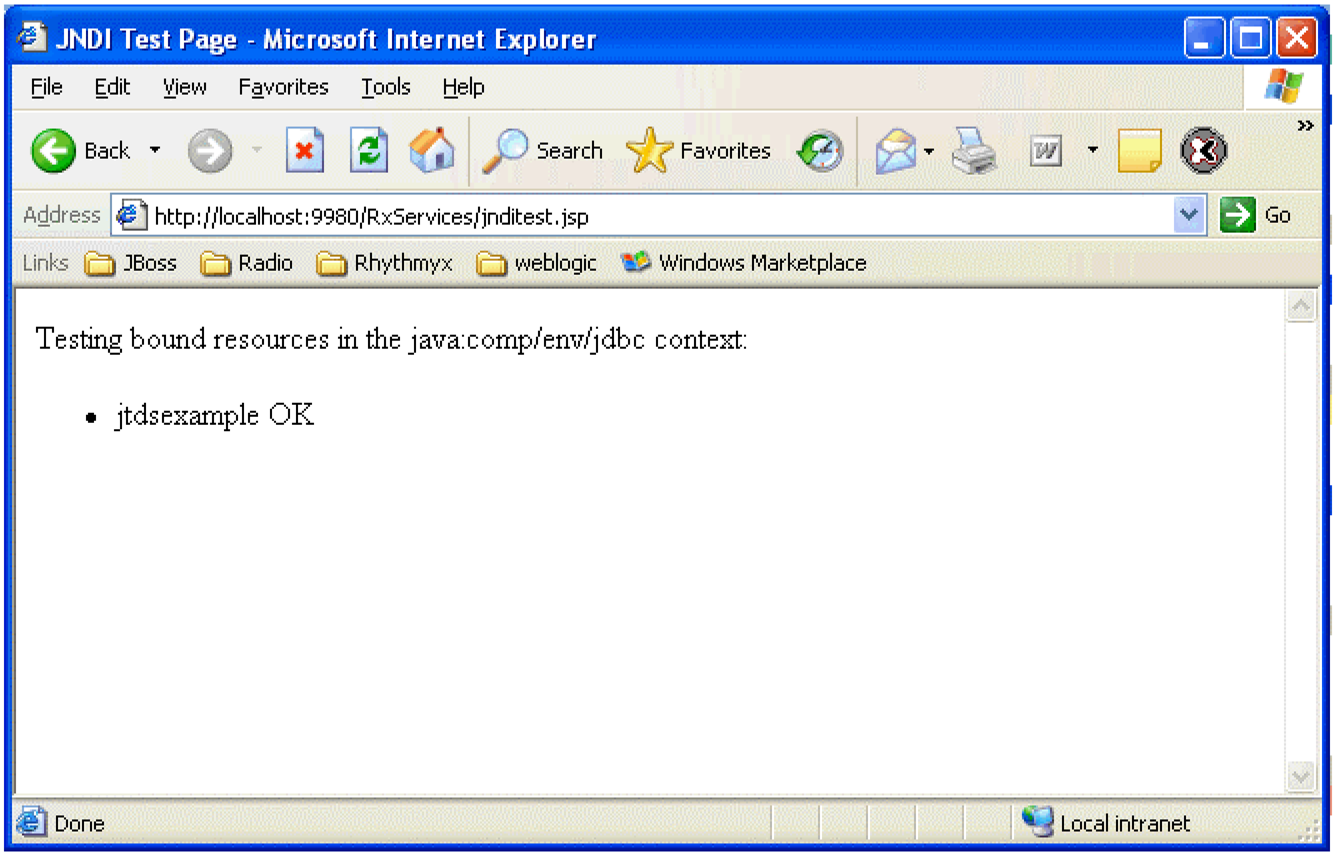
host is the name of the machine where your Rhythmyx server resides; and

port is the port on which your application server listens (9980 by default).

For example:

```
http://localhost:9980/RxServices.jnditest.jsp
```

If your data source is configured correctly, Tomcat will return a message similar to the following screenshot:



http://localhost:9980/RxServices/jnditest.jsp

Testing bound resources in the java:comp/env/jdbc context:

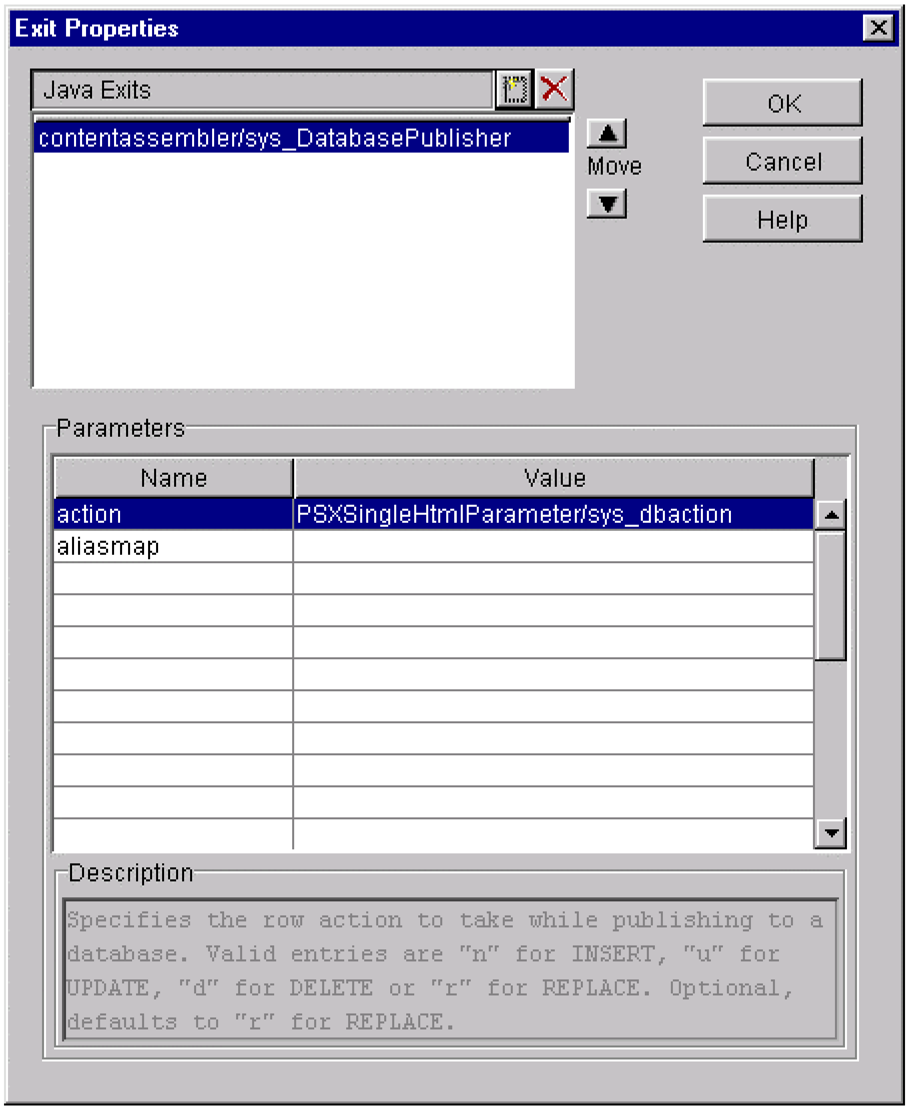
- jtdsexample OK

Unpublishing with the Database Publisher

To unpublish with the Database Publisher, the `sys_DatabasePublisher` exit on your assembler application must have an action parameter set to `d`. You can use the same assembler application for publishing and unpublishing if you insert an HTML parameter in the action parameter's **Value** field.

To set up a database content assembler to do both publishing and unpublishing:

1. Set the action in the `sys_DatabasePublisher` ***exit*** (see "sys_DatabasePublisher" on page 14) to an HTML parameter instead of `r`.



Name	Value
action	PSXSingleHtmlParameter/sys_dbaction
aliasmap	

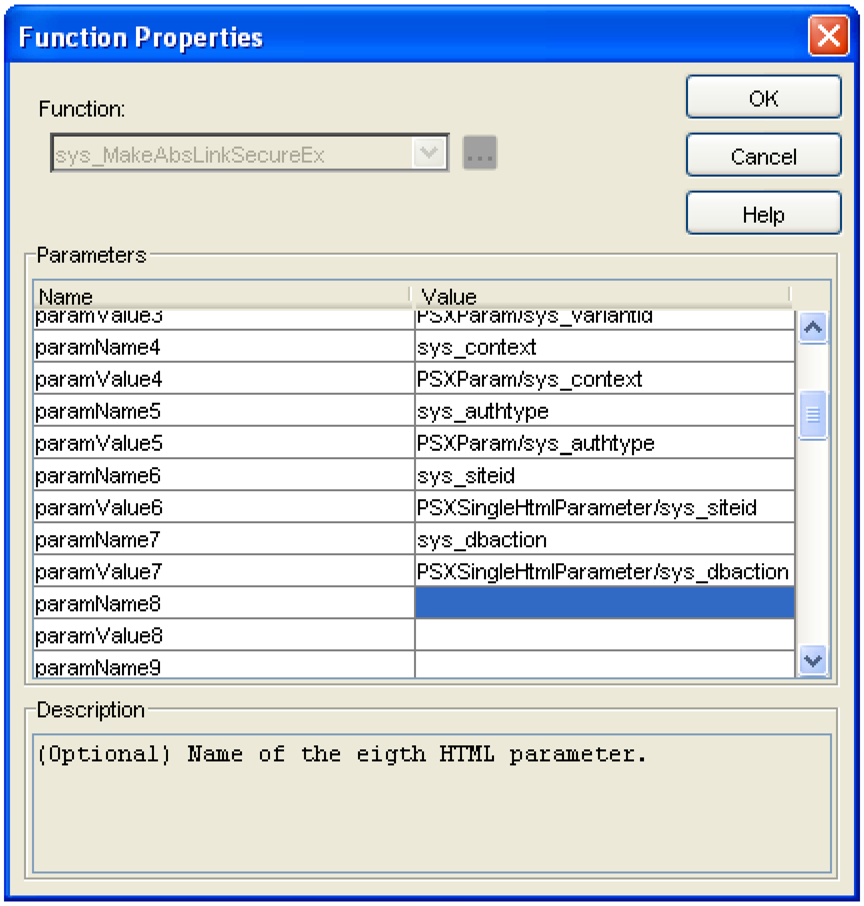
```
Specifies the row action to take while publishing to a
database. Valid entries are "n" for INSERT, "u" for
UPDATE, "d" for DELETE or "r" for REPLACE. Optional,
defaults to "r" for REPLACE.
```

Figure 7: Mapping the sys_DatabasePublisher exit

2. In the content list registration, add the HTML parameter name to the end of the URL. Set it equal to `d` to unpublish. (You may set it to `r` to publish.)

Example unpublishing content list URL:
`/Rhythmyx/xrd_casGeneric/contentlist_database.xml?sys_variantid=329&sys_dbaction=d`

3. When you set the sys_DatabasePublisher action parameter to `sys_dbaction` instead of a value, you must explicitly pass the parameter to your content list application in the sys_MakeAbsLinkSecure or sys_MakeAbsLinkSecureEx exit.



Function: sys_MakeAbsLinkSecureEx

Name	Value
paramValue3	PSXParam/sys_variantid
paramName4	sys_context
paramValue4	PSXParam/sys_context
paramName5	sys_authtype
paramValue5	PSXParam/sys_authtype
paramName6	sys_siteid
paramValue6	PSXSingleHtmlParameter/sys_siteid
paramName7	sys_dbaction
paramValue7	PSXSingleHtmlParameter/sys_dbaction
paramName8	
paramValue8	
paramName9	

Description: (Optional) Name of the eigth HTML parameter.

Figure 8: sys_dbaction parameter added to sys_MakeAbsLinkSecureEx

If you do not pass sys_dbaction through sys_MakeAbsLinkSecure or sys_MakeAbsLinkSecureEx, sys_DatabasePublisher is never passed its value, and the value defaults to `r`.

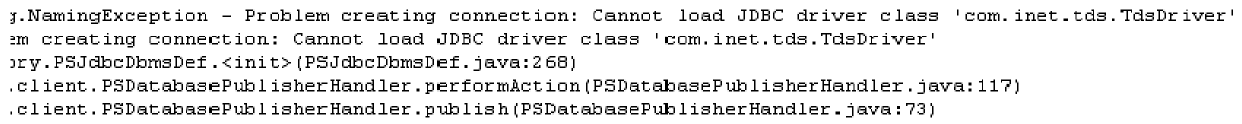
Database Publishing Error Messages

1 NameNotFoundException

```
Unexpected error:
Unexpected Exception in javax.naming.NameNotFoundException - Name
jdbc is not bound in this Context
javax.naming.NameNotFoundException: Name jdbc is not bound in
this Context at
org.apache.naming.NamingContext.lookup(NamingContext.java:811)
```

Meaning: DB Name in server.xml for resource does not match name in mapper for resourceName.

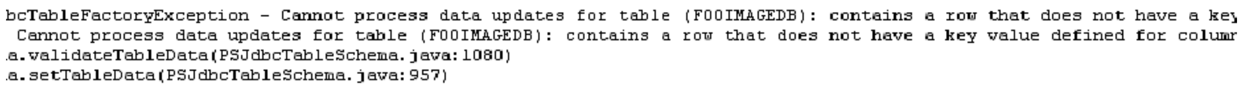
2 NamingException



```
g.NamingException - Problem creating connection: Cannot load JDBC driver class 'com.inet.tds.TdsDriver'
em creating connection: Cannot load JDBC driver class 'com.inet.tds.TdsDriver'
ory.PSJdbcDbmsDef.<init>(PSJdbcDbmsDef.java:268)
.client.PSDatabasePublisherHandler.performAction(PSDatabasePublisherHandler.java:117)
.client.PSDatabasePublisherHandler.publish(PSDatabasePublisherHandler.java:73)
```

Meaning: driverclassname or url incorrect in content list application.

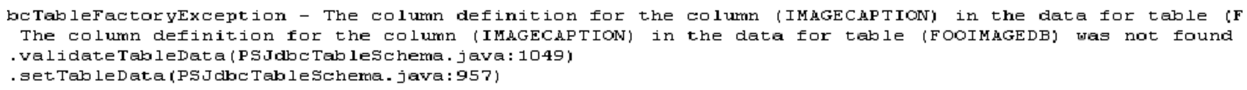
3 jdbcTableFactoryException - cannot process



```
bcTableFactoryException - Cannot process data updates for table (FOOIMAGEDB): contains a row that does not have a key
 Cannot process data updates for table (FOOIMAGEDB): contains a row that does not have a key value defined for column
a.validateTableData(PSJdbcTableSchema.java:1080)
a.setTableData(PSJdbcTableSchema.java:957)
```

Meaning: Target table does not contain a primary key or columns in tableDef.xml do not match column names mapped in assembly application (case sensitive).

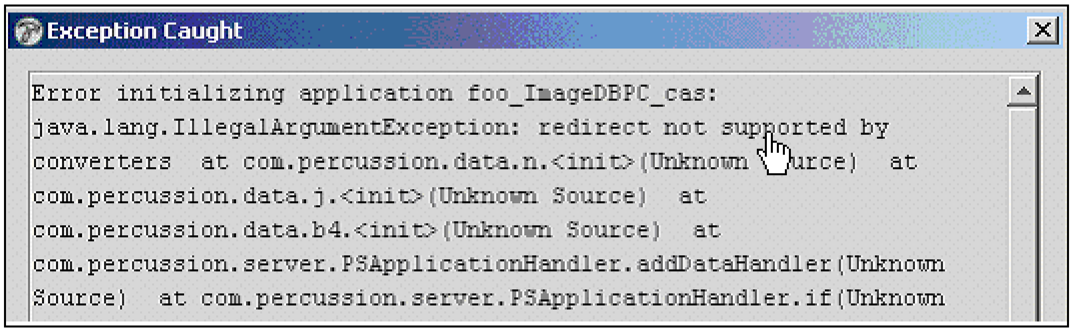
4 jdbcTableFactoryException - column definition



```
bcTableFactoryException - The column definition for the column (IMAGECAPTION) in the data for table (F
 The column definition for the column (IMAGECAPTION) in the data for table (FOOIMAGEDB) was not found
.validateTableData(PSJdbcTableSchema.java:1049)
.setTableData(PSJdbcTableSchema.java:957)
```

Meaning: Column names defined in tableDef.xml do not match the table names in the assembly application mapping.

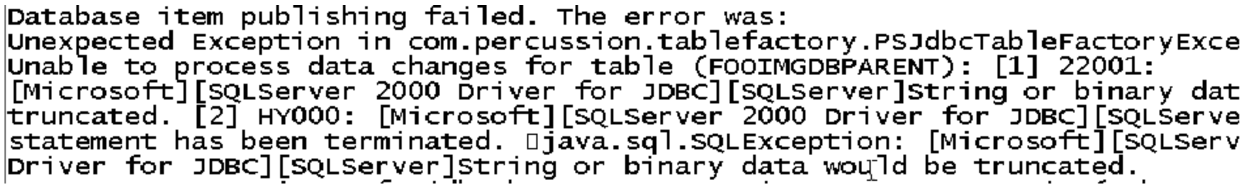
5 IllegalArgumentException - redirect not supported



```
Error initializing application foo_ImageDBPC_cas:
java.lang.IllegalArgumentException: redirect not supported by
converters  at com.percussion.data.n.<init>(Unknown Source)  at
com.percussion.data.j.<init>(Unknown Source)  at
com.percussion.data.b4.<init>(Unknown Source)  at
com.percussion.server.PSApplicationHandler.addDataHandler(Unknown
Source)  at com.percussion.server.PSApplicationHandler.if(Unknown
```

Meaning: Link between Query Resource and page (XSL) broken.

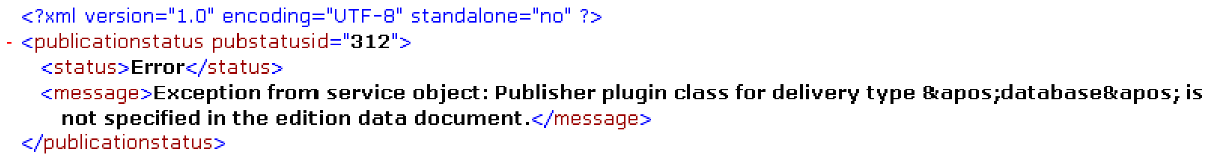
6 PSJdbcTableFactoryException



```
Database item publishing failed. The error was:
Unexpected Exception in com.percussion.tablefactory.PSJdbcTableFactoryExce
Unable to process data changes for table (FOOIMGDBPARENT): [1] 22001:
[Microsoft][SQLServer 2000 Driver for JDBC][SQLServer]String or binary dat
truncated. [2] HY000: [Microsoft][SQLServer 2000 Driver for JDBC][SQLServe
statement has been terminated. □java.sql.SQLException: [Microsoft][SQLServ
Driver for JDBC][SQLServer]String or binary data would be truncated.
```

Meaning: Column length not big enough (possibly of wrong type) to accommodate length of data being inserted or "Elements can appear exactly once" not set in DTD.

7 Exception from service object: Publisher plugin



```
  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <publicationstatus pubstatusid="312">
    <status>Error</status>
    <message>Exception from service object: Publisher plugin class for delivery type &apos;database&apos; is
      not specified in the edition data document.</message>
  </publicationstatus>
```

Meaning: When registering Publisher the parameter database and its value com.percussion.publisher.client.PSDatabasePublisherHandler missing or entered incorrectly.

8 Unexpected Exception in javax.naming.NamingException - Problem creating connection

```
Unexpected error:
Unexpected Exception in javax.naming.NamingException - Problem
creating connection: Cannot create JDBC driver of
javax.naming.NamingException: Problem creating connection: Cannot
create JDBC driver of class 'oracle.jdbc.driver.OracleDriver' for
connect URL 'null'
class 'oracle.jdbc.driver.OracleDriver' for connect URL 'null'
```

Meaning: Username (name) not defined properly in server.xml.

CHAPTER 4

Example: Publishing the News Content Type to a Database

The following example shows how to create the assembler for publishing a News content type to a target database. You can use this example as a model for database publishing using any server.

This example publishes content in several formats:

- content metadata as raw data;
- a content abstract as rich text data;
- body content as a PDF;
- body content as an HTML fragment.

For details about how to use the tools and applications in this example, see the previous chapter, ***Database Publishing Reference and Guide*** (on page 9).

News Repository Tables and Content Editor

The repository (source database) tables that store the News content in Rhythmyx are the user-created XRDNEWS content table, the user-created XRDCONTACT contact metadata table, and the Rhythmyx CONTENTSTATUS table. For detailed information about these tables, see ***XRDNEWS Table*** (on page 36), ***XRDCONTACT Table*** (on page 37), and the topic "CONTENTSTATUS" in the *Rhythmyx CMS Online Help*.

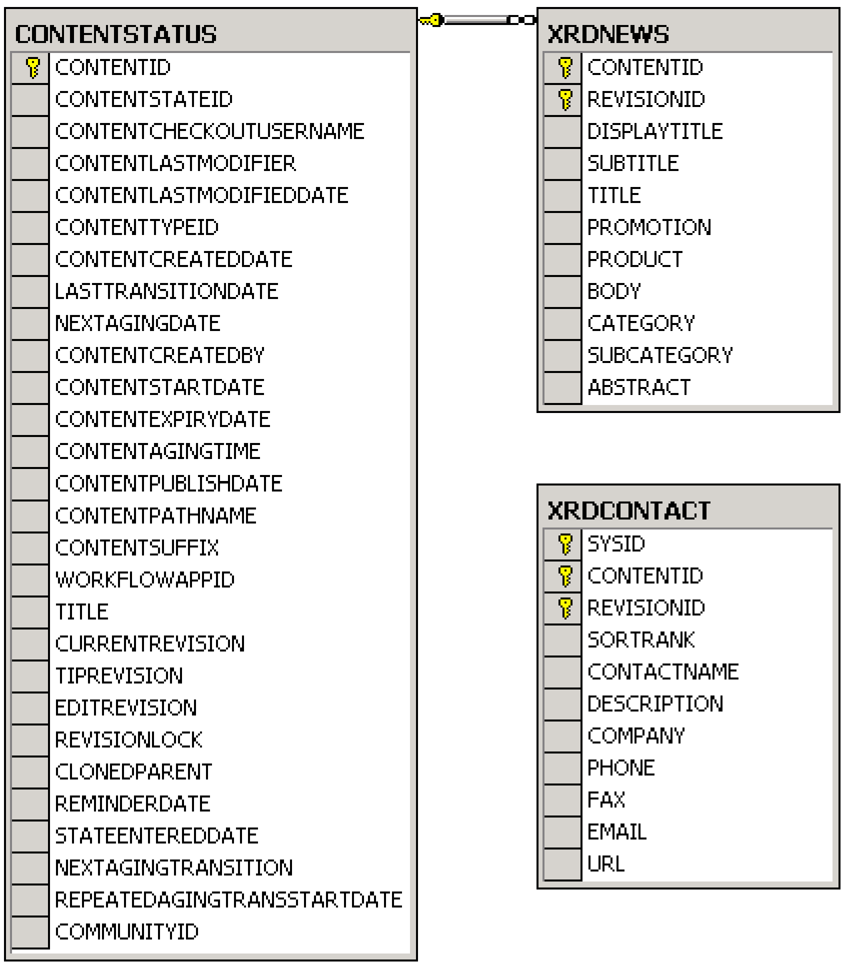


Figure 9: News and Content Repository Tables

The following graphic shows the News content editor:

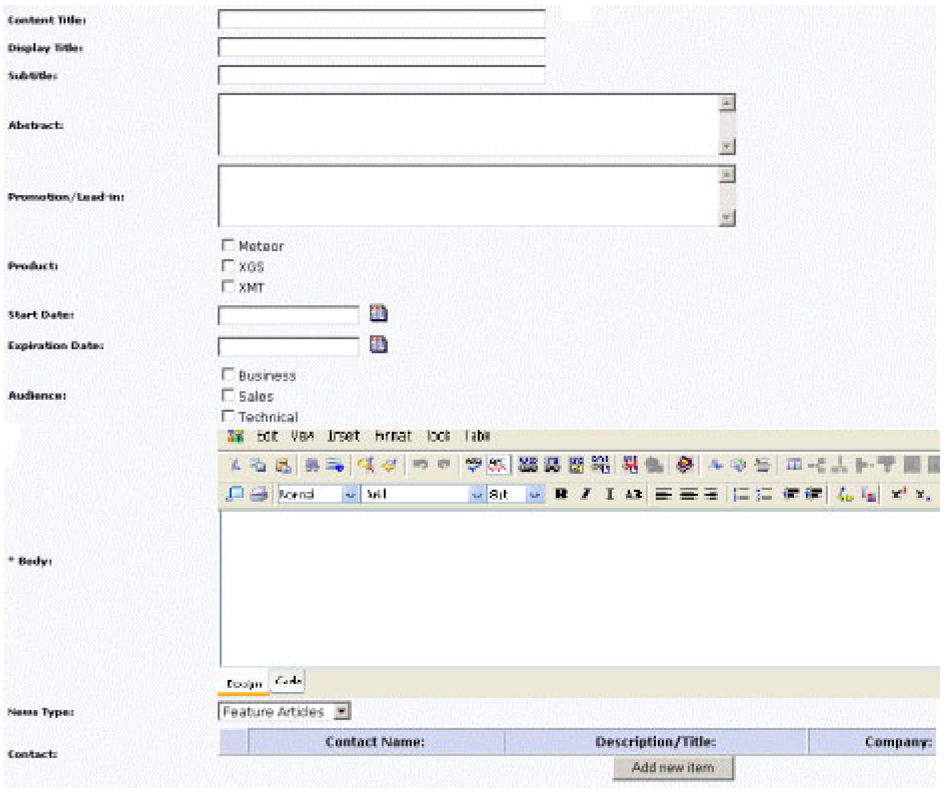


Figure 10: News Content Editor

XRDNEWS Table

This table stores content for the News content type.

Name	Type	Parameters	Allows null values
CONTENTID (primary key)	INT	4	no
REVISIONID (primary key)	INT	4	no
DISPLAYTITLE	NVARCHAR	100	yes
SUBTITLE	NVARCHAR	100	yes
TITLE	NVARCHAR	100	yes
PROMOTION	NVARCHAR	512	yes
PRODUCT	NVARCHAR	100	yes
BODY	TEXT	16	yes
CATEGORY	INT	4	yes
SUBCATEGORY	INT	4	yes
ABSTRACT	NVARCHAR	512	yes

XRDCONTACT Table

This table stores contact information.

Name	Type	Parameters	Allows null values
SYSID (primary key)	INT	4	no
CONTENTID (primary key)	INT	4	no
REVISIONID (primary key)	INT	4	no
SORTRANK	INT	4	no
CONTACTNAME	NVARCHAR	100	yes
DESCRIPTION	NVARCHAR	255	yes
COMPANY	NVARCHAR	30	yes
PHONE	NVARCHAR	30	yes
FAX	NVARCHAR	30	yes
EMAIL	NVARCHAR	100	yes
URL	NVARCHAR	255	yes

News Target Database

The target database will use two database tables: CONTENT and CONTENT_CONTACT. CONTENT is the parent table and CONTENT_CONTACT is its child table. CONTENT_CONTACT is associated with CONTENT by the key column ID. The CONTENT table holds the content body HTML and PDF variants and content metadata. The CONTENT_CONTACT table holds contact information related to the content.

If you publish to a parent/child table structure as in this example (the CONTENT table is the parent and the CONTENT_CONTACT table is the child table), you must specify a relationship. The example uses the ID column in both tables to specify the relationship. The relationship used is stored in the <foreignkey> element in the ***table definition file*** (see "Creating the News Database Table Definition" on page 52).

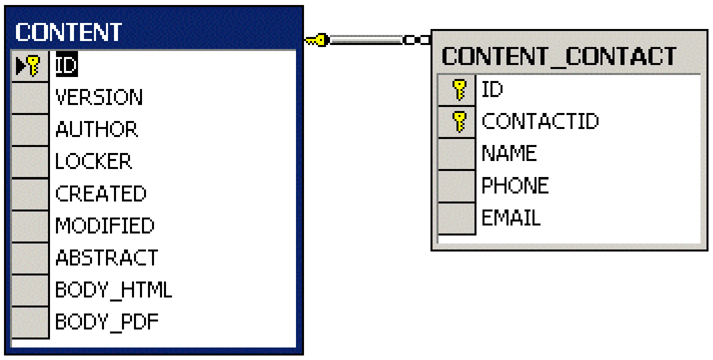


Figure 11: CONTENT and CONTENT_CONTACT Tables

This example will publish content, content metadata and related contact information to the target database.

The example assumes:

- the News content editor and source database are already built and available;
- the database server (for the Rhythmyx repository and target database) is MS SQL;
- the Rhythmyx Publisher was installed with its standard servlet runner Tomcat.

CONTENT Table

This table stores content body variants and metadata.

Name	Type	Parameters	Allows null values	Description
ID	INT	4	no	Primary key. Stores the CONTENTSTATUS.CONTENTIDcolumn as raw data.
VERSION	INT	4	yes	Stores the CONTENTSTATUS.CURRENTREVISION column as raw data.
AUTHOR	NVARCHAR	50	yes	Stores the CONTENTSTATUS.CONTENTCREATEDBY column as raw data.
LOCKER	NVARCHAR	50	yes	Stores the CONTENTSTATUS.CONTENTCHECKOUTUSERNAME column as raw data.
CREATED	DATETIME	8	no	Stores the CONTENTSTATUS.CONTENTCREATEDDATE column as raw data.
MODIFIED	DATETIME	8	no	Stores the CONTENTSTATUS.CONTENTLASTMODIFIEDDATE column as raw data.
ABSTRACT	NTEXT	16	no	Stores the XRDNEWS.ABSTRACT column as an HTML fragment.
BODY_HTML	NTEXT	16	no	Stores the XRDNEWS.BODY column as an HTML page.
BODY_PDF	BINARY	50	no	Stores the XRDNEWS.BODY column as a PDF.

CONTENT_CONTACT Table

This table stores contact information related to content in the CONTENT table. It is related to the CONTENT table by the key column ID.

Name	Type	Parameters	Allows null values	Description
ID	INT	4	no	Primary key field that associates CONTENT_CONTACT table to CONTENT table by matching CONTENT.ID field.
CONTACTID	INT	4	no	Primary key that stores XRDCONTACT.SYSID field as raw data
NAME	NVARCHAR	50	yes	Stores XRDCONTACT.CONTACTNAME field as raw data.
PHONE	NVARCHAR	50	yes	Stores XRDCONTACT.PHONE field as raw data.
EMAIL	NVARCHAR	50	yes	Stores XRDCONTACT.EMAIL field as raw data.

Procedure for Building the News Publisher

The following steps outline the procedure for building the News Publisher.

1 Build the assembler for publishing content, content metadata and related contacts to the target database

   a) ***Mark up HTML files for each target parent and child table*** (see "Marking Up HTML Files for the News Target Database" on page 42).

   b) Drag and drop the marked up HTML files onto the Workbench and attach the ***sys_DatabasePublisher exit*** (see "sys_DatabasePublisher" on page 14) to the parent table.

   c) For each resource, attach tables, define the Selector, and ***map the application resources*** (see "Mapping the News content and contentContact resources" on page 45).

   d) ***Specify the repetition of columns in the resources*** (see "Specifying Column Repetition in News Resources" on page 51).

   e) Attach sys_GetBase64EncodedBody and sys_Base64Encoder as UDFs to your content resource.

   f) ***Create the table definition file*** (see "Creating the News Database Table Definition" on page 52).and copy it into the xrd_casGeneric directory as a static resource to the database assembler application

   g) Register the Database Assembly Variant.

2 Build the assemblers for creating formatted Variants.

   a) Build resources, Slots and Variants.

   b) ***Build Variant assembler applications*** (see "Variant Assembler Applications" on page 58).

3 ***Build the content list application to publish items to the database*** (see "Building the News Content List Application" on page 60).

4 Register the Publisher, Site, Edition and Content Lists in the Content Explorer.

5 ***Set Up Tomcat Data Sources*** (see "Setting Up Tomcat Data Sources" on page 27).

Marking Up HTML Files for the News Target Database

Create HTML markup files for the CONTENT and CONTENT_CONTACT tables to use as source files in the assembler application.

Sample HTML markup file for CONTENT (the parent table):

```
<html>
   <head>
      <title>CONTENT: parent table markup</title>
   </head>
   <body>
      <!-- The complete table definition, required in parent tables. -->
      <span id="psx-tabledefset/@lookup" />
      <!-- The database definitions, required in parent tables. -->
      <span id="psx-database/@dbname" />
      <span id="psx-database/@drivertype" />
       <span id="psx-database/@resourceName" />
       <span id="psx-database/@origin" />
      <!-- The table definition. -->
      <span id="psx-table/@name" />
      <span id="psx-table/row/ID" />
      <span id="psx-table/row/VERSION" />
      <span id="psx-table/row/AUTHOR" />
      <span id="psx-table/row/LOCKER" />
      <span id="psx-table/row/CREATED" />
      <span id="psx-table/row/MODIFIED" />
<span id="psx-table/row/ABSTRACT" />
      <span id="psx-table/row/ABSTRACT/@encoding" />
      <span id="psx-table/row/BODY_HTML" />
      <span id="psx-table/row/BODY_HTML/@encoding" />
<span id="psx-table/row/BODY_PDF" />
      <span id="psx-table/row/BODY_PDF/@encoding" />
      <!-- Include all child tables, optional -->
      <span id="psx-table/row/children/contentContact/@lookup" />
   </body>
</html>
```

The XML for the CONTENT table includes the required tabledefset and database elements that store the table definition and the database name, driver type, resource name, and origin. It includes a table element that stores the table name and a row element that stores all of the columns in the CONTENT table: ID, VERSION, AUTHOR, LOCKER, CREATED, MODIFIED, ABSTRACT, BODY_HTML, and BODY_PDF. The XML includes @encoding parameters for the ABSTRACT, BODY_HTML, and BODY_PDF columns because they do not use the default encoding of `text` (they use `escaping` and `base64` instead). The row element also includes a children element that stores the child database table, contentContact.

Sample HTML markup file for CONTENT_CONTACT (the child table):

```
<html>
   <head>
      <title>CONTENT_CONTACT: child table markup</title>
   </head>
   <body>
      <!-- The table definition. -->
      <span id="psx-table/@name" />
      <span id="psx-table/row/CONTACTID" />
      <span id="psx-table/row/NAME" />
      <span id="psx-table/row/PHONE" />
      <span id="psx-table/row/EMAIL" />
   </body>
</html>
```

Because the CONTENT_CONTACT table is a child table, it does not require tabledefset and database elements. It includes a table element that stores the table name, and the columns CONTACTID, NAME, PHONE and EMAIL.

For explanations of the elements and attributes in these tables, see Parent Table HTML Markup Rules.

Creating the News Database Assembler Application

Drag and drop the two HTML markup files onto an empty application in the Workbench. XSpLit converts them into the content and contentContact resources as shown in the graphic below. To render the XML page correctly, delete the created stylesheets and add the default stylesheet from the menu option *insert > page*.

Then attach the ***sys_DatabasePublisher exit*** (see "sys_DatabasePublisher" on page 14) to the content resource. In this exit, use the default action replace ("r") and no aliasmap. Save it as xrd_casGeneric.

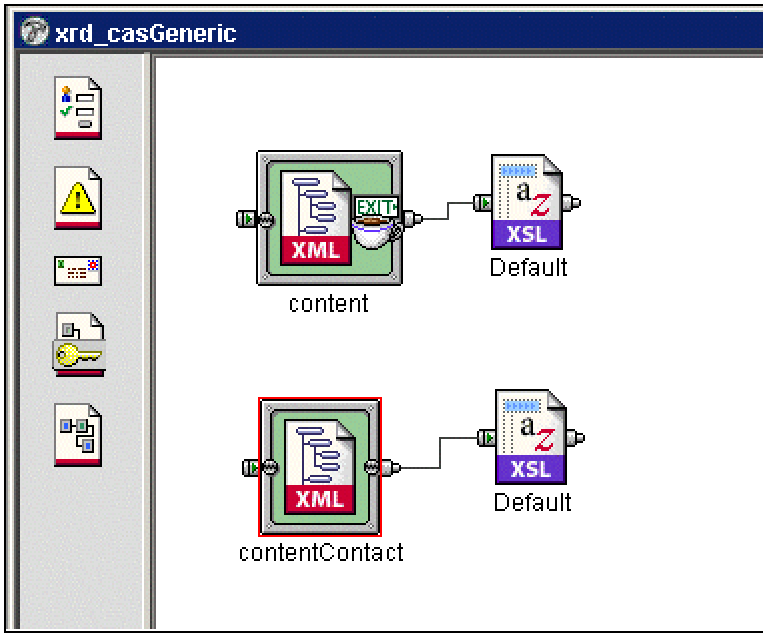


Figure 12: News Resource

Next, populate the application with the required backend table, selectors, and pagers, ***and ma***p it (see "Mapping the News content and contentContact resources" on page 45).

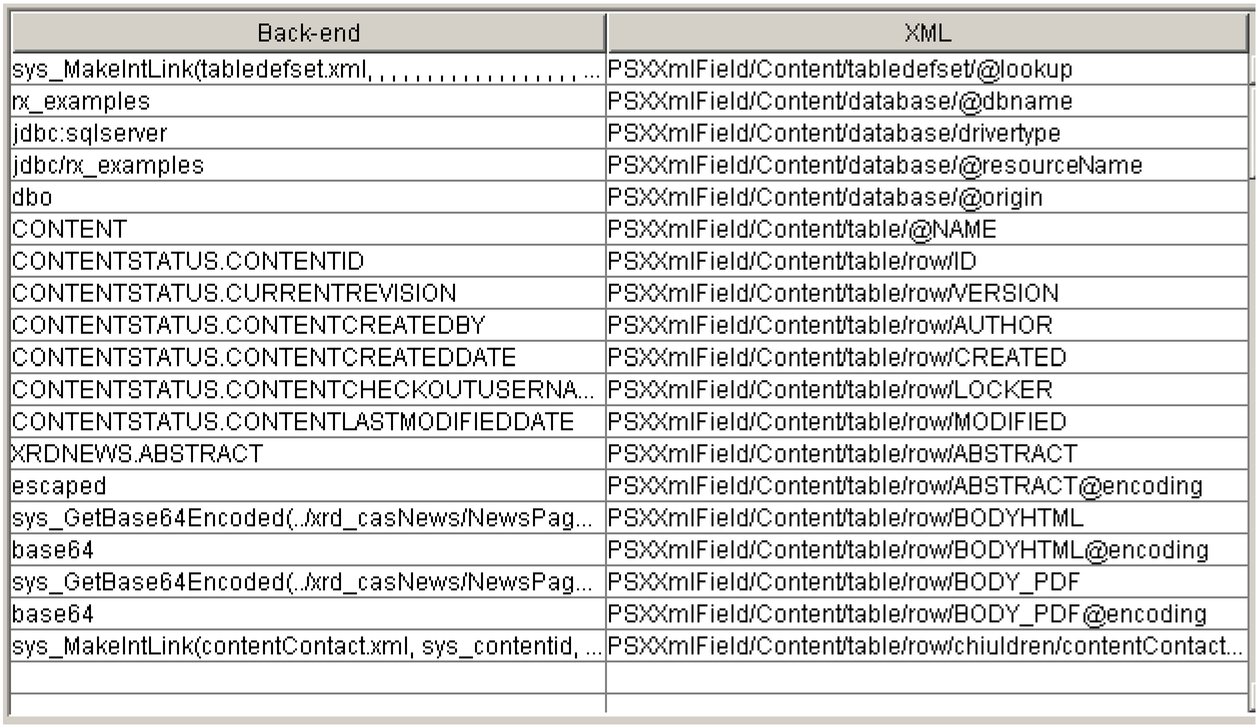
Mapping the News content and contentContact resources

Mapper for content resource:

Map the content table definition and the contentContact child table as internal lookup requests. Provide target database and connection information needed by the Database Publisher as literals.

The lookup for the table definition specifies the resource tabledefset.xml. You created this table definition with the ***Table Definition Builder*** (see "Creating the News Database Table Definition" on page 52).

For information about using the Mapper, see the section "Rhythmyx Application Programming Interface" in the *Rhythmyx CMS Online Help*.



Back-end	XML
sys_MakeIntLink(tabledefset.xml, , , , , , , , , , , , , , , , , , , ...	PSXXmlField/Content/tabledefset/@lookup
rx_examples	PSXXmlField/Content/database/@dbname
jdbc:sqlserver	PSXXmlField/Content/database/drivertype
jdbc/rx_examples	PSXXmlField/Content/database/@resourceName
dbo	PSXXmlField/Content/database/@origin
CONTENT	PSXXmlField/Content/table/@NAME
CONTENTSTATUS.CONTENTID	PSXXmlField/Content/table/row/ID
CONTENTSTATUS.CURRENTREVISION	PSXXmlField/Content/table/row/VERSION
CONTENTSTATUS.CONTENTCREATEDBY	PSXXmlField/Content/table/row/AUTHOR
CONTENTSTATUS.CONTENTCREATEDDATE	PSXXmlField/Content/table/row/CREATED
CONTENTSTATUS.CONTENTCHECKOUTUSERNA...	PSXXmlField/Content/table/row/LOCKER
CONTENTSTATUS.CONTENTLASTMODIFIEDDATE	PSXXmlField/Content/table/row/MODIFIED
XRDNEWS.ABSTRACT	PSXXmlField/Content/table/row/ABSTRACT
escaped	PSXXmlField/Content/table/row/ABSTRACT@encoding
sys_GetBase64Encoded(../xrd_casNews/NewsPag...	PSXXmlField/Content/table/row/BODYHTML
base64	PSXXmlField/Content/table/row/BODYHTML@encoding
sys_GetBase64Encoded(../xrd_casNews/NewsPag...	PSXXmlField/Content/table/row/BODY_PDF
base64	PSXXmlField/Content/table/row/BODY_PDF@encoding
sys_MakeIntLink(contentContact.xml, sys_contentid, ...	PSXXmlField/Content/table/row/chiuldren/contentContact...

Figure 13: Content Resource Mapper

Backend	XML	Comments
sys_MakeIntLink(tabledefset.xml)	PSXmlField/content/tabledefset/@lookup	The URL used by the sys_DatabasePublisher exit to make an internal request that looks up the tabledefset.xml from the application.
rx_examples	PSXmlField/content/database/@dbname	The name of the target database.
database driver type May be one of the following values, depending on your RDBMS: ▪ inetdae7 ▪ jtds:sqlserver ▪ oracle:thin ▪ oracle:oci ▪ db2	PSXmlField/content/database/@drivertype	The database driver type to use to perform database publishing.
jdbc/rx_examples	PSXmlField/content/database/@resourceName	The name of the resource.
dbo	PSXmlField/content/database/@origin	Name of the target database schema.
CONTENT	PSXmlField/content/table/@name	The name of the target table.
CONTENTSTATUS.CONTENTID	PSXmlField/content/table/row/ID	The content ID. This metadata field holds raw data from the CONTENTSTATUS source table.

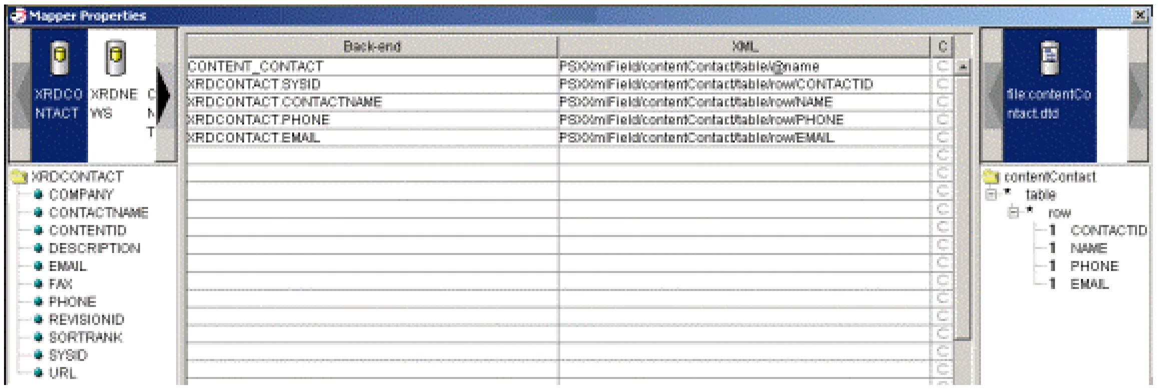
Backend	XML	Comments
CONTENTSTATUS.CURRENTREVISION	PSXmlField/content/table/row/VERSION	The content version. This metadata field holds raw data from the CONTENTSTATUS source table.
CONTENTSTATUS.CONTENTCREATEDBY	PSXmlField/content/table/row/AUTHOR	The content author. This metadata field holds raw data from the CONTENTSTATUS source table.
CONTENTSTATUS.CONTENTCREATEDDATE	PSXmlField/content/table/row/CREATED	The content creation date. This metadata field holds raw data from the CONTENTSTATUS source table.
CONTENTSTATUS.CONTENTCHECKOUTUSERNAME	PSXmlField/content/table/row/LOCKER	The user who currently has this content checked out or locked. This metadata field holds raw data from the CONTENTSTATUS source table.
CONTENTSTATUS.CONTENTLASTMODIFIEDDATE	PSXmlField/content/table/row/MODIFIED	The content's last modification date. This metadata field holds raw data from the CONTENTSTATUS source table.
XRDNEWS.ABSTRACT	PSXmlField/content/table/row/ABSTRACT	An abstract of the body. This content field holds HTML encoded with no escaping (`escaped`).
escaped	PSXmlField/content/table/row/ABSTRACT/@encoding	The type of encoding for the content Abstract. See ***Encoding Data*** (on page 16) for more information.

Backend	XML	Comments
sys_GetBase64Encoded(../xrd_casNews/NewsPage.html, sys_contentid, PSXParam/sys_contentid, sys_revision, PSXParam/sys_revision, sys_context, PSXParam/sys_context, sys_authtype, PSXParam/sys_authtype, sys_variantid, 318, sys_siteid, PSXParam/sys_siteid)	PSXmlField/content/table/row/BODY_HTML	The base64 encoded HTML variant of the body content. It is necessary to encode this content because it includes special characters. To encode the body text as base64, the sys_GetBase64Encoded exit requires parameters that form the URL of the variant. It locates the variant, base64 encodes its body, and returns it. In this example, sys_GetBase64Encoded encodes variantid 318 of the xrd_casNews assembler application. See "sys_GetBase64Encoded" in the *Rhythmyx CMS Online Help* for more information about the exit.
base64	PSXmlField/content/table/row/BODY_HTML/@encoding	The type of encoding for the HTML body variant. See ***Encoding Data*** (on page 16) for more information. .
sys_GetBase64Encoded(../xrd_casNews/NewsPage.html, sys_contentid, PSXParam/sys_contentid, sys_revision, PSXParam/sys_revision, sys_context, PSXParam/sys_context, sys_authtype, PSXParam/sys_authtype, sys_variantid, 319, sys_siteid, PSXParam/sys_siteid)	PSXmlField/content/table/row/BODY_PDF	The `base64` encoded PDF variant of the body content. It is necessary to encode this content because it includes special characters. In this example, sys_GetBase64Encoded encodes variantid 319 of the xrd_casNews assembler application. See the comments for PSXmlField/content/table/row/BODY_HTML for details about using the output of the sys_GetBase64Encoded exit as a field.

Backend	XML	Comments
base64	PSXmlField/content/table/row/BODY_PDF/@encoding	The type of encoding for the content body. See ***Encoding Data*** (on page 16) for more information.
sys_MakeIntLink(contentContact.xml, sys_contentid, PSXParam/sys_contentid, sys_revision, PSXParam/sys_revision, sys_context, PSXParam/sys_context, sys_authtype, PSXParam/sys_authtype, sys_siteid, PSXParam/sys_siteid)	PSXmlField/content/table/row/children/contentContact/@lookup	Creates an internal lookup to the child table resource contentContact. See "sys_MakeIntLink" in the *Rhythmyx CMS online help* for more information.

Mapper for contentContact resource

The table name is provided as a literal. All other columns are mapped as usual. Note that you do not need to map the primary key in childtables, it is provided by the Database Publisher.



Back-end	XML	C
CONTENT_CONTACT	PSXXmlField/contentContact/table/@name	
XRDCONTACT.SYSID	PSXXmlField/contentContact/table/row/CONTACTID	
XRDCONTACT.CONTACTNAME	PSXXmlField/contentContact/table/row/NAME	
XRDCONTACT.PHONE	PSXXmlField/contentContact/table/row/PHONE	
XRDCONTACT.EMAIL	PSXXmlField/contentContact/table/row/EMAIL	

Figure 14: Content_contact Resource Mapper

Backend	**XML**	**Comments**
CONTENT_CONTACT	PSXmlField/contentContact/table/@name	The name of the target table. This is a literal.
XRDCONTACT.SYSID	PSXmlField/contentContact/table/row/CONTACTID	The contact ID number. This metadata field holds raw data from the XRDCONTACT source table.
XRDCONTACT.CONTACTNAME	PSXmlField/contentContact/table/row/NAME	The contact name. This metadata field holds raw data from the XRDCONTACT source table.
XRDCONTACT.PHONE	PSXmlField/contentContact/table/row/PHONE	The contact phone number. This metadata field holds raw data from the XRDCONTACT source table.
XRDCONTACT.EMAIL	PSXmlField/documentContact/table/row/EMAIL	The contact email address. This metadata field holds raw data from the XRDCONTACT source table.

You must check the **Return empty XML** checkbox in all mappers for childtables. Otherwise the Database Publisher tries to enter columns with invalid (empty) keys.

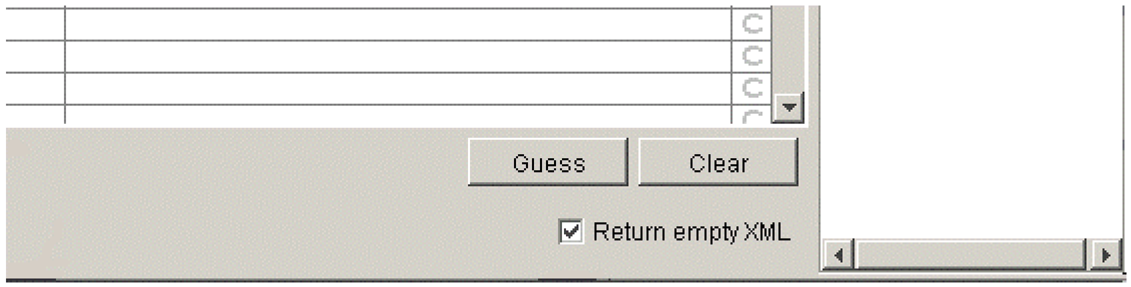
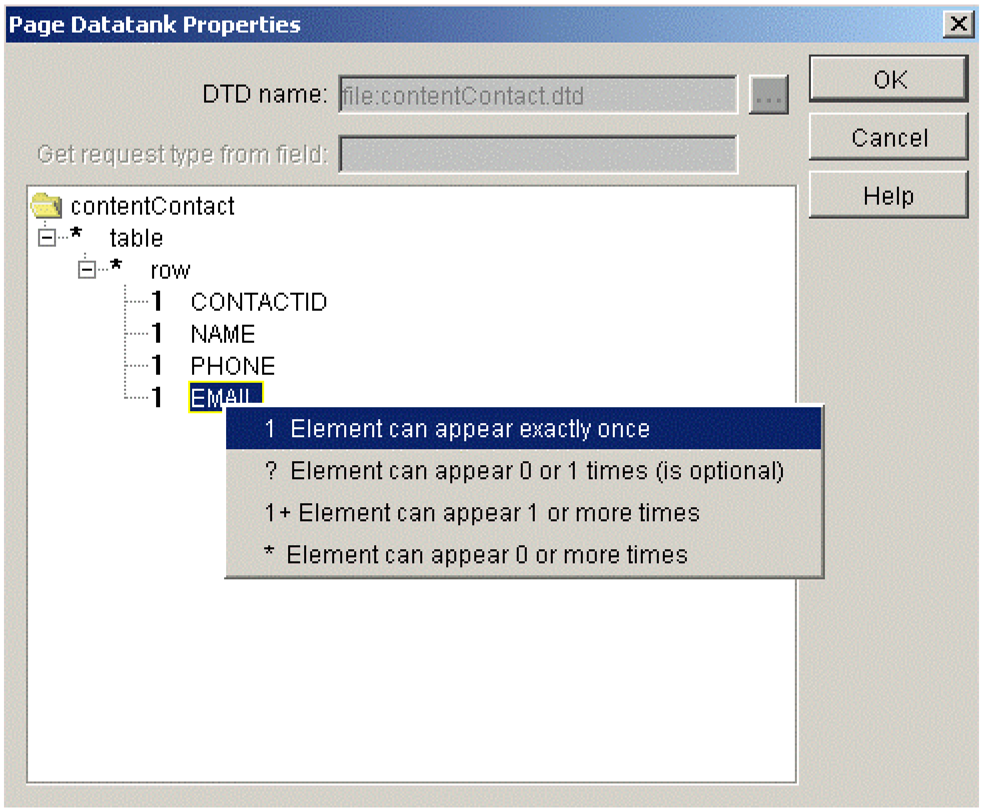


Figure 15: Mapper displaying Return empty XML checkbox

Specifying Column Repetition in News Resources

The Rhythmyx Workbench specifies that all XML columns can appear n times per row (** Element can appear 0 or more times*). This may cause columns to repeat in row elements instead of appearing in separate rows. To prevent this in the News database, change the definition of each column in the contentContact resource in the Page Datatank.


Page Datatank Properties

DTD name: file:contentContact.dtd

Get request type from field:

contentContact
* table
* row
1 CONTACTID
1 NAME
1 PHONE
1 EMAIL

OK
Cancel
Help
1 Element can appear exactly once
? Element can appear 0 or 1 times (is optional)
1+ Element can appear 1 or more times
* Element can appear 0 or more times

Figure 16: Page Datatank Properties

To change the definition of the column:

1 Open the Page Datatank for the XML resource.

2 Right-click on each column name (CONTACTID, NAME, PHONE, and EMAIL) and select *1 Element can appear exactly once* in the drop list.

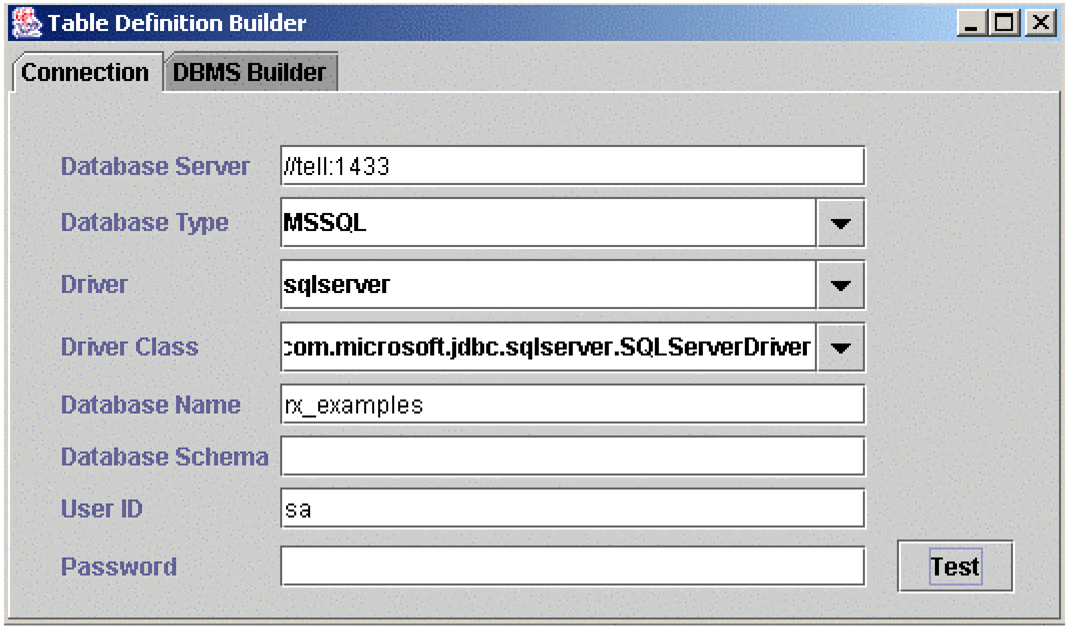
3 Click [**OK**].

NOTE: You must repeat this step each time you drag and drop the resource HTML file.

Creating the News Database Table Definition

Create the table definition file for the Database Publisher to use to determine the target schema. Use the Table Definition Builder to create the file. For help using and filling out the fields in the Table Definition Builder, see the section Table Definition Builder and its sub-topics.

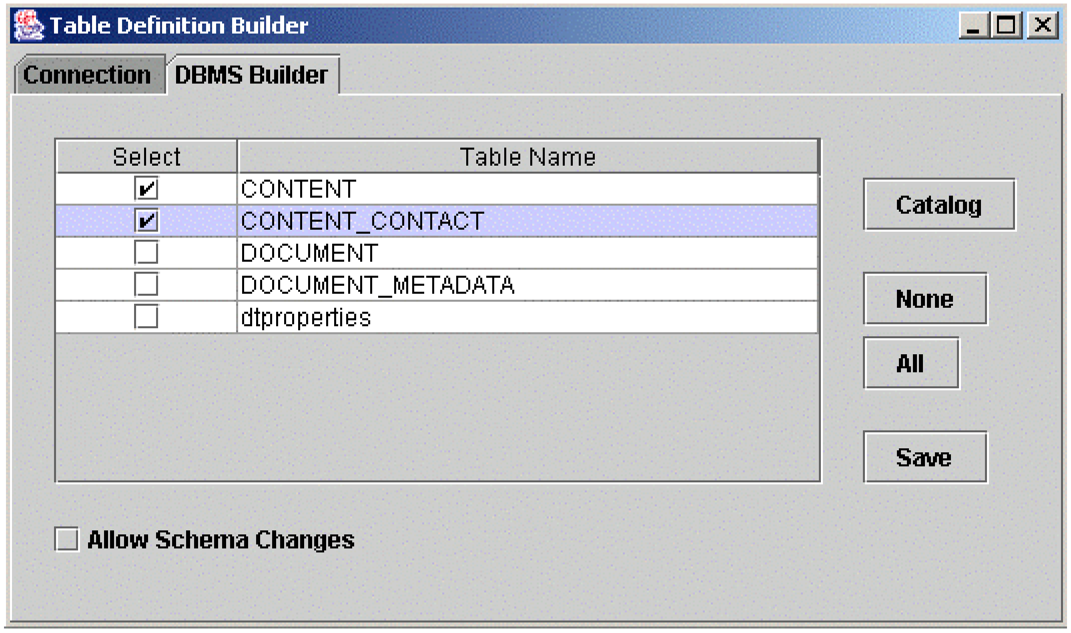
In the Connection tab of this tool, provide the target database connection information. Test the connection with the [**Test**] button.



Database Server	//tell:1433
Database Type	MSSQL
Driver	sqlserver
Driver Class	com.microsoft.jdbc.sqlserver.SQLServerDriver
Database Name	rx_examples
Database Schema	
User ID	sa
Password	

Figure 17: Table Definition Builder, Connection Tab

Click the DBMS Builder tab. Click [**Catalog**] to catalog all tables available through the connection specified. Select the CONTENT and CONTENT_CONTACT tables. Click [**Save**] to save the table definition to the file system as tabledefset.xml. The created file conforms to the ***sys_Tabledef.dtd*** (see "Database Publisher DTDs" on page 26).



Select	Table Name
✓	CONTENT
✓	CONTENT_CONTACT
☐	DOCUMENT
☐	DOCUMENT_METADATA
☐	dtproperties

☐ Allow Schema Changes

Figure 18: Table Definition Builder, DBMS Tab

To add tabledefset.xml as a static resource to the assembler application, insert it into the xrd_casGeneric directory in the Rhythmyx root after you create the application.

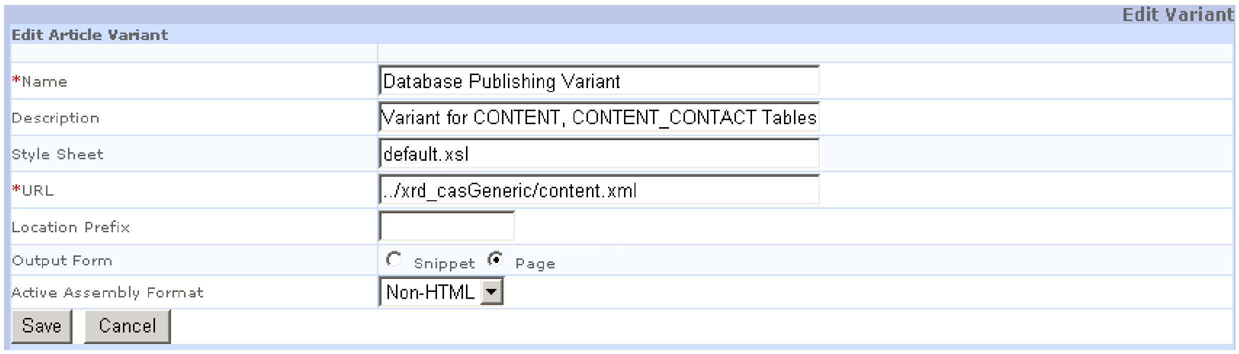
tabledefset.xml example

For definitions of the elements and attributes in this file, see <Rhythmyx root>/DTD/sys_Tabledef.dtd.

```
<?xml version="1.0" encoding="UTF-8"?>
<tabledefset>
   <tabledef allowSchemaChanges="n" alter="n" create="y" delolddata="n"
name="CONTENT">
      <rowdef>
         <columndef action="c" name="ID">
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>no</allowsnull>
         </columndef>
         <columndef action="c" name="VERSION">
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="AUTHOR">
            <jdbctype>VARCHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="LOCKER">
            <jdbctype>VARCHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="CREATED">
            <jdbctype>TIMESTAMP</jdbctype>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="MODIFIED">
            <jdbctype>TIMESTAMP</jdbctype>
            <allowsnull>yes</allowsnull>
          </columndef>
         <columndef action="c" name="ABSTRACT">
            <jdbctype>LONGVARCHAR</jdbctype>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="BODY_HTML">
            <jdbctype>LONGVARCHAR</jdbctype>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="BODY_PDF">
            <jdbctype>BINARY</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
      </rowdef>
      <primarykey action="c">
         <name>ID</name>
      </primarykey>
   </tabledef>
   <tabledef allowSchemaChanges="n" alter="n" create="y" delolddata="n"
name="CONTENT_CONTACT">
      <rowdef>
         <columndef action="c" name="ID">
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>no</allowsnull>
         </columndef>
         <columndef action="c" name="CONTACTID">
            <jdbctype>INTEGER</jdbctype>
            <allowsnull>no</allowsnull>
         </columndef>
         <columndef action="c" name="NAME">
            <jdbctype>VARCHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="PHONE">
            <jdbctype>VARCHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
         <columndef action="c" name="EMAIL">
            <jdbctype>VARCHAR</jdbctype>
            <size>50</size>
            <allowsnull>yes</allowsnull>
         </columndef>
      </rowdef>
      <primarykey action="c">
         <name>ID</name>
         <name>CONTACTID</name>
      </primarykey>
      <foreignkey action="c">
         <fkColumn>
            <name>ID</name>
            <externalTable>CONTENT</externalTable>
            <externalColumn>ID</externalColumn>
         </fkColumn>
      </foreignkey>
   </tabledef>
</tabledefset>
```

Registering the Database Assembly Variant

After you create the Database Assembler application, register the Variant that formats the data for the CONTENT and CONTENT_CONTACT tables.



*Name	Database Publishing Variant
Description	Variant for CONTENT, CONTENT_CONTACT Tables
Style Sheet	default.xsl
*URL	../xrd_casGeneric/content.xml
Location Prefix	
Output Form	Snippet [✓ Page]
Active Assembly Format	Non-HTML

Figure 19: Database Assembler Variant

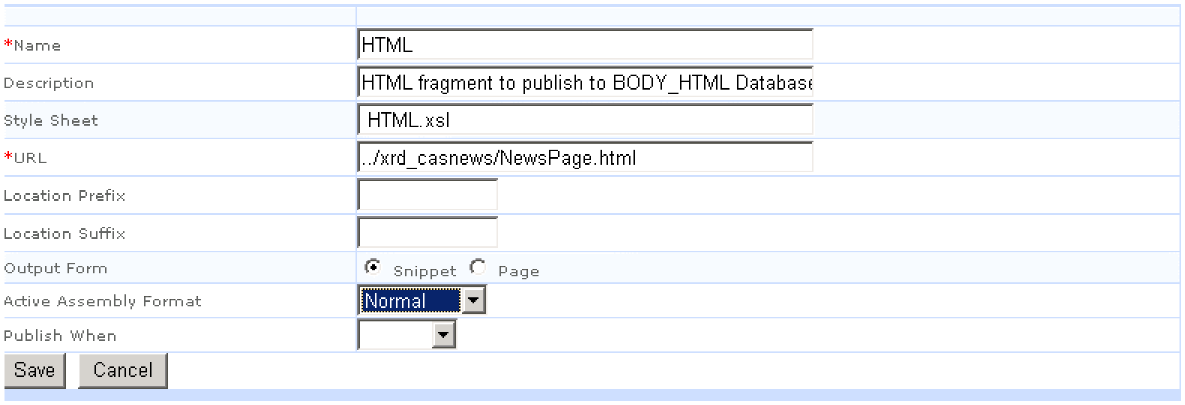
To create the Database Assembly Variant:

1. In **Name**, enter a name for the Variant.
2. Optionally, in **Description**, enter a description for the Variant.
3. In **Style Sheet**, enter *default.xsl*.
4. Choose *Page* as the **Output Form**.
5. Choose *Non-HTML* as the **Active Assembly Format**.

Developing Variants, Resources, and Slots

If your existing Variant assembly applications assemble body content into the format that you want to store it in your target database, you may modify them or use them as they are to assemble content that you publish to your database. However, you may also want to create new resource files, Variants, and Slots for the particular formats in which you want to store content in your target database. See the instructions in the *Rhythmyx Workbench Online Help*, "Managing Applications/Defining Page Content" section and the topic "Defining Variants" in the document *Implementing Content Editors and Content Assembly in Rhythmyx* for general help on creating assembler resources. After you create Variants and Slots in your resources, you must register them with Rhythmyx.

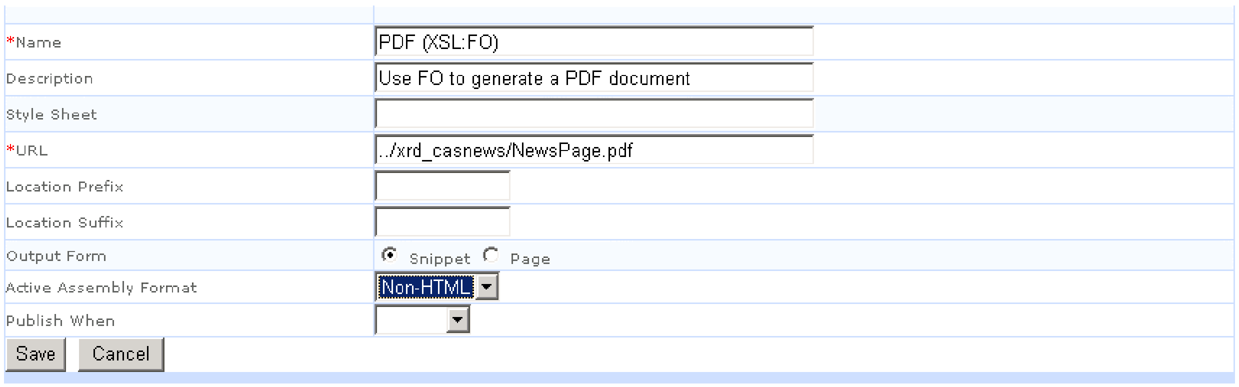
The following graphic shows the sample registration for the HTML Variant for the News Content Type. In this example, Snippet is selected as the output form because the Variant represents a portion of a page. This sample Variant does not include any Slots.



*Name	HTML
Description	HTML fragment to publish to BODY_HTML Database
Style Sheet	HTML.xsl
*URL	../xrd_casnews/NewsPage.html
Location Prefix	
Location Suffix	
Output Form	[✓ Snippet] Page
Active Assembly Format	Normal
Publish When	

Figure 20: HTML Variant Registration

The following graphic shows the sample registration for the PDF Variant for the News Content Type. In this example, Page is selected as the output form because you are creating a PDF file. It is not necessary to specify a Style Sheet when you are creating a PDF (or any binary file). The Active Assembly format is Non-HTML because the Variant is in file format.



*Name	PDF (XSL:FO)
Description	Use FO to generate a PDF document
Style Sheet	
*URL	../xrd_casnews/NewsPage.pdf
Location Prefix	
Location Suffix	
Output Form	[✓ Snippet] Page
Active Assembly Format	Non-HTML
Publish When	

Figure 21: PDF Variant Registration

To register your Variants:

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

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

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

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

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

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

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

Variant Assembler Applications

The content resource in the xrd_casGeneric application obtains Variants produced by the xrd_casNews assembler application and inserts the Variants into the BODY_HTML field and BODY_PDF field of the CONTENT database table.

The graphic below shows xrd_casNews as it appears in the Rhythmyx Workbench. html.xsl produces the HTML Variant and pdf.xsl produces the PDF Variant.

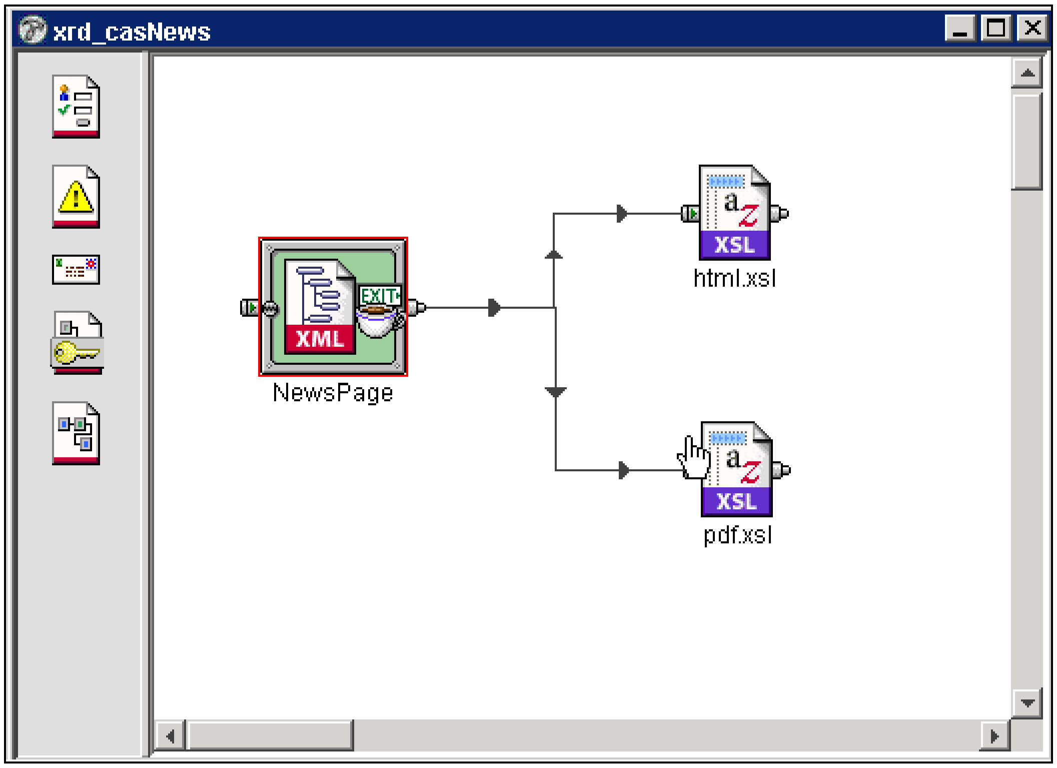
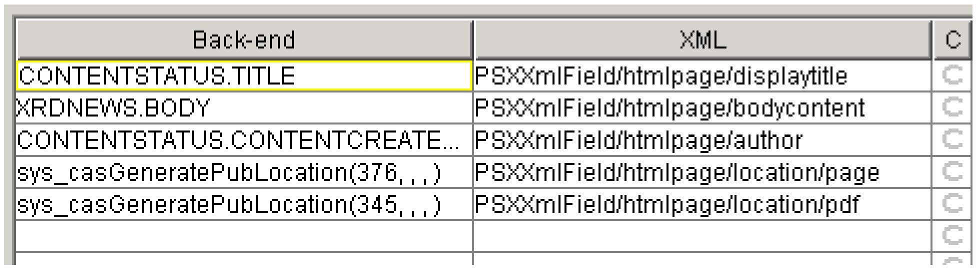


Figure 22: xrd_casNews

The html and pdf resources map information to the fields in the XML file like other Rhythmyx assembly resources. The following graphic shows the mapping for the XML resource. Back-end data is mapped to the displaytitle, bodycontent, and author elements. The HTML Variant (376) is mapped to the location/page element of the XML resource and the PDF Variant (345) is mapped to the location/pdf element of the XML resource:



Back-end	XML	C
CONTENTSTATUS.TITLE	PSXXmlField/htmlpage/displaytitle	C
XRDNEWS.BODY	PSXXmlField/htmlpage/bodycontent	C
CONTENTSTATUS.CONTENTCREATE...	PSXXmlField/htmlpage/author	C
sys_casGeneratePubLocation(376, , ,)	PSXXmlField/htmlpage/location/page	C
sys_casGeneratePubLocation(345, , ,)	PSXXmlField/htmlpage/location/pdf	C
		C

Figure 23: NewsPage Mapper

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

1 Drag and drop the resource HTML for your Variants in a new application in the Workbench.

   a) Attach the sys_casAddAssemblerInfo post-exit to any text resource.

   b) Attach the sys_xdTextToTree post-exit to any text resource that that uses the sys_EditLive control.

   c) Create static resources for image files and binary files and map them. See "Mapping Images and Other Mime Types" in the *Rhythmyx Workbench* online help.

   d) For text resources, configure the back end tables, mappers, and selectors as specified in the "Creating Assembly Applications" section in the Rhythmyx document *Implementing Content Editors and Content Assembly*.

   e) For resources that create inline images or inline links, requests are not made to backend tables, so add the RXDUAL "dummy" table to your datatank and do not define any requestor properties. Map elements as explained in the topic "Creating Inline Image and Inline Link Assemblers" in the *Implementing Content Editors and Content Assembly* document.

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

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

For more information about creating PDF Variants, see the section "Assembling a PDF" in the *Rhythmyx Workbench* online help.

Building the News Content List Application

Build a content list application for the News target database with a content list resource for the Database Publisher that has its delivery type mapped to the literal `database`.

1. Open a new application.
2. Copy and rename the contentlist_generic resource from the rx_PubContentLists application in the new application, or create a resource from scratch.
3. Define request properties for the content list.
4. Open the Resource Editor.

   a) On the Page Datatank, drag and drop `contentlist.dtd`.

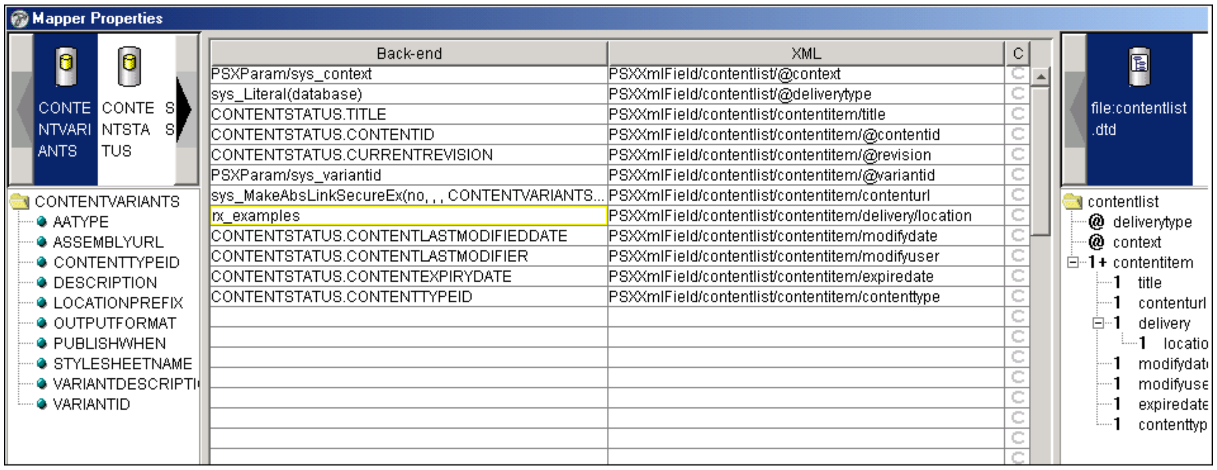
   b) On the Backend Datatank, drag and drop the tables CONTENTVARIANTS, CONTENTSTATUS, and STATES. Join the columns CONTENTVARIANTS.CONTENTTYPEID and CONTENTSTATUS.CONTENTTYPEID and the columns CONTENTSTATUS.CONTENTSTATEID and STATEID.

   c) In the Selector Properties, enter:

   ```
   CONTENTVARIANTS.VARIANTID = PSXParam/sys_variantid
   ```

   d) In the Result Page Properties, insert a pager that sorts on CONTENTSTATUS.CONTENTID.

   e) Enter the following in Mapper Properties:



Back-end	XML	C
PSXParam/sys_context	PSXXmlField/contentlist/@context	C
sys_Literal(database)	PSXXmlField/contentlist/@deliverytype	C
CONTENTSTATUS.TITLE	PSXXmlField/contentlist/contentitem/title	C
CONTENTSTATUS.CONTENTID	PSXXmlField/contentlist/contentitem/@contentid	C
CONTENTSTATUS.CURRENTREVISION	PSXXmlField/contentlist/contentitem/@revision	C
PSXParam/sys_variantid	PSXXmlField/contentlist/contentitem/@variantid	C
sys_MakeAbsLinkSecureEx(no, , , CONTENTVARIANTS...	PSXXmlField/contentlist/contentitem/contenturl	C
rx_examples	PSXXmlField/contentlist/contentitem/delivery/location	C
CONTENTSTATUS.CONTENTLASTMODIFIEDDATE	PSXXmlField/contentlist/contentitem/modifydate	C
CONTENTSTATUS.CONTENTLASTMODIFIER	PSXXmlField/contentlist/contentitem/modifyuser	C
CONTENTSTATUS.CONTENTEXPIRYDATE	PSXXmlField/contentlist/contentitem/expiredate	C
CONTENTSTATUS.CONTENTTYPEID	PSXXmlField/contentlist/contentitem/contenttype	C

Figure 24: News Content List Application Mapper

Backend	**XML**
PSXParam/sys_context	PSXXmlField/contentlist@context
sys_literal(database)	PSXXmlField/contentlist/@deliverytype
CONTENTSTATUS.TITLE	PSXXmlField/contentlist/contentitem/title

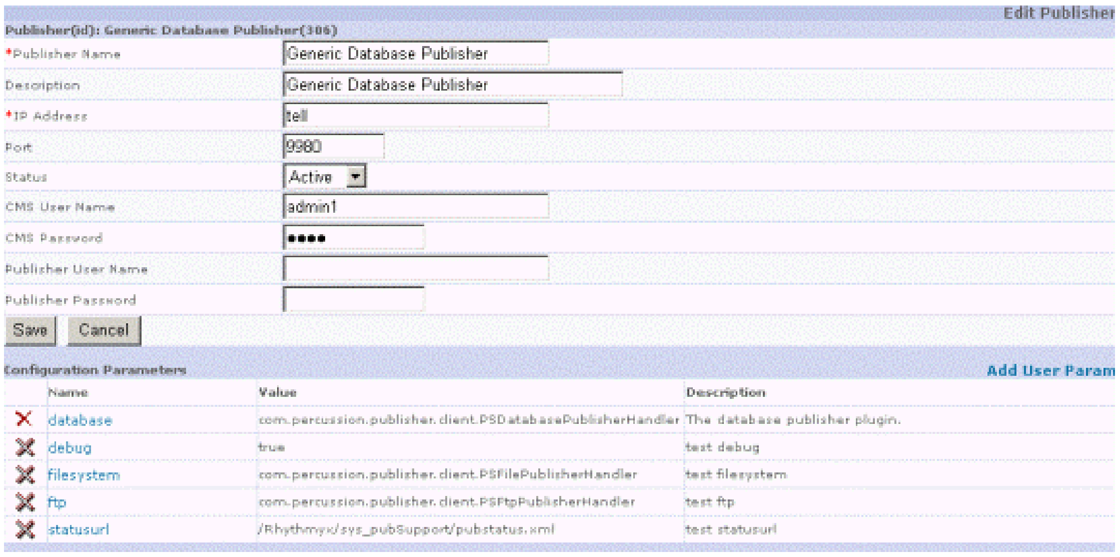
Backend	XML
CONTENTSTATUS.CONTENTID	PSXXmlField/contentlist/contentitem/@contentid
CONTENTSTATUS.CURRENTREVISION	PSXXmlField/contentlist/contentitem/@revision
PSXParam/sys_variantid	PSXXmlField/contentlist/@variantid
sys_MakeAbsLinkSecureEx(no,,,CONTENT.XML, sys_contentid, CONTENTSTATUS.CONTENTID, sys_revision, CONTENTSTATUS.CURRENTREVISION, sys_variantid, PSXParam/sys_variantid, sys_context,PSXParam/sys_context,sys_authtype, PSXParam/sys_authtype)	PSXXmlField/contentlist/contentitem/contenturl
sys_literal(rx_examples) NOTE: The sys_literal parameter is the database name.	PSXXmlField/contentlist/contentitem/delivery/location
CONTENTSTATUS.CONTENTLASTMODIFIEDDATE	PSXXmlField/contentlist/contentitem/modifydate
CONTENTSTATUS.CONTENTLASTMODIFIER	PSXXmlField/contentlist/contentitem/modifyuser
CONTENTSTATUS.CONTENTEXPIREDATE	PSXXmlField/contentlist/contentitem/expiredate
CONTENTSTATUS.CONTENTTYPEID	PSXXmlField/contentlist/contentitem/contenttype

5 Save the content list application.

Registering News Publishing Components

Register all the components for the Database Publisher in the Publishing Administrator of the CMS. See the document *Implementing Publishing in Rhythmyx* for complete information about registering each component.

1. Register a new Database Publisher in the Edit Publisher page. Add a configuration parameter for the Database Publisher plugin called **database**. Set the value of this parameter to `com.percussion.publisher.client.PSDatabasePublisherHandler`.



Edit Publisher

Publisher(id): Generic Database Publisher(306)

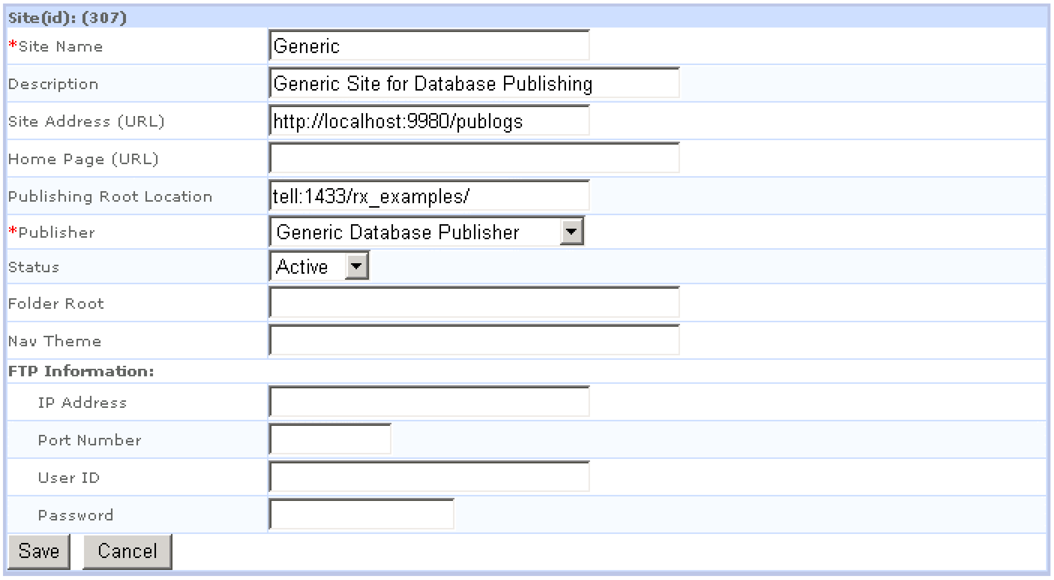
*Publisher Name	Generic Database Publisher
Description	Generic Database Publisher
*IP Address	tell
Port	9980
Status	Active
CMS User Name	admin1
CMS Password	●●●●
Publisher User Name	
Publisher Password	

Configuration Parameters

	Name	Value	Description
✗	database	com.percussion.publisher.client.PSDatabasePublisherHandler	The database publisher plugin.
✗	debug	true	test debug
✗	filesystem	com.percussion.publisher.client.PSFilePublisherHandler	test filesystem
✗	ftp	com.percussion.publisher.client.PSFtpPublisherHandler	test ftp
✗	statusurl	/Rhythmyx/sys_pubSupport/pubstatus.xml	test statusurl

Figure 25: Publisher Registration for News Publisher

2. Register a site for the Database Publisher. The **Publishing Root Location** should be the label that you want to appear in the ***Publication Details Map***.



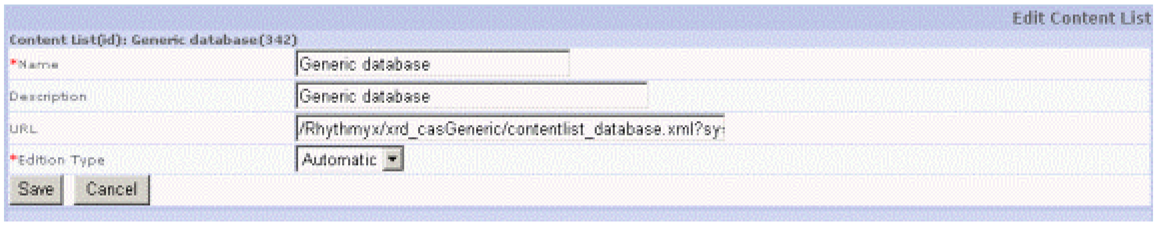
Site(id): (307)

*Site Name	Generic
Description	Generic Site for Database Publishing
Site Address (URL)	http://localhost:9980/publogs
Home Page (URL)	
Publishing Root Location	tell:1433/rx_examples/
*Publisher	Generic Database Publisher
Status	Active
Folder Root	
Nav Theme	
FTP Information:	
IP Address	
Port Number	
User ID	
Password	

Figure 26: Site Registration

3 Register the content list that will publish the News content to the target database. The content list URL should have the format:
`/<Rhythmyx root>/<content list application name>/<content list query>?sys_variantid=<# of application that uses source table>`

For the news example, the content list URL is:
`/Rhythmyx/xrd_casGeneric/contentlist_database.xml?sys_variantid=329`

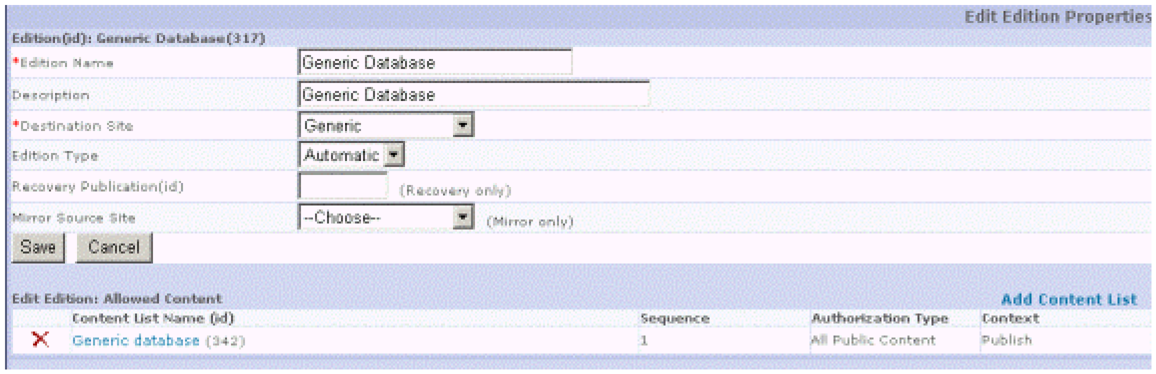

Edit Content List

Content List(id): Generic database(342)

*Name	Generic database
Description	Generic database
URL	/Rhythmyx/xrd_casGeneric/contentlist_database.xml?sy
*Edition Type	Automatic

Figure 27: Content List Registration for News Content List

4 Register the Edition that will publish the database. Include the content list created in the previous step.


Edit Edition Properties

Edition(id): Generic Database(317)

*Edition Name	Generic Database
Description	Generic Database
*Destination Site	Generic
Edition Type	Automatic
Recovery Publication(id)	(Recovery only)
Mirror Source Site	--Choose-- (Mirror only)

Edit Edition: Allowed Content

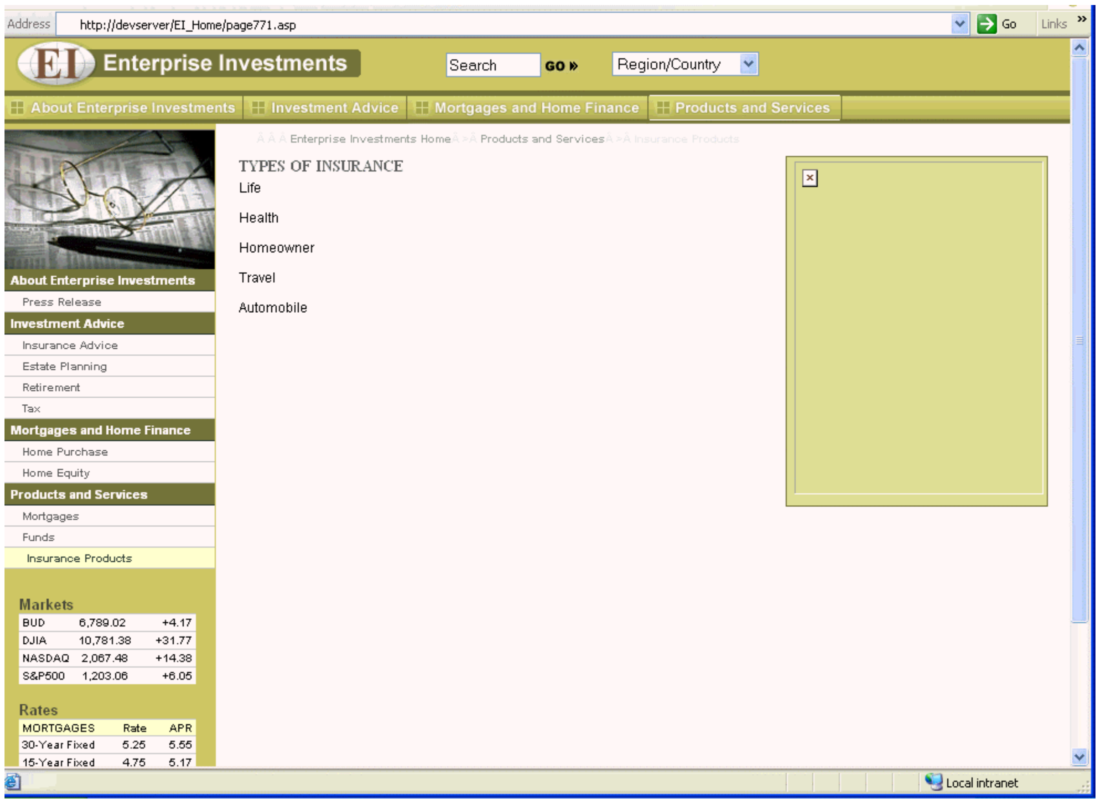
	Content List Name (id)	Sequence	Authorization Type	Context
✕	Generic database (342)	1	All Public Content	Publish

Figure 28: Edition Registration for News Edition

Configuring Previews of Dynamic Content

Some of your Variants may use code (for example, JSP or ASP code) that enables them to perform dynamic processing, such as retrieving current data from a database, variable substitution, or conditional branching. If you attempt to preview these Variants in Rhythmyx as you preview other Variants, the Rhythmyx server cannot serve the dynamic processing code, and the preview Page may return an error or render improperly. One way to avoid errors is to add conditional logic to your stylesheet so that it replaces dynamic areas of the page with placeholder text and images when you perform a preview. Another option is to perform the preview on the server that performs the dynamic processing (the Web application server). Configuring this type of preview involves manually publishing the Content Item as its preview Variant to the Web application server. When a user chooses Preview > [preview Variant] in the Action Menu for the item, Rhythmyx redirects the user to the assembled item on the Web application server, and the user is able to preview the Content Item with the dynamic content.

We illustrate the procedure for configuring this type of preview with an example of a Variant of a FastForward Generic Content Type that displays dynamic data. The Variant creates an .asp file that Rhythmyx publishes to an IIS Web application server for previewing. When previewed, our sample page with dynamic data appears as:


Address
http://devserver/EI_Home/page771.asp
Go
Links
Enterprise Investments
Search
GO »
Region/Country
About Enterprise Investments
Investment Advice
Mortgages and Home Finance
Products and Services
Enterprise Investments Home > Products and Services > Insurance Products
Local intranet

TYPES OF INSURANCE

Life

Health

Homeowner

Travel

Automobile

About Enterprise Investments
- Press Release

Investment Advice
- Insurance Advice
- Estate Planning
- Retirement
- Tax

Mortgages and Home Finance
- Home Purchase
- Home Equity

Products and Services
- Mortgages
- Funds
- Insurance Products

Markets

BUD	6,789.02	+4.17
DJIA	10,781.38	+31.77
NASDAQ	2,067.48	+14.38
S&P500	1,203.06	+6.05

Rates

MORTGAGES	Rate	APR
30-Year Fixed	5.25	5.55
15-Year Fixed	4.75	5.17

Figure 29: Content Item viewd through dynamic preview

Note: The graphic in the upper right of the page is missing because it has not been published to the preview site.

The dynamic portion of the page is the table under the markets header in the left panel. The table data is retrieved from an external database that stores updated market information.

Dynamic previewing uses two types of Variants that we refer to as assembly Variants and preview Variants. Assembly Variants are typical Rhythmyx Variants that format content for viewing. Preview Variants supply URLs that redirect publishing of the assembly Variants to the Web application server instead of the default publish location.

To configure previewing of dynamic content:

1 Add the code that accesses the dynamic information to a Variant of the Content Type.

2 ***Configure your Web application server to connect to your database*** (see "Configuring Your Web Application Server to Connect to Your Database" on page 71).

3 ***Set up the dynamic preview Site*** (see "Setting up the Dynamic Preview Site" on page 72).

4 ***Register a dynamic preview Context and dynamic preview Location Scheme*** (see "Registering a Dynamic Preview Context and Location Scheme" on page 74).

5 ***Register a dynamic preview Location Scheme in the Publish Context*** (see "Registering a Dynamic Preview Location Scheme in the Publish Context" on page 75).

6 ***Define publishing variable values for the preview Site*** (see "Defining Publishing Variable Values for the Preview Site" on page 77).

7 ***Create your preview Variant*** (see "Creating the Dynamic Preview Variant" on page 79).

8 ***Set up the dynamic preview Manual Edition and dynamic preview Content List*** (see "Setting Up the Dynamic Preview Content List" on page 83).

9 ***Test the dynamic preview*** (see "Testing the Dynamic Preview" on page 86).

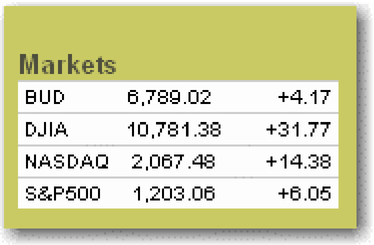
After you set up your system following these steps, users will automatically be redirected to the Web application server that serves the dynamic content when they choose Preview > [dynamic preview Variant] in an Action Menu.

Adding Code that Accesses the Dynamic Information

In order to add dynamic information to a Content Item, you must include script for accessing and manipulating data from the database that contains the dynamic information. If you want to preview the dynamic information, you must view it on a Web application server that can read the dynamic script. In the Variant for the Content Item, you may either include the ASP script in an XSL statement or you may embed an XSL statement that calls the ASP script which is included in another file. The benefit of maintaining the script in a separate location is that the XSL embedded in the Variant remains simpler, and the ASP script is easier to reuse and troubleshoot. However, if the ASP script is simple, it may be easier to include it in the Variant.

In our example, we want the dynamic information to appear in the FastForward global template enterprise-global-template.html. We embed an XSL statement that calls ASP script in the HTML of the global template.

In this example, the script is included in the adovbs.inc file, which IIS uses to access the database that contains the dynamic information. In addition to connection information, our adovbs.inc file includes the ASP script that creates the portion of the page that displays the dynamic data. Our adovbs.inc file connects to a database of stock market information and presents a table of current market data in the global template that appears as:


Markets

BUD	6,789.02	+4.17
DJIA	10,781.38	+31.77
NASDAQ	2,067.48	+14.38
S&P500	1,203.06	+6.05

Figure 30: Dynamic Content

The adovbs.inc file is included in the directory on the Web application Server where the files for preview are published, in our example, Inetpub\wwwroot\EI_Home.

ASP Script

Our example adovbs.inc file which includes our ASP script is shown below. The first six lines create the connection to the marketsdb database and establish that data is being retrieved from the MARKETS table. The code that follows creates a table of dynamic market information using data from the MARKETID, RATE, and CHANGE columns in the MARKETS table:

```
 <%
set con = server.createobject("ADODB.Connection")
set cmd = Server.CreateObject("ADODB.Command")

con.open "PROVIDER=SQLOLEDB;DATA
SOURCE=MKTSERVER\MKTDATA;UID=sa;PWD=password;DATABASE=marketsdb"
Set cmd.ActiveConnection = con
cmd.CommandText = "SELECT * FROM MARKETS"

'Creates a read-only, forward only recordset
Set rs = cmd.execute

'response.write "<table width='160' border='1' cellspacing='0'
cellpadding='0' bgcolor='#FFFFFF' bordercolor='#666666'>"
Do While Not rs.EOF
   'wrapper table
   response.write "<tr>"
   response.write "<td width='53'>"
   response.write "<table width='160' border='0' cellspacing='0'
cellpadding='0' class='bodyblacksmall'>"
   response.write "<tr>"
   For iCtr = 0 To rs.fields.Count - 1
      If rs.fields(iCtr).Name = "MARKETID" Or rs.fields(iCtr).Name =
"RATE" Then
         response.write "<td align='left' valign='middle' width='50'
class='bodyblacksmall' height='10'>"
         If rs.fields(iCtr).Name = "RATE" Then
            response.write FormatNumber(rs.fields(iCtr).Value, 2)
         Else
            response.write rs.fields(iCtr).Value
         End If
         response.write "</td>"
      End If
      If rs.fields(iCtr).Name = "CHANGE" Then
         response.write "<td align='right' valign='middle' width='50'
class='bodyblacksmall' height='10'>"
         If CInt(rs.fields(iCtr).Value) > 0 Then
            response.write "+"
         End If
         response.write FormatNumber(rs.fields(iCtr).Value, 2) & "</td>"
      End If
   Next
   response.write "</tr></table></td></tr>"
   rs.MoveNext
Loop

'cleanup
Set rs = Nothing
Set cmd = Nothing
con.Close
set conn = nothing
%>
```

Embedded XSL Statement

In our example, we embed the XSL statement that calls the adovbs.inc file in our FastForward enterprise-global-template.html because the dynamic table appears in the global template portion of the Web page. Below, we include the beginning of the the horizontal_nav portion of the file, where the markets table is coded. The adovbs.inc include statement is in bold. It is followed by commented out code for a markets table including static information. Note: The enterprise-global-template.html file is located in <Rhythmyx root>/rxs_GlobalTemplates.

```
  <!-- ==========  HORIZONTAL NAV STARTS HERE ========== -->
        <div id="horizontal_nav">
          <!-- start slot nav_top --><xsl:variable name="rxslot-
enterprise-global-template-12"><rxslot psxeditslot="no"
slotname="nav_top" template="nav_top"><xsl:copy-of
select="$related/linkurl[@slotname=&#39;nav_top&#39;]"/></rxslot></xsl:v
ariable><xsl:apply-templates mode="rxslot-enterprise-global-template"
select="$rxslot-enterprise-global-template-12/*"/><!-- end slot nav_top
-->
        </div>
      </div>
      <div id="MainPortion">
        <div id="LeftSide">
          <div id="LeftNav">
            <!-- start slot nav_left --><xsl:variable name="rxslot-
enterprise-global-template-14"><rxslot psxeditslot="no"
slotname="nav_left" template="nav_left"><xsl:copy-of
select="$related/linkurl[@slotname=&#39;nav_left&#39;]"/></rxslot></xsl:
variable><xsl:apply-templates mode="rxslot-enterprise-global-template"
select="$rxslot-enterprise-global-template-14/*"/><!-- end slot nav_left
-->
          </div>
          <div class="leftTables">
            <span class="headergreen">Markets</span>
            <!-- begin XSL -->
                              <xsl:comment>#include
file=&quot;adovbs.inc&quot;</xsl:comment>
                              <!-- end XSL -->
                               <!--


<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;DJIA</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">8,022.02</td>

<td class="bodyblacksmall" align="right" valign="middle" height="10"
width="50">+108.81</td>

</tr>

</table>

</td>

</tr>

<tr>

<td width="53">

<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;NASDAQ</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">1,327.99</td>

<td class="bodyblacksmall" align="right" valign="middle" height="10"
width="50">+14.16</td>

</tr>

</table>

</td>

</tr>

<tr>
```

```
<td width="53">

<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;S&amp;P500 </td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">850.17</td>

<td class="bodyblacksmall" align="right" valign="middle" height="10"
width="50">+11.24 </td>

</tr>

</table>

</td>

</tr>

<tr>

<td width="53">

<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;RJQ</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">45.09</td>

<td class="bodyredsmall" align="right" valign="middle" height="10"
width="50">-1.18</td>

</tr>

</table>

</td>

</tr>

<tr>

<td width="53">
```

```
<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;WKM</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">13.16</td>

<td class="bodyblacksmall" align="right" valign="middle" height="10"
width="50">+4.98</td>

</tr>

</table>

</td>

</tr>

<tr>

<td width="53">

<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>

<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;YTB</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">23.73</td>

<td class="bodyredsmall" align="right" valign="middle" height="10"
width="50">-18.71</td>

</tr>

</table>

</td>

</tr>

<tr>

<td width="53">

<table width="160" border="0" cellspacing="0" cellpadding="0"
class="bodyblacksmall">

<tr>
```

```
<td align="left" valign="middle" class="bodyblacksmall" width="50"
height="10">&nbsp;TWUR</td>

<td align="left" valign="middle" width="50" class="bodyblacksmall"
height="10">56.27</td>

<td class="bodyredsmall" align="right" valign="middle" height="10"
width="50">-1.01</td>

</tr>

</table>

</td>

</tr>

</table>

-->
```

Configuring Your Web Application Server to Connect to Your Database

Check the documentation provided with your Web application server and the database containing your dynamic data for information about connection requirements for the type of dynamic page that you are using.

We use the IIS Web application server and a SQL Server database. One way to connect IIS to the database is to use an ASP page that begins with the following connection information:

```
set con = server.createobject("ADODB.Connection")
set cmd = Server.CreateObject("ADODB.Command")

con.open "PROVIDER=SQLOLEDB;DATA
SOURCE=SQLSERVERNAME\SQLSERVERINSTANCE;UID=sa;PWD=mypassword;DATABASE=db
name"
```

We include this information at the top of the Inetpub\wwwroot\EI_Home\adovbs.inc file.

NOTE: Depending on the way your database server is defined, you may be able to set SOURCE in the above configuration to *localhost.* However, the actual SQL server name and instance may be necessary to avoid errors stating that the page cannot be displayed or the connection is not valid.

Setting up the Dynamic Preview Site

Your dynamic preview Site may be located on your publishing Web Server or on another Web Server that can serve your dynamic content. In our example, the Variants are assembled as .asp files and are published to an IIS Web application server. In our IIS publishing root, we create a subfolder named EI_Home which will hold the ASP files that we publish during the preview process.

To set up the dynamic preview Site:

1. In the Web application server's document root, create the folder where you want to publish your content for dynamic preview. In our example, we publish to the IIS Web Server which has the document root: Inetpub/wwwroot. We add the subfolder EI_Home: Inetpub/wwwroot/EI_Home. We will publish the dynamic content that we want to preview to the EI_Home folder.

   Check your particular Web application server's documentation for additional steps you must take when creating the subfolder.

NOTE: Since our output document looks for certain static files associated with the preview Site under the name of the folder where they are stored on our publishing site, we give our dynamic previewing sub-folder the same name as our publishing sub-folder, EI_Home.

2. Copy the static resources, including your global templates and managed navigation components into the locations in the Web application server where your preview document will look for them. You may omit any static images if it is unimportant for them to appear in the dynamic preview. In our example we include the following:

- We copy the rx_resources folder and the enterprise_investments portion of the *web_resources* folder from our Rhythmyx root to the *EI_Home* Folder. Along with static Rhythmyx files, the *rx_resources* file includes our global template xsl files. The *web_resources* folder contains Site design elements. We rename *rx_resources* to *resources* to match the folder specified in the publisher variables for our publishing context.
- We also insert folders containing static images and navigation graphics into the *EI_Home* Folder; normally these files would be published to the Site during a full publish, and in that manner become available as static resources. We place the files in the locations where our published preview items look for them (in our example, our published preview items look for them in the *EI_Home* folder and the sub-folders they would be published to during publishing: *About Enterprise Investments*, *Files*, *Images*, *InvestmentAdvice*, *MortgagesAndHomeFinance*, and *ProductsAndServices.* We generated these folders and graphics by doing a full publish to the publishing site root, and then copied the folders into the dynamic preview site root).
- Notice that the adovbs.inc file is also included in the *EI_Home* Folder.

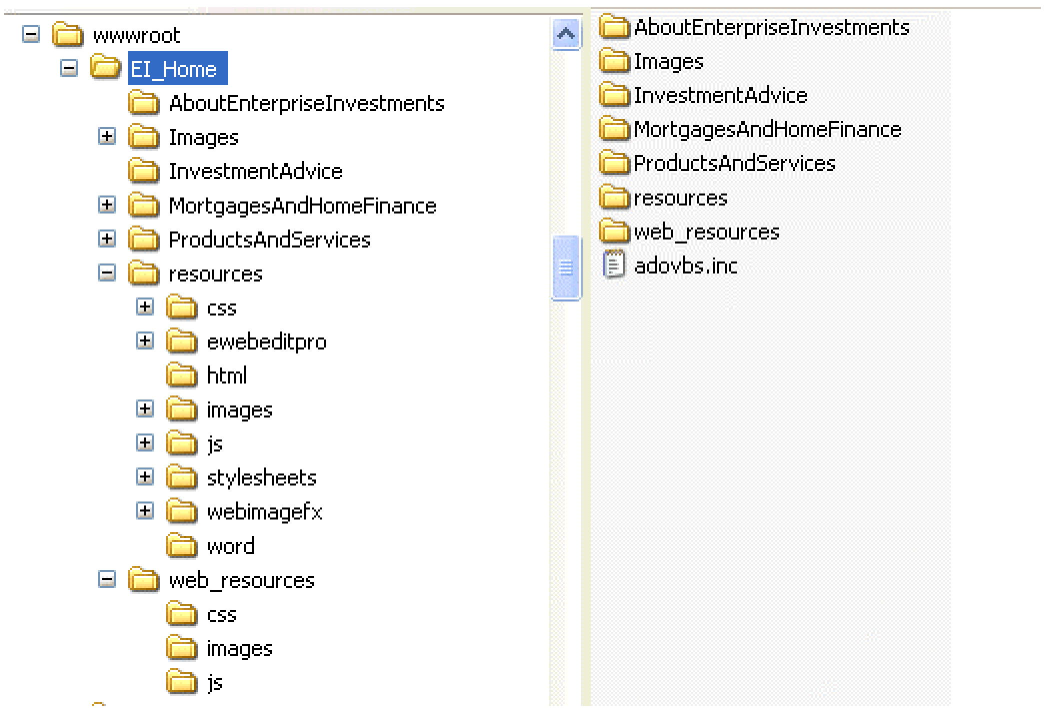
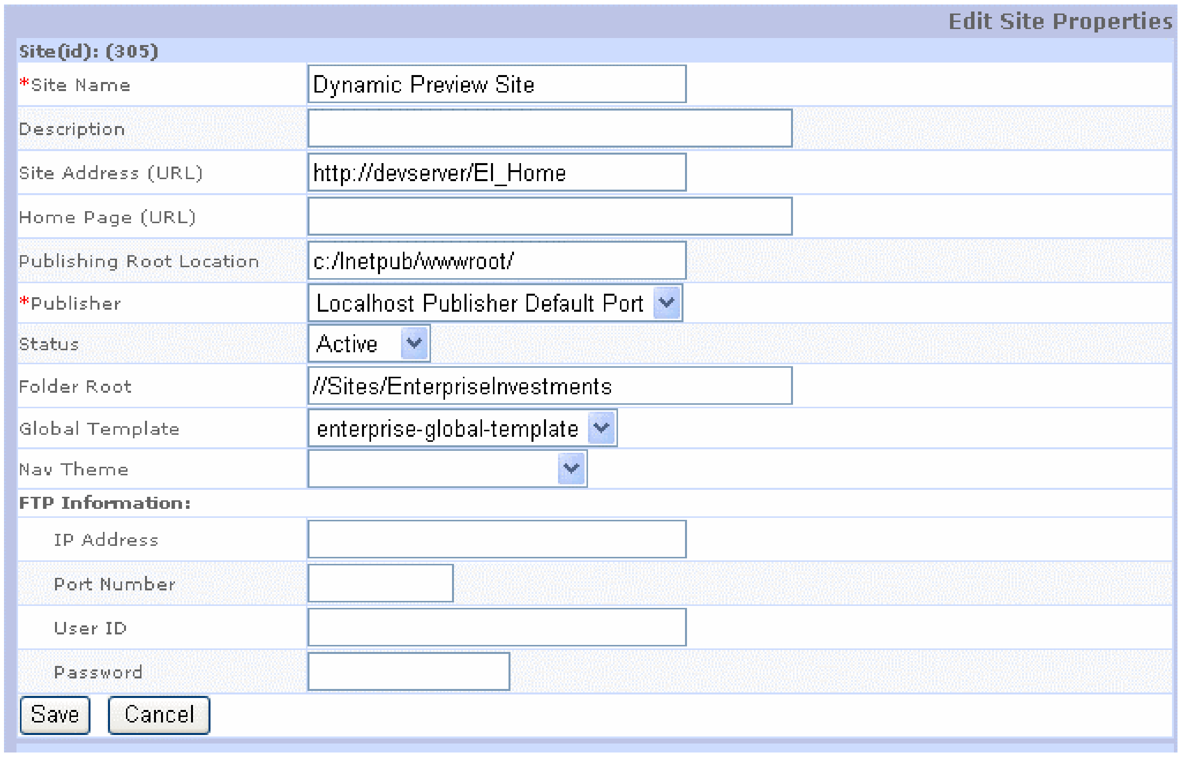


Figure 31: Dyanmic Preview Site

1 Register the dynamic preview Site in Rhythmyx. In the Publishing tab, open the Sites section and click New Site.



Edit Site Properties

Site(id): (305)

*Site Name	Dynamic Preview Site
Description	
Site Address (URL)	http://devserver/EI_Home
Home Page (URL)	
Publishing Root Location	c:/Inetpub/wwwroot/
*Publisher	Localhost Publisher Default Port
Status	Active
Folder Root	//Sites/EnterpriseInvestments
Global Template	enterprise-global-template
Nav Theme	
FTP Information:	
IP Address	
Port Number	
User ID	
Password	

Figure 32: Site registration for dynamic preview Site

a) Enter a **Site Name**.

b) In **Site Address**, enter the URL as you would enter it in your browser address line. (In IIS we have mapped the http://devserver virtual path to the physical path C:\Inetpub\wwwroot).

c) In **Publishing Root Location**, enter the absolute value of your Web application server's publishing directory. In our example, we enter the default publishing root for IIS, C:/Inetpub/wwwroot.

d) In **Publisher**, choose the default publisher or a publisher that you have defined for publishing for dynamic preview.

e) In **Folder Root** we enter the Content Explorer Folder Root that contains our managed navigation components, *//Sites/EnterpriseInvestments*. If your output pages use managed navigation, you must enter the **Folder Root**. It is the same as the **Folder Root** that you would enter if you were performing Site Folder Publishing.

f) In **Global Template**, choose the global template that you want to use for the dynamic preview. In our example, we choose the global template for the FastForward Enterprise Investments Site, enterprise-global-template.

Registering a Dynamic Preview Context and Location Scheme

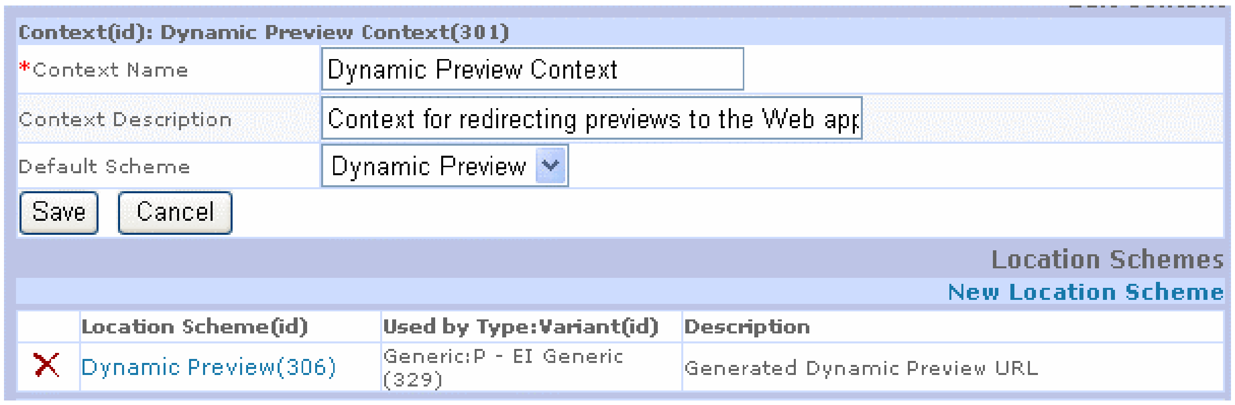
When you preview dynamic content, you use your default publish Context and a dynamic preview context that you must register. Location Schemes registered by the generic publish Context are not used when Previewing dynamic content.

Rhythmyx uses the publish Context when it assembles the output. The dynamic preview Context generates the URL to which the Preview Action redirects publication of the file.

Add to the dynamic preview context a Location Scheme that specifies the address of the Web application Server and the filenames given to the preview Variants when they are published.

To register the dynamic preview Context:

1 Register a new Context for dynamic preview only.



Context(id): Dynamic Preview Context(301)

*Context Name	Dynamic Preview Context
Context Description	Context for redirecting previews to the Web apr
Default Scheme	Dynamic Preview

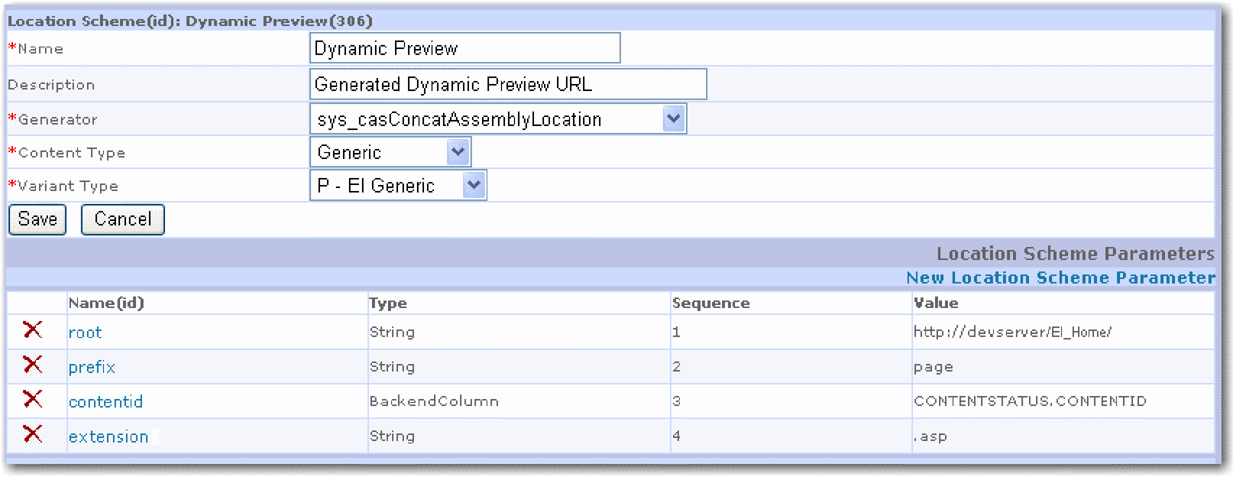
	Location Scheme(id)	Used by Type:Variant(id)	Description
✗	Dynamic Preview(306)	Generic:P - EI Generic (329)	Generated Dynamic Preview URL

Figure 33: Dynamic Preview Context

To register a dynamic Preview Location Scheme:

1. Enter a **Name** and optionally, a **Description** for the Location Scheme.
2. Choose the **Generator** that you want to use to generate the Location Scheme. Our example uses sys_casConcatAssemblyLocation, which concatenates the Location Scheme Parameters in the order indicated.
3. Set **Content Type** to the Content Type that you will be previewing dynamically and set **Variant Type** to the assembly Variant.
4. Assign the absolute **Value** of the Web application Server to the first **Location Scheme Parameter** in the **Sequence** to prevent Rhythmyx from concatenating the **Publishing Root Location** in front of the Location Scheme. Depending on the way your database server is defined, if you use localhost instead of the server name, your Web application server may be unable to locate the database server.
5. Assign values to the other parameters as you would for any Location Scheme. In our example, we use the common pattern of concatenating the word page in front of the contentid.
6. Assign the final parameter to the file type to be previewed on the Web application server. In our example, we hard code the file type as .asp.

NOTE: You could also use the Content Editor sys_suffix field to specify the file extension. In this example, you would set the value with a drop list that would let users choose .asp or .htm depending on the Variant they intend to use for the Content Item.



Location Scheme(id): Dynamic Preview(306)

*Name	Dynamic Preview
Description	Generated Dynamic Preview URL
*Generator	sys_casConcatAssemblyLocation
*Content Type	Generic
*Variant Type	P - EI Generic

Name(id)	Type	Sequence	Value
root	String	1	http://devserver/EI_Home/
prefix	String	2	page
contentid	BackendColumn	3	CONTENTSTATUS.CONTENTID
extension	String	4	.asp

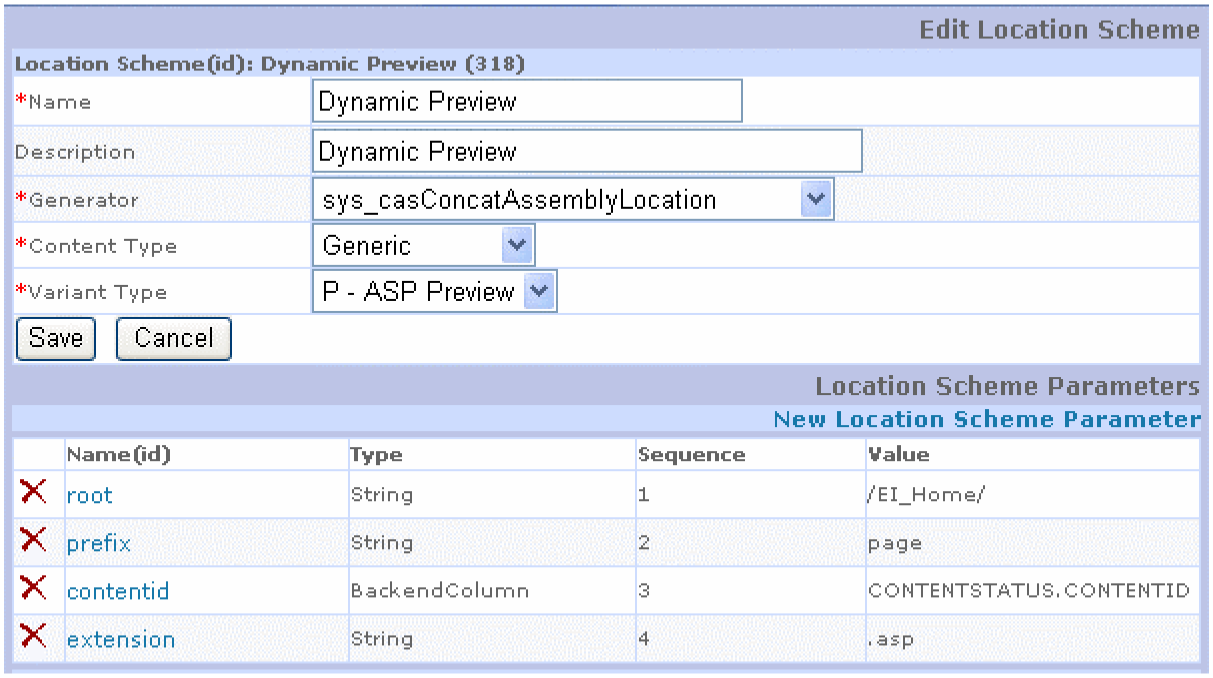
Figure 34: Dynamic Preview Location Scheme in Preview Context

Registering a Dynamic Preview Location Scheme in the Publish Context

If you have installed FastForward, you already have a location scheme for publishing the Generic Content Type in your Publish Context. If this location scheme were publishing the Content to the location where we are performing our dynamic previews, we could simply use this location scheme. However, in FastForward we publish to our AppServer directory, and in this example we dynamically preview on the IIS publish Site. Therefore, we have to add a dynamic preview location scheme in our publish Context that links to the IIS publish Site.

To register the dynamic preview Location Scheme in the Publish Context:

1. Open the Publish Context.
2. Click New Location Scheme.



Location Scheme(id): Dynamic Preview (318)

*Name	Dynamic Preview
Description	Dynamic Preview
*Generator	sys_casConcatAssemblyLocation
*Content Type	Generic
*Variant Type	P - ASP Preview

Name(id)	Type	Sequence	Value
root	String	1	/EI_Home/
prefix	String	2	page
contentid	BackendColumn	3	CONTENTSTATUS.CONTENTID
extension	String	4	.asp

Figure 35: Dynamic Preview Location Scheme in Publish Context

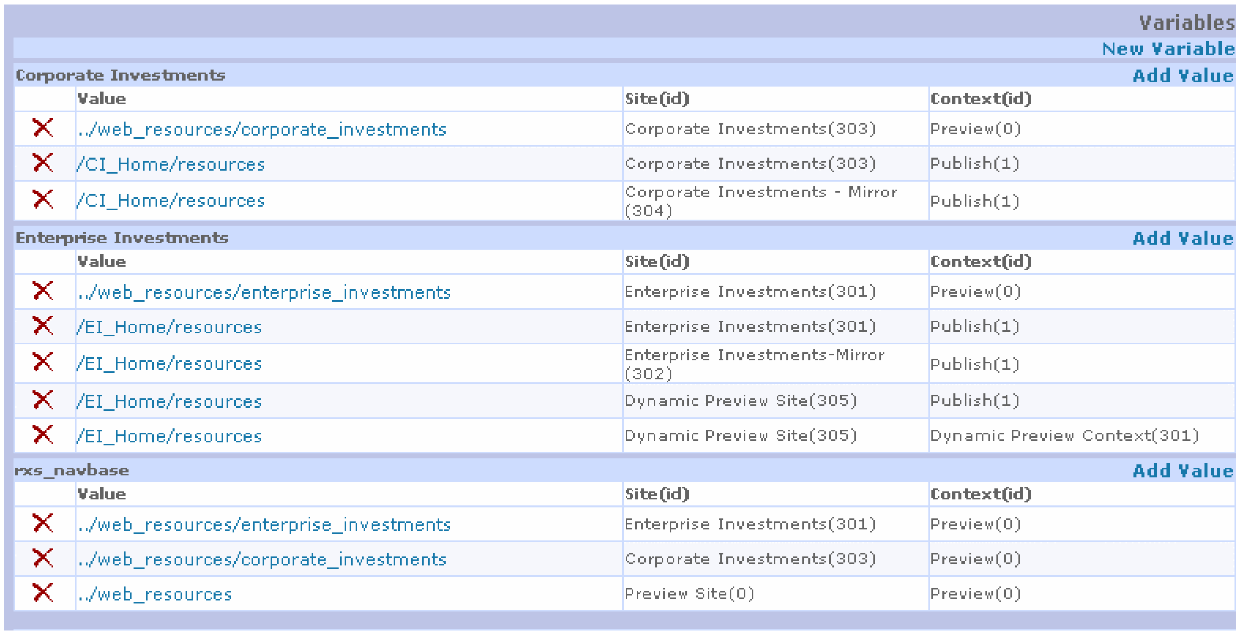
3. Enter a **Name** and optionally, a **Description** for the Location Scheme.
4. Since this Location Scheme is almost identical to the Dynamic Preview Context's Dynamic Preview Location Scheme, as the **Generator**, choose sys_casConcatAssemblyLocation, which concatenates the Location Scheme Parameters in the order indicated.
5. Set Content Type to the Content Type that you will be previewing dynamically and set Variant Type to the preview Variant.
6. Assign the Value of the publish directory in the Web application Server to the first Location Scheme Parameter in the Sequence. Rhythmyx will concatenate the Publishing Root Location in front of the Location Scheme.
7. Assign the same values to the other parameters as those in the Dynamic Preview Context's Dynamic Preview Location Scheme.

Defining Publishing Variable Values for the Preview Site

As in regular Rhythmyx publishing, the locations of some of the static resources are specified in variables. Set the value of publishing variables that apply to the preview Site.

To set the values of publishing variables:

1 In the Publisher tab, open the Variables page, and add values for variables that point to the static content that you have put on the preview Site.



	Value	Site(id)	Context(id)
Corporate Investments			
✗	../web_resources/corporate_investments	Corporate Investments(303)	Preview(0)
✗	/CI_Home/resources	Corporate Investments(303)	Publish(1)
✗	/CI_Home/resources	Corporate Investments - Mirror (304)	Publish(1)
Enterprise Investments			
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
✗	/EI_Home/resources	Enterprise Investments(301)	Publish(1)
✗	/EI_Home/resources	Enterprise Investments-Mirror (302)	Publish(1)
✗	/EI_Home/resources	Dynamic Preview Site(305)	Publish(1)
✗	/EI_Home/resources	Dynamic Preview Site(305)	Dynamic Preview Context(301)
rxs_navbase			
✗	../web_resources/enterprise_investments	Enterprise Investments(301)	Preview(0)
✗	../web_resources/corporate_investments	Corporate Investments(303)	Preview(0)
✗	../web_resources	Preview Site(0)	Preview(0)

Figure 36: Publishing Variables including those for Dynamic Preview

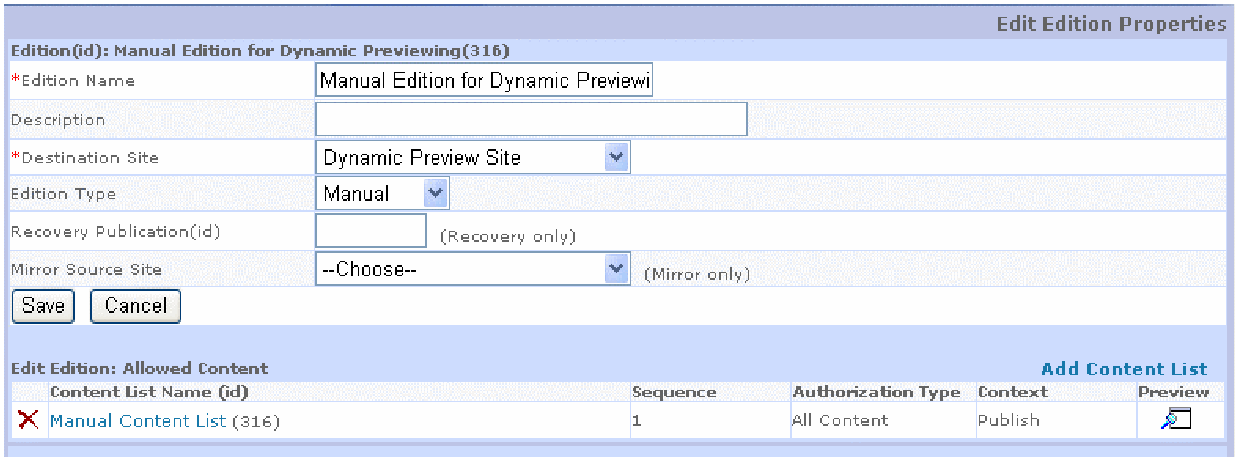
In our example, we add the value /EI_Home/resources to the Enterprise Investments variable for the Dynamic Preview Site/Dynamic Preview Context and the Dynamic Preview Site/Publish Context because our output document looks for static resources in both Contexts.

Note that the static resources stored under wwwroot/EI_Home/web_resources are not located through publishing variables, so the path does not have to be added in this screen.

Setting Up the Dynamic Preview Edition

Dynamic preview Editions are manual Editions that specify the Web application server as a destination site.

Register a manual Edition for publishing dynamic preview Variants that specifies your dynamic preview Site and Content List.


Edit Edition Properties

Edition(id): Manual Edition for Dynamic Previewing(316)

*Edition Name	Manual Edition for Dynamic Previewi
Description	
*Destination Site	Dynamic Preview Site
Edition Type	Manual
Recovery Publication(id)	(Recovery only)
Mirror Source Site	--Choose-- (Mirror only)

Edit Edition: Allowed Content

	Content List Name (id)	Sequence	Authorization Type	Context	Preview
✗	Manual Content List (316)	1	All Content	Publish	

Figure 37: Dynamic Preview Edition

To register the manual

1. Enter an **Edition Name** and optionally a **Description.**
2. In **Destination Site**, choose the Site you registered on your Web application server for dynamic preview.
3. In **Edition Type**, choose Manual.
4. Leave **Recovery** Publication and **Mirror Source** Site blank.
5. Include the Content List that you create for dynamic previewing (we provide the sample *Manual Content List*). Set the Context to *Publish.* Since the Content List is populated by the sys_PublishEditionForPreview exit, you cannot preview the Content List in this screen. Clicking returns an error page.

Creating the Dynamic Preview Variant

For each dynamic preview Variant registered, a corresponding Variant without the dynamic portion always exists because it is required to format the Variant. In our example, our dynamic preview Variant is P – ASP Preview, and the non-dynamic preview (assembly) Variant associated with it is P – EI Generic.

The XSL for the Variant for dynamic preview redirects the file to the URL on the Web application Server. A dynamic preview Variant is supplied in your default implementation in the rxs_DynamicPreview_cas/p_dynamic_preview resource and the corresponding p_aspRedirect.xsl. You must modify some of the values in the resource to match your implementation and associate the XSL with an assembly Variant.

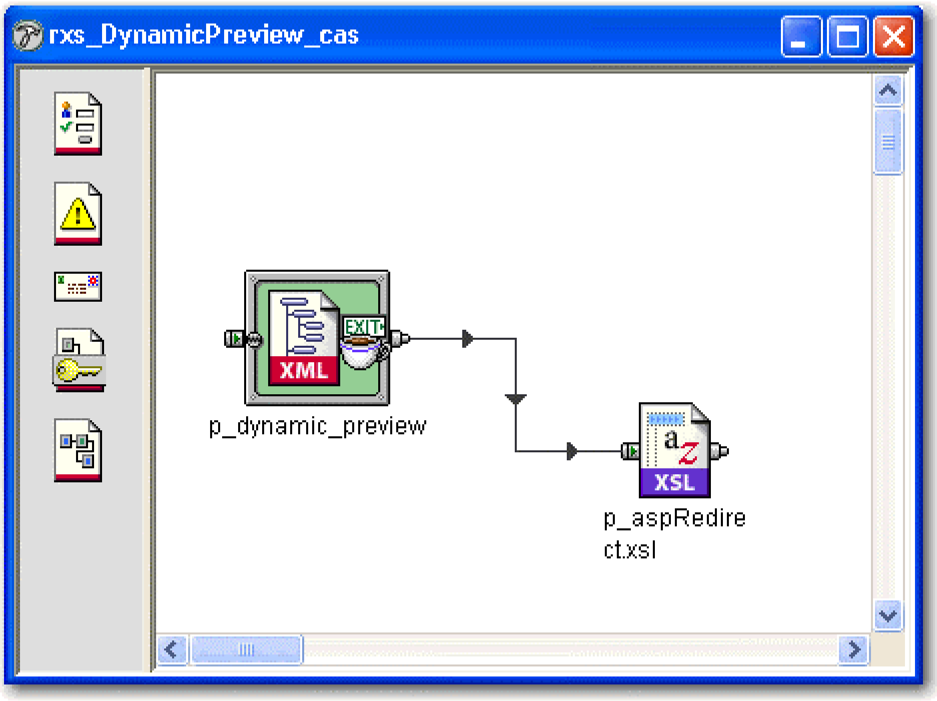
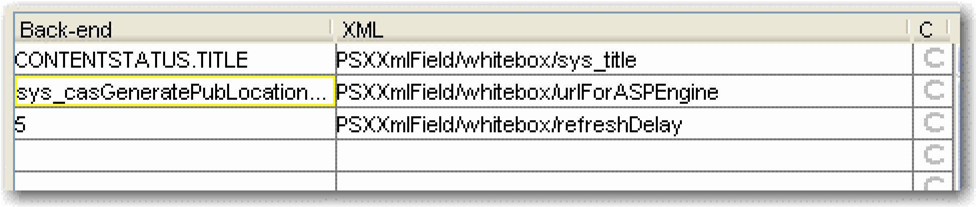


Figure 38: Application for redirecting dynamic previews

To modify the Variant for dynamic preview:

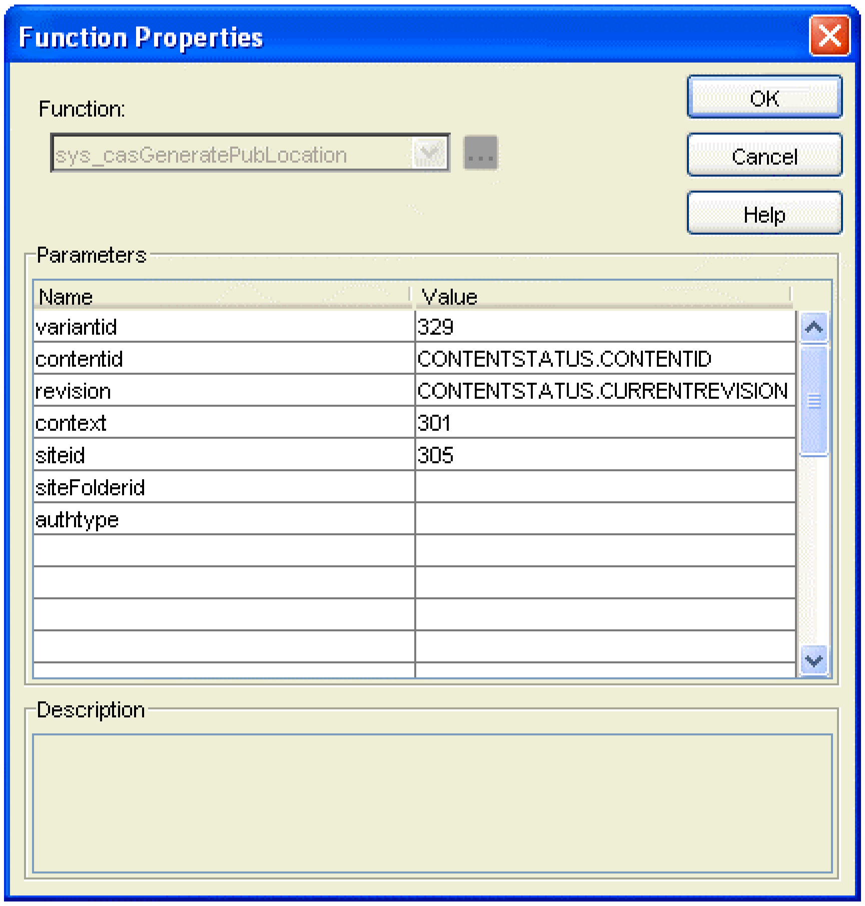
1 In the Workbench, open the mapper for the p_dynamic_preview resource.



Back-end	XML	C
CONTENTSTATUS.TITLE	PSXXmlField/whitebox/sys_title	
sys_casGeneratePubLocation...	PSXXmlField/whitebox/urlForASPEngine	
5	PSXXmlField/whitebox/refreshDelay	

Figure 39: p_dynamic_preview resource mappings

2 The XML field <urlforASPEngine> holds the URL to which the Variant is redirected on the Web application server. The sys_casGeneratePubLocation user-defined function creates the URL. Double-click sys_casGeneratePubLocation to open its properties.



Name	Value
variantid	329
contentid	CONTENTSTATUS.CONTENTID
revision	CONTENTSTATUS.CURRENTREVISION
context	301
siteid	305
siteFolderid	
authtype	

Figure 40: Properties for sys_casGeneratePubLocation

3 Change the value of variantid to the ID of the assembly Variant.

4 Change the value of the context to your dynamic preview Context.

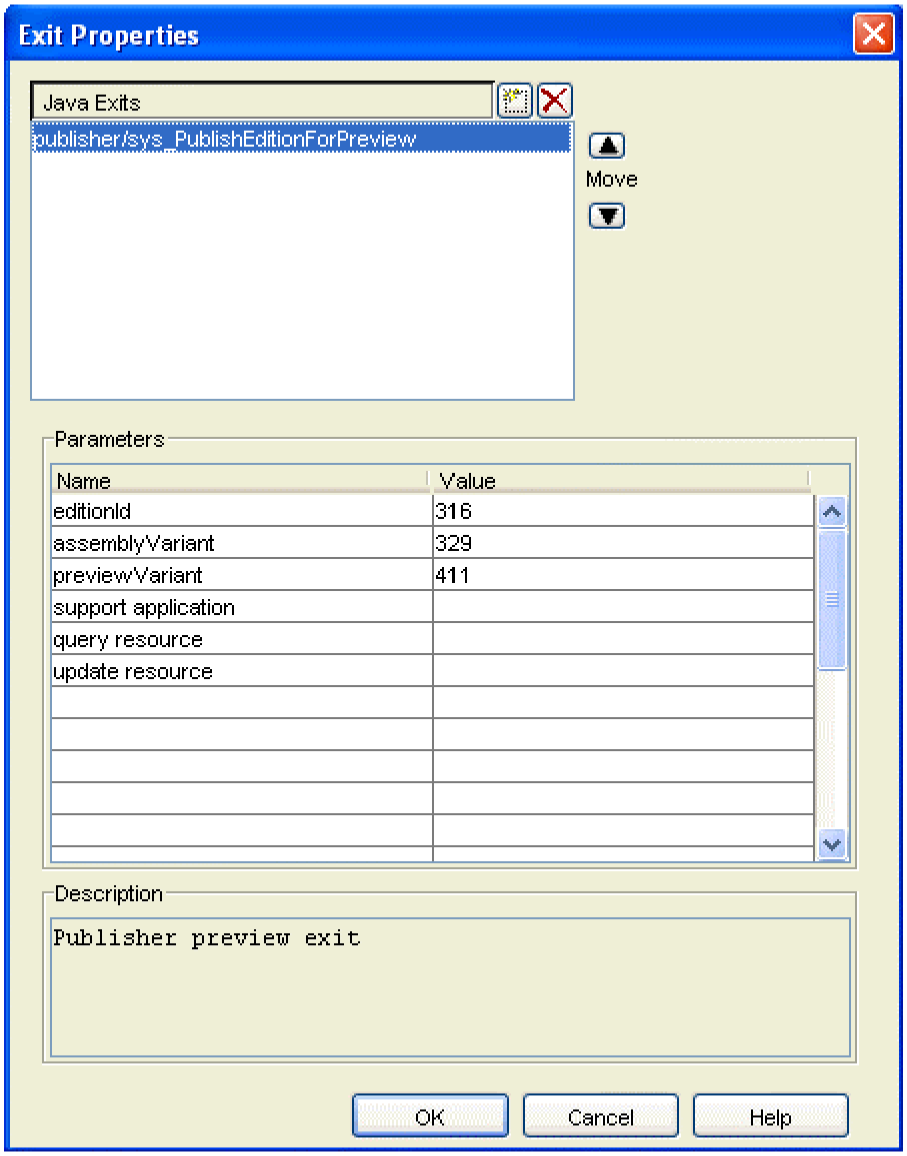
5 Set the siteid equal to your dynamic preview Site ID.

6 Click [**OK**].

7 The XML field <refreshDelay> holds the number of seconds that the Web application server waits before serving the page with the dynamic content. If the delay is too brief, the Web server may attempt to serve the page before it is published. Increase <refreshDelay> if you have encountered this problem.

8 Click [**OK**].

9 On the p_dynamic_preview resource, double-click the post-exit to open it.



publisher/sys_PublishEditionForPreview

Name	Value
editionId	316
assemblyVariant	329
previewVariant	411
support application	
query resource	
update resource	

Publisher preview exit

Figure 41: Values for sys_PublishEditionForPreview

10 In editionID, enter the ID of the manual edition for dynamic previewing.

11 In assemblyVariant, enter the ID of the Variant that formats the content.

12 In previewVariant, enter the ID of the Variant that redirects the preview to the Web application server. (This is the Variant that you edited in the beginning of this section. FastForward provides you with a sample of this Variant for the Generic Content Type named P-ASP Preview.)

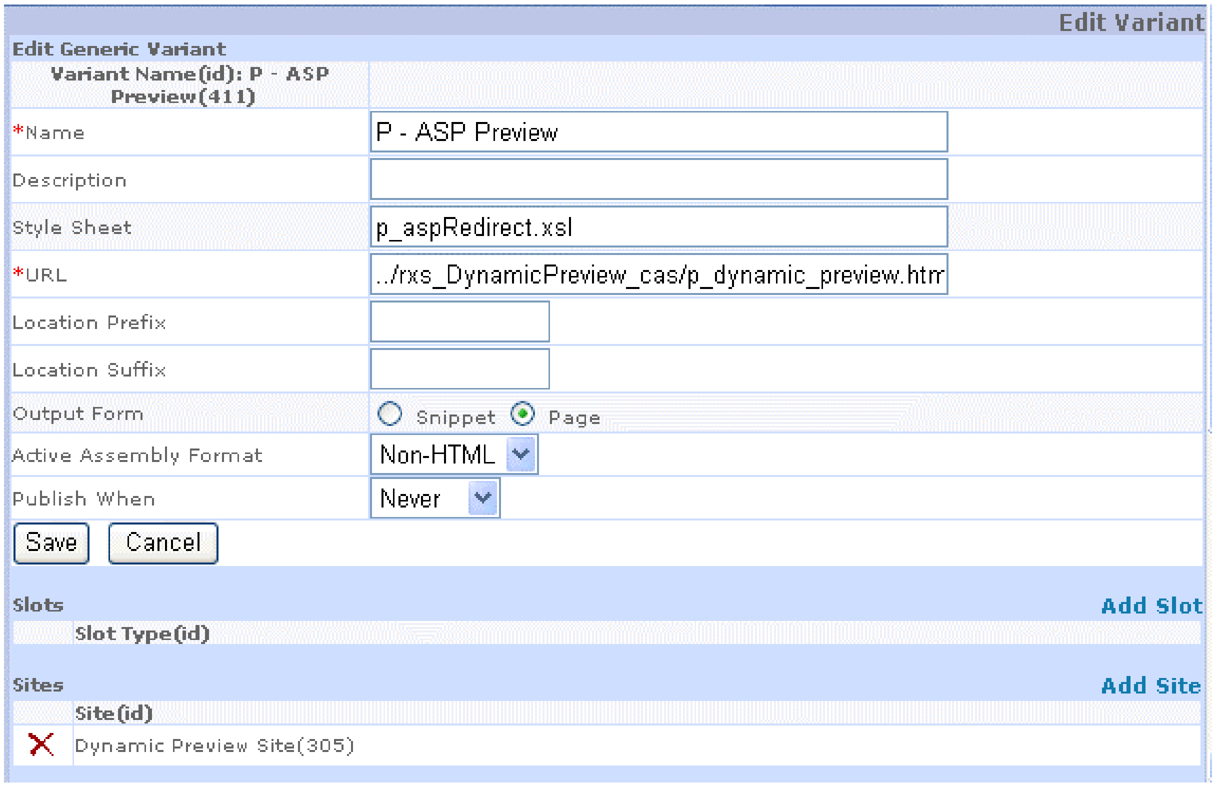
13 The other parameters take default values. See the topic sys_PublishEditionForPreview in the Workbench online help for information about these parameters.

14 Open p-aspRedirect.xsl and change the value of the Variant to the value of your preview Variant.

15 Click [**OK**].

16 Save rxs_DynamicPreview_cas to apply your changes to the current session.

17 In the System tab of Content Explorer open the Variant for dynamic preview, P – ASP Preview. You must add the Site associated with the Variant.



Edit Variant

Edit Generic Variant

Variant Name(id): P - ASP Preview(411)	
*Name	P - ASP Preview
Description	
Style Sheet	p_aspRedirect.xsl
*URL	../rxs_DynamicPreview_cas/p_dynamic_preview.htm
Location Prefix	
Location Suffix	
Output Form	Snippet [✓ Page]
Active Assembly Format	Non-HTML
Publish When	Never

Save Cancel

Slots — Add Slot

Slot Type(id)

Sites — Add Site

Site(id)

Dynamic Preview Site(305)

Figure 42: Registration for Preview Variant

To complete the registration of the Variant for dynamic Preview :

1 If you have changed any of the default names for the application or resources for dynamic preview, change the names in the registration.

2 Do not change **Active Assembly Format** from *Non-HTML*; dynamic preview Variants cannot be actively assembled on the Rhythmyx Server.

3 In **Output Form** choose *Page* so that Rhythmyx can publish the output.

4 Set **Publish When** to *Never*. Otherwise, it may appear as an option for Default Variant in the Content Editor or Site Folder Publishing may publish it..

5 Click Add Site, and associate that Site that you have registered for dynamic preview with the Variant.

By default, the Dynamic Preview Variant is disabled for all Communities because you must complete its configuration before it works properly.

To enable the Dynamic Preview Variant for Communities:

1 In the System tab of Content Explorer, go to the Communities page.

2 Click the name of the Community that you want to give access to the Dynamic Preview Variant.

3 Click Variants.

4 Click Add Variant.

5 Check the Dynamic Preview Variant.

6 Click [**Save**].

The Dynamic Preview Variant is now included in the list of Action Menu preview options for Content Items of the associated Content Type. A corresponding Variant without the dynamic portion always exists because it is required to format the Variant. In our example, the non-dynamic preview Variant associated with P – ASP Preview is P – EI Generic.

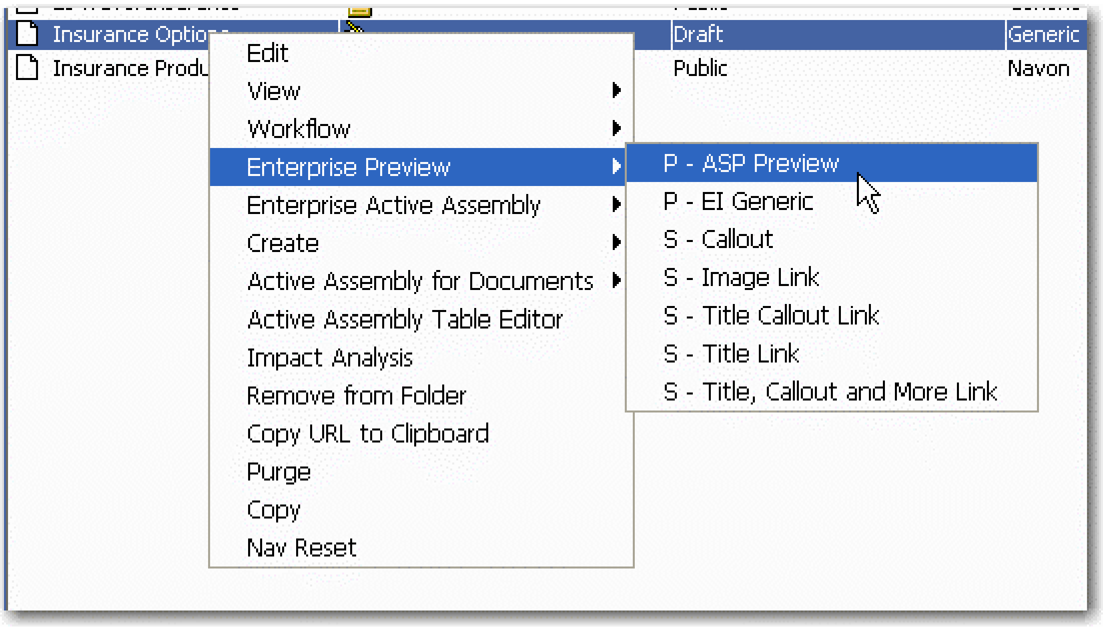


Figure 43: Dynamic Preview Variant in Preview menu

Setting Up the Dynamic Preview Content List

Dynamic preview Content Lists are associated with Manual Editions so that only the Content Item chosen for preview is published.

A default application and registration for the dynamic preview content List are included in Rhythmyx. The application is rx_pubPreviewEdition. It contains the resource contentlist_manual which generates the Content List containing the Content Item to be previewed. It also contains the resources queryEdition and updateEdition. These resources update the RXEDITIONITEMS table with the Content Item to be previewed and delete it from the table after it is previewed.

The resources in the rx_PubPreviewEdition application perform the following functions:

- updateEdition updates the RXEDITIONITEMS table with the Content Item to preview;
- queryEdition queries the RXEDITIONITEMS table to access the Content Item to preview;

- contentlist_manual generates the Content List for the manual Edition and select which Edition to publish;
- updateEdition deletes the Content Item from the RXEDITIONS table after the user has previewed the Content Item.

The sys_PublishEditionForPreview exit in the assembly resource uses the update resource to write the row for the Edition ID in RXEDITIONITEM, and then to delete it after the Publisher delivers the content. It also uses the query resource to retrieve the row for the preview item and publish it to the preview server.

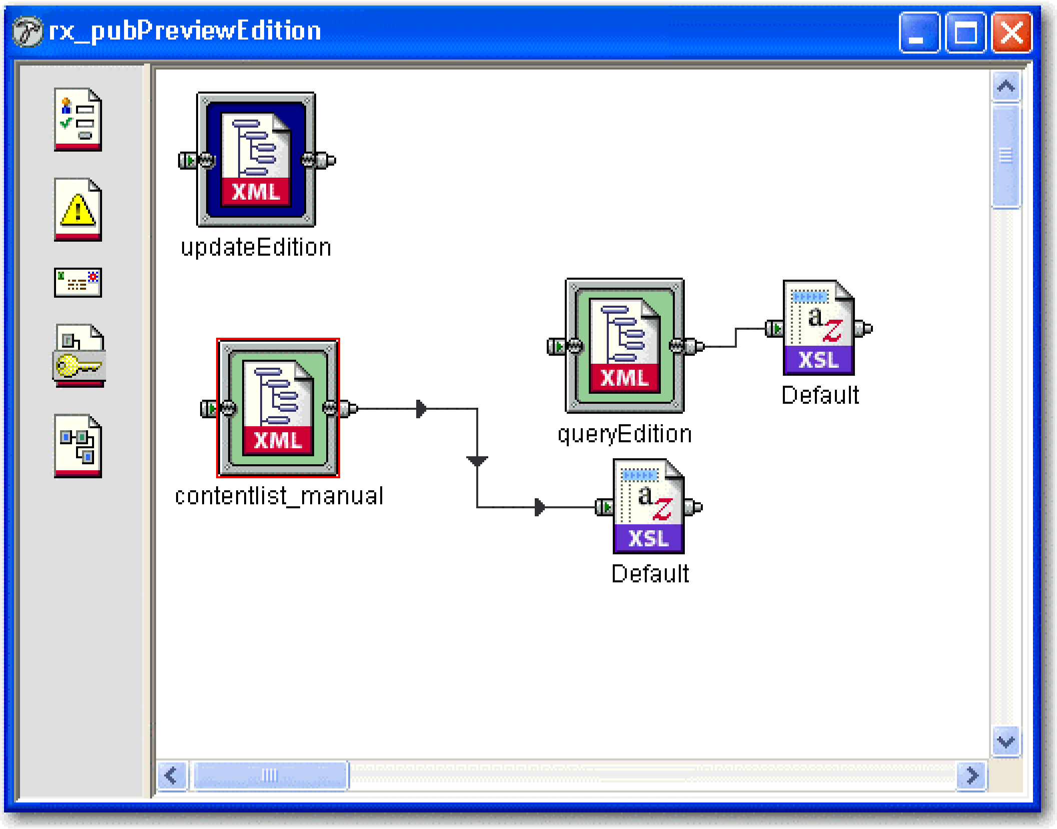
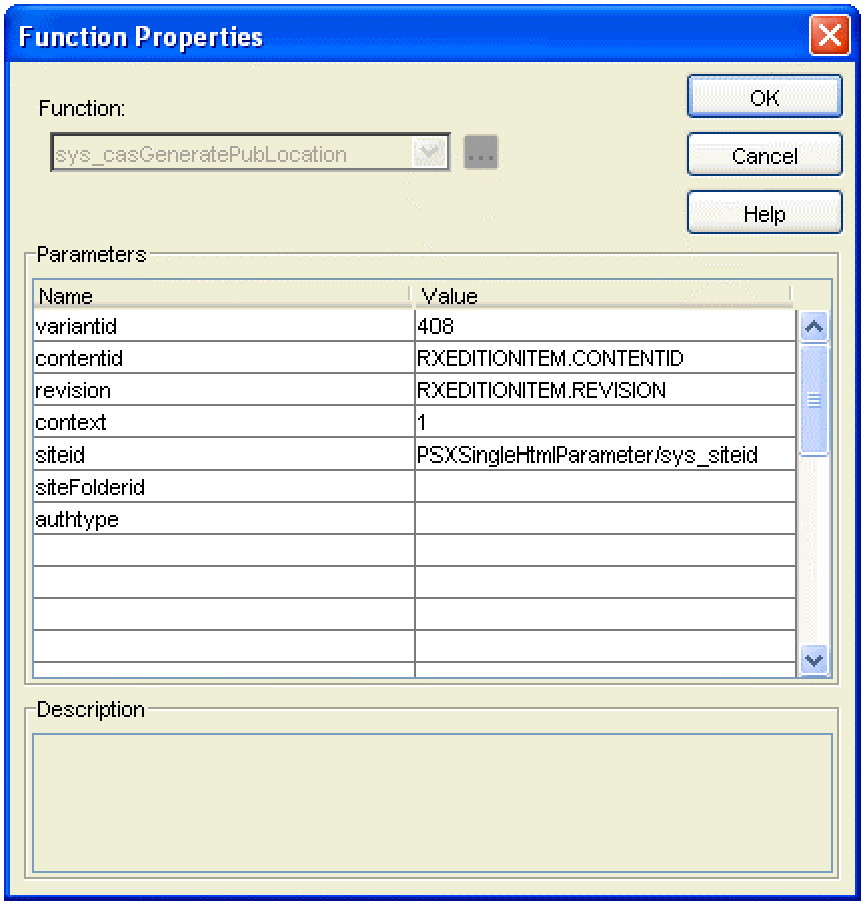


Figure 44: rx_PubPreviewEdition

To map rx_pubPreviewEdition to your own values for Variant, Context, or Site:

1 Open the mapper in contentlist_manual and double-click on the function sys_casGeneratePubLocation to open it.



Function: sys_casGeneratePubLocation

Name	Value
variantid	408
contentid	RXEDITIONITEM.CONTENTID
revision	RXEDITIONITEM.REVISION
context	1
siteid	PSXSingleHtmlParameter/sys_siteid
siteFolderid	
authtype	

Figure 45: sys_casGeneratePubLocation values

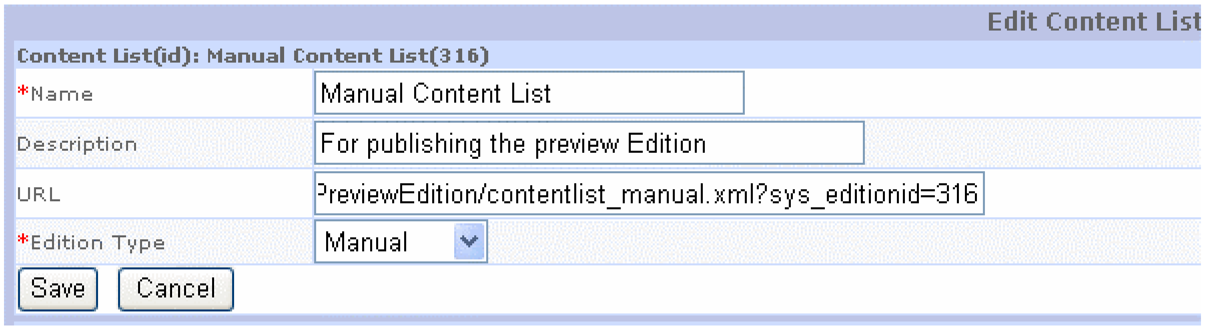
2 Set varaintid equal to your preview Variant ID.

3 Set Context to your publish Context (1 by default).

4 Set siteid equal to PSXSingleHtmlParameter/sys_siteid.

5 Close the mapper and save the application.

The registration for the Content List for dynamic publishing is also included by default. Note that the Edition ID is passed as a parameter through the URL. The Edition ID is included because the Publisher uses the Edition ID to read the RXEDITIONITEM table to get the Content Item to be published.



Content List(id): Manual Content List(316)

*Name	Manual Content List
Description	For publishing the preview Edition
URL	PreviewEdition/contentlist_manual.xml?sys_editionid=316
*Edition Type	Manual

Figure 46: Dynamic Preview Content List

To customize the Content List for dynamic previewing:

1 In **URL**, change sys_editionid to match the id of the edition that you are publishing.

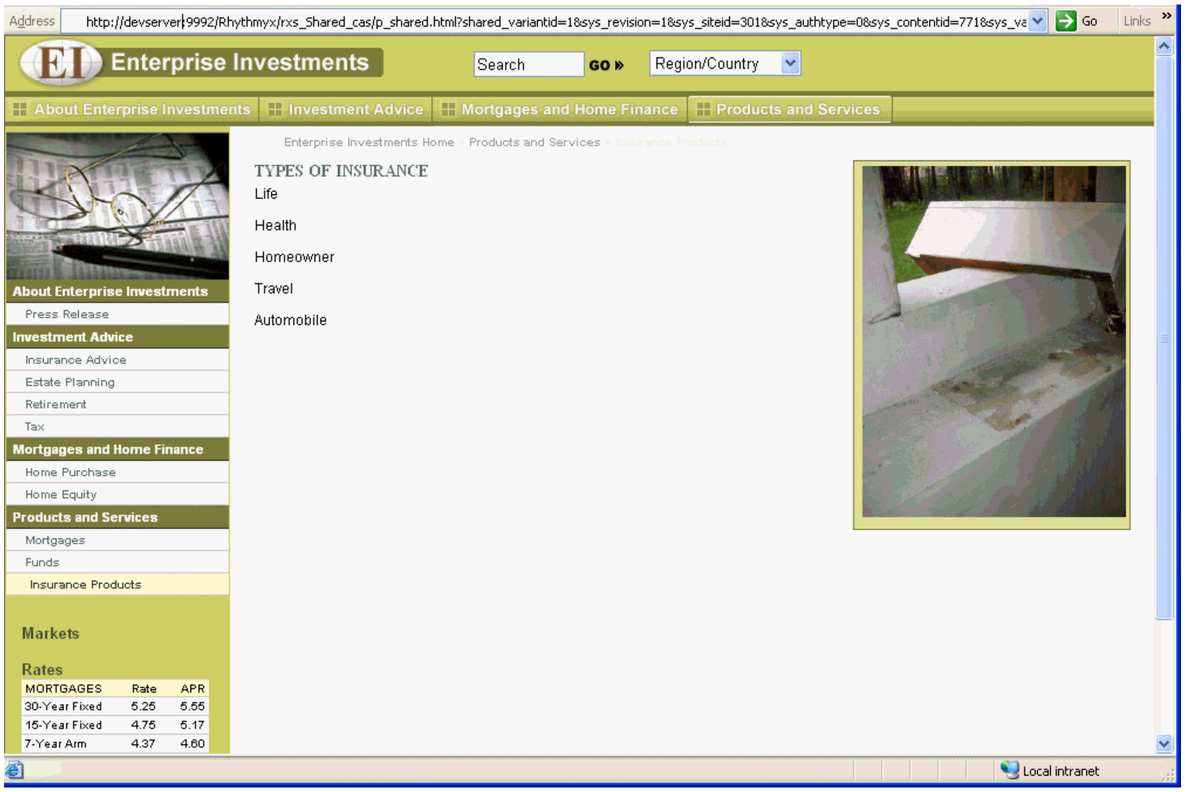
Testing the Dynamic Preview

To test the dynamic Preview Variant, view the non-dynamic version of the Variant, and then compare it to the dynamic preview version. View the dynamic preview version a few times to make sure that the publishing mechanism is working as intended. The first time you preview it, it is actually published to the Site; after that, the publishing mechanism should detect it on the Site and Rhythmyx should display the preview to you without attempting to publish it again (unless you modify it).

1 Create a new Content Item of the Content Type associated with the dynamic preview Variant.

2 Right-click on the Content Item to open the Action Menu and preview the non-dynamic version of the Variant. Since our example is taken from FastForward, the preview action is called Enterprise Preview. To preview the non-dynamic version of the Variant, we choose Entepise Preview > P – EI Generic. The non-dynamic version of the Variant opens. Notice that:

- The URL indicates that it is generated from the Rhythmyx server;
- Nothing appears beneath the Markets header in the left panel because the .asp code is not included (and cannot be read on the Rhythmyx server);
- Through active assembly, a graphic is included in the upper right corner of the page.


Address http://devserver:9992/Rhythmyx/rxs_Shared_cas/p_shared.html?shared_variantid=1&sys_revision=1&sys_siteid=301&sys_authtype=0&sys_contentid=771&sys_va

Enterprise Investments

Search GO » Region/Country

About Enterprise Investments | Investment Advice | Mortgages and Home Finance | Products and Services

Enterprise Investments Home > Products and Services > Insurance Products

TYPES OF INSURANCE

Life

Health

Homeowner

Travel

Automobile

About Enterprise Investments
- Press Release

Investment Advice
- Insurance Advice
- Estate Planning
- Retirement
- Tax

Mortgages and Home Finance
- Home Purchase
- Home Equity

Products and Services
- Mortgages
- Funds
- Insurance Products

Markets

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

Figure 47: Assembly Variant

1 Close the page.

2 Make sure that your Publishing application server and your Web application server are both running.

3 Preview the Content Item through the dynamic preview Variant. In our example, we choose *Enterprise Preview > P – ASP Preview*.

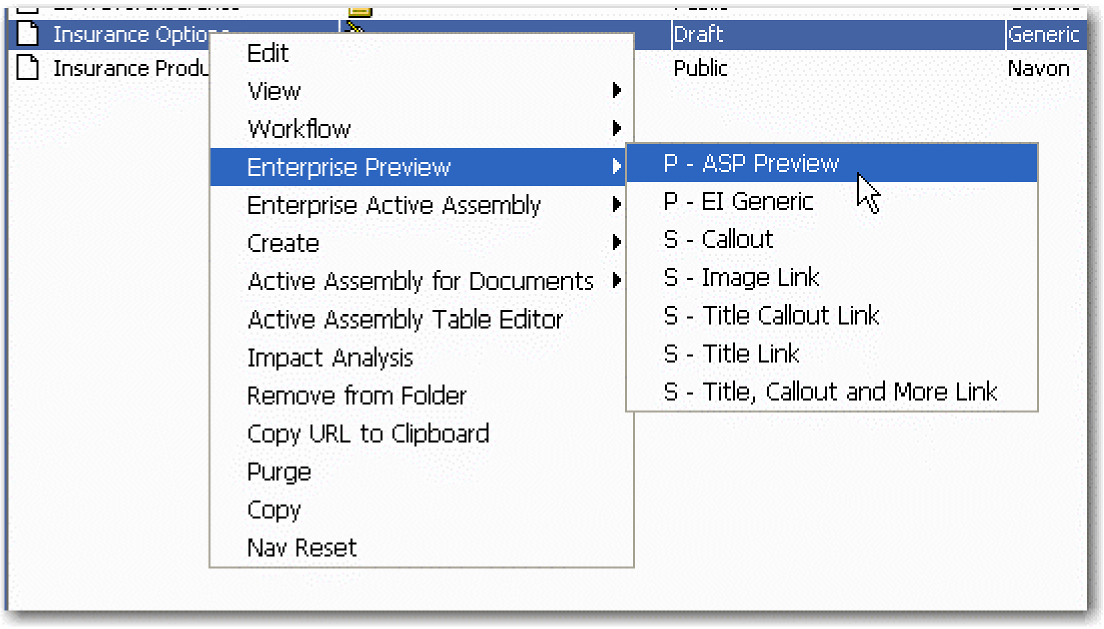
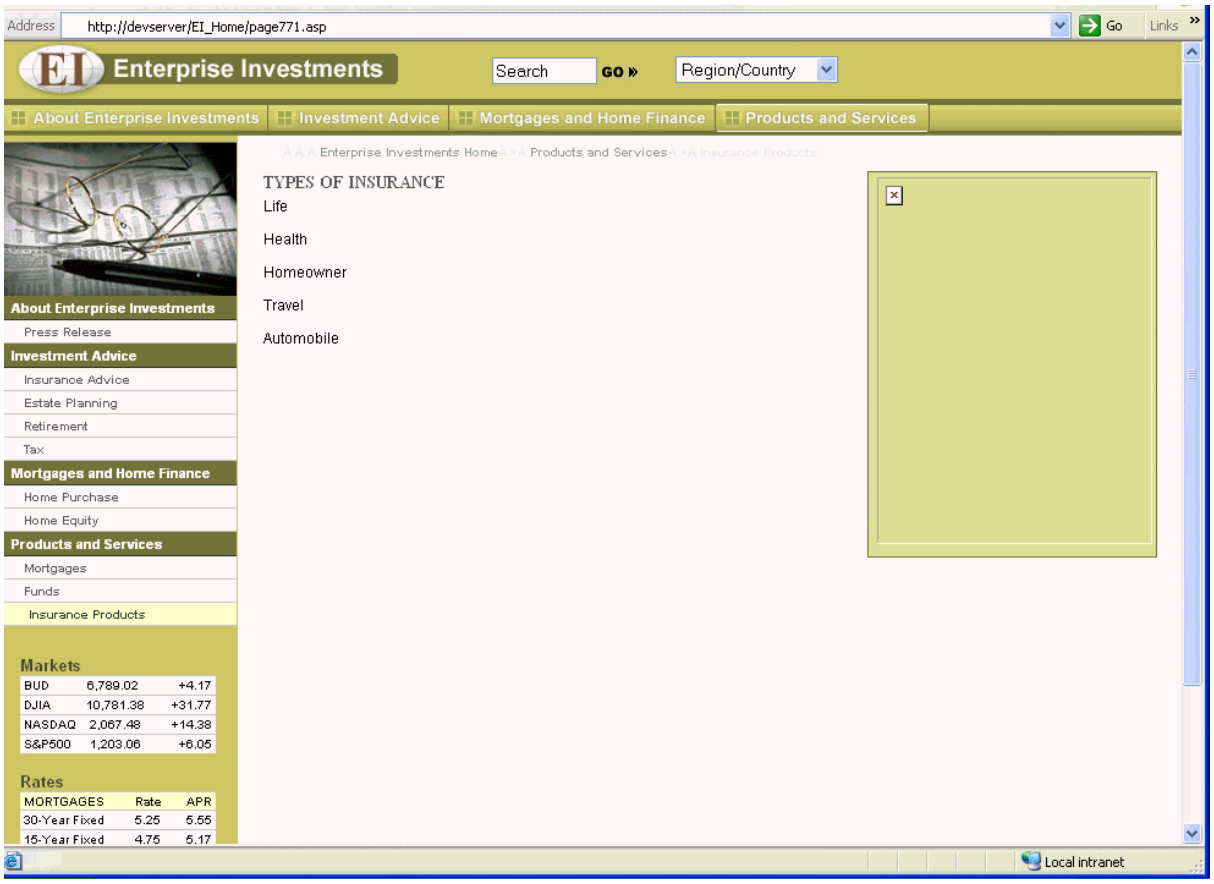


Figure 48: Dynamic Preview Variant selected

The dynamic version of the Variant opens. Notice that:

- The URL indicates that the page is served from the Web application server.
- The table of dynamic content now appears under the Markets header in the left panel.
- The image included through active assembly does not appear because it has not been published to the preview Site. (You can make sure content included through active assembly is included by publishing it to the preview Site.)



TYPES OF INSURANCE

Life

Health

Homeowner

Travel

Automobile

Markets

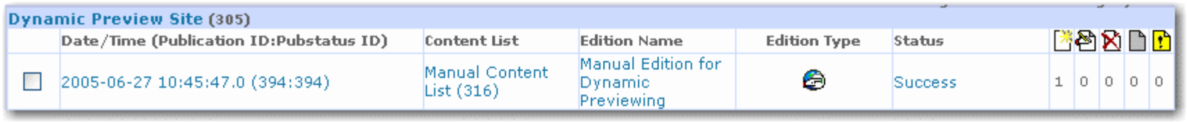
BUD	6,789.02	+4.17
DJIA	10,781.38	+31.77
NASDAQ	2,067.48	+14.38
S&P500	1,203.06	+6.05

Rates

MORTGAGES	Rate	APR
30-Year Fixed	5.25	5.55
15-Year Fixed	4.75	5.17

Figure 49: Dynamic Preview Variant

1 In Content Explorer, on the Publishing tab, open the Publication log. Verify that your Edition has published successfully and that one Content Item has been added.



Dynamic Preview Site (305)

	Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status					
☐	2005-06-27 10:45:47.0 (394:394)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Success	1	0	0	0	0

Figure 50: Publishing Log for Dynamic Previewing

Either click on the Date/Time link or go to the Preview Site in Windows Explorer to verify that the published Content Item is in the expected location:

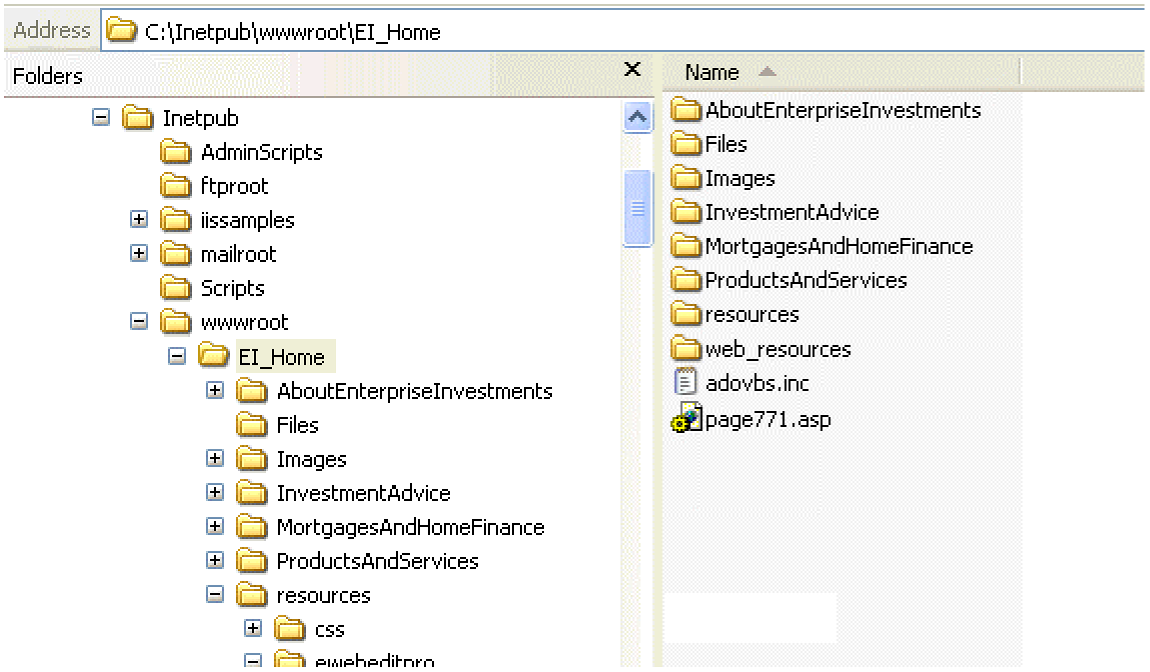
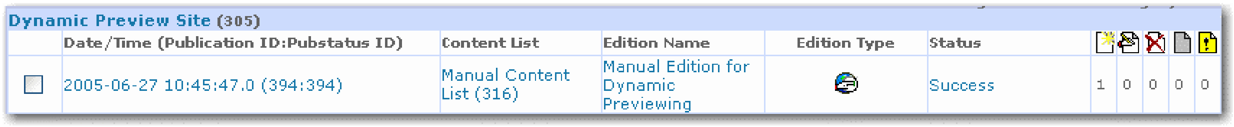


Figure 51: Dynamic Preview Site after item has been previewed

2 Preview the Content Item through the dynamic preview Variant again. The same page should open. Check the Publishing log. The Edition should not have been published again since the Publisher should have detected that the Content Item already exists on the Site and has not been modified.

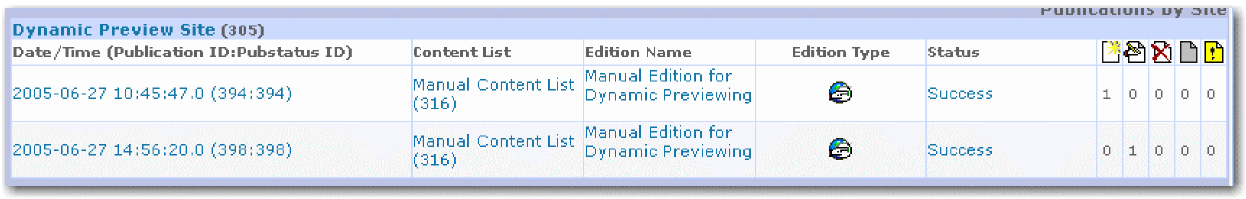


Dynamic Preview Site (305)

	Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status					
☐	2005-06-27 10:45:47.0 (394:394)	Manual Content List (316)	Manual Edition for Dynamic Previewing	🌐	Success	1	0	0	0	0

Figure 52: Publishing Log for Dynamic Previewing

3 Edit the Content Item and make a change in any field. Then preview it through the dynamic preview Variant again. The page that opens should reflect the change if it appears in a field that this Variant displays. Check the Publishing log. The Edition should have been published again, and the log should indicate that one Content Item was updated:



Dynamic Preview Site (305)

Date/Time (Publication ID:Pubstatus ID)	Content List	Edition Name	Edition Type	Status					
2005-06-27 10:45:47.0 (394:394)	Manual Content List (316)	Manual Edition for Dynamic Previewing	🌐	Success	1	0	0	0	0
2005-06-27 14:56:20.0 (398:398)	Manual Content List (316)	Manual Edition for Dynamic Previewing	🌐	Success	0	1	0	0	0

Figure 53: Publisher log after updated item is previewed

Note: If you simply delete the published items from the preview site and attempt to preview them dynamically again, your Web application server returns an error message. You must unpublish the items so that they are removed from all of the Rhythmyx backend tables that record them as published.

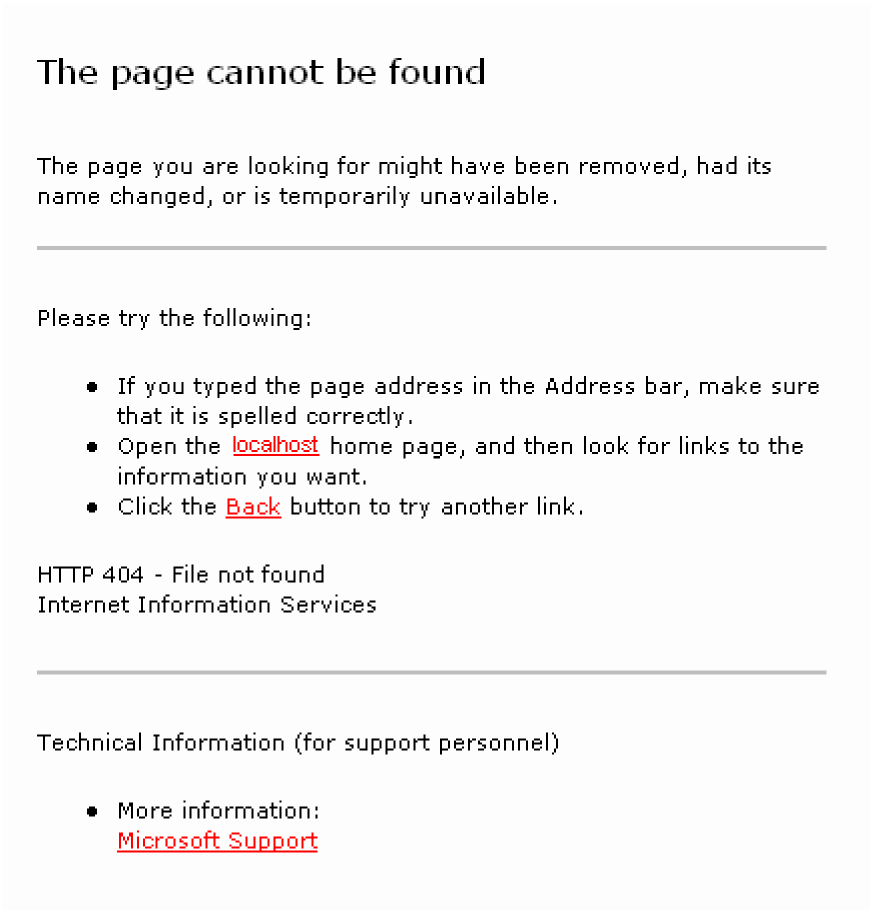
Troubleshooting

Errors occurring during the dynamic preview process are generally due to publishing problems or a failed connection between the Web application server and the external database.

The following topics cover some common errors that may occur.

The page cannot be found

The page cannot be found/HTTP 404 – File not found


The page cannot be found

The page you are looking for might have been removed, had its name changed, or is temporarily unavailable.

Please try the following:

- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- Open the localhost home page, and then look for links to the information you want.
- Click the Back button to try another link.

HTTP 404 - File not found
Internet Information Services

Technical Information (for support personnel)

- More information:
  Microsoft Support

Figure 54: Page Cannot Be Found error

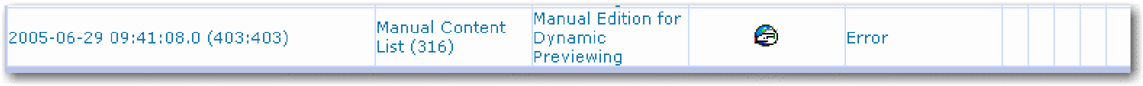
Possible causes and resolutions:

1 The preview file was not published, or was not published to the correct location.

To resolve:

a) In the Publisher tab of Content Explorer, go to the Publisher log.

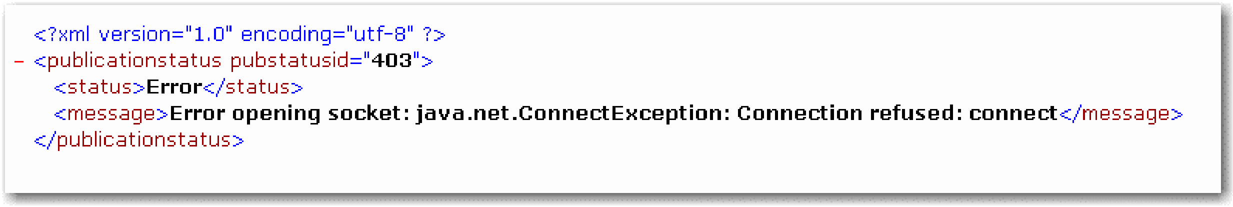
b) If there is an entry for the Edition with a status of Error, click on the word *Error*.



2005-06-29 09:41:08.0 (403:403)	Manual Content List (316)	Manual Edition for Dynamic Previewing		Error

Figure 55: Error Link

If the link opens the following page, you have probably failed to start the publishing application Server (by default, Tomcat). Start the publishing application server, and attempt the preview again.



```
<?xml version="1.0" encoding="utf-8" ?>
- <publicationstatus pubstatusid="403">
    <status>Error</status>
    <message>Error opening socket: java.net.ConnectException: Connection refused: connect</message>
  </publicationstatus>
```

Figure 56: Open socket message

c) If there is an entry for the edition with a status of Success, your redirection URL may not match the location where you published the content.

Look at the Site to determine where the content has published. Then check your Site registration's **Site Address** and **Publishing Root Location** and the dynamic preview Context's Location Scheme to determine where you entered an incorrect path. Correct the path and attempt the preview again.

2 The preview mechanism is attempting to display the content before it is published.

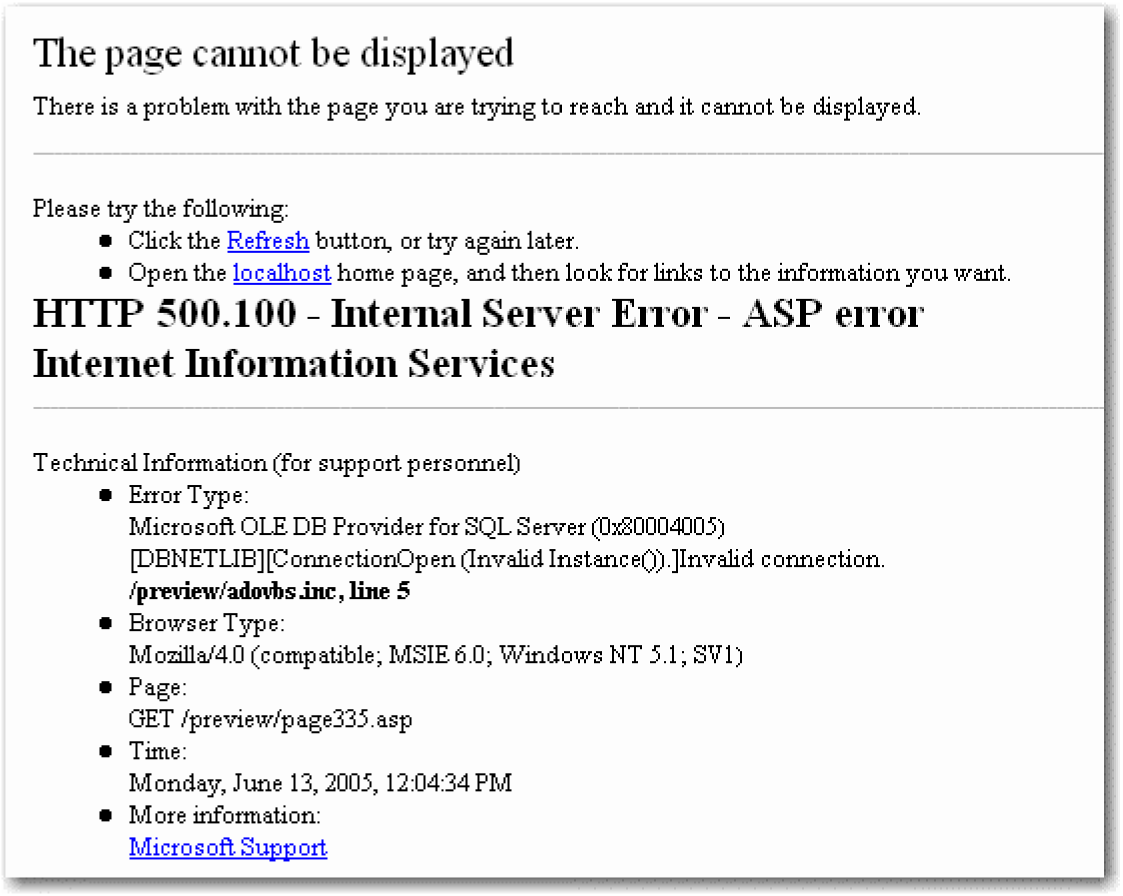
To verify and resolve:

a) Wait a few seconds and check the Publisher log to see if the item published successfully. Then check the Site to see if the Content Item has published to the correct location. If the item published successfully, the server is probably refreshing too quickly.

In the Workbench, open the mapper in rxs_DynamicPreview_cas/p_dynamic_preview. Increase the value of refreshDelay.

The page cannot be displayed

The page cannot be displayed/HTTP 500.100 – Internal Server Error – ASP error/Internet Information Services



The page cannot be displayed

There is a problem with the page you are trying to reach and it cannot be displayed.

Please try the following:

- Click the Refresh button, or try again later.
- Open the localhost home page, and then look for links to the information you want.

HTTP 500.100 - Internal Server Error - ASP error
Internet Information Services

Technical Information (for support personnel)

- Error Type:
  Microsoft OLE DB Provider for SQL Server (0x80004005)
  [DBNETLIB][ConnectionOpen (Invalid Instance()).]Invalid connection.
  /preview/adovbs.inc, line 5
- Browser Type:
  Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
- Page:
  GET /preview/page335.asp
- Time:
  Monday, June 13, 2005, 12:04:34 PM
- More information:
  Microsoft Support

Figure 57: Page cannot be displayed error

This type of error page indicates an IIS error. The lower part of the page gives specific information. In most cases, IIS cannot connect to the server. In this instance, the message under Error Type "(Invalid Instance()).]Invalid connection /preview/adovbs.inc, line 5" indicates that IIS cannot resolve the server name on line 5 of the adovbs.inc file. This may occur if your SQL server is defined as a named instance.

Possible causes and resolutions:

To resolve:

Open the adovbs.inc file and check the connection information in line 5. In this case the connection information appears as:

```
con.open "PROVIDER=SQLOLEDB;DATA
SOURCE=localhost;UID=sa;PWD=password;DATABASE=marketsdb"
```

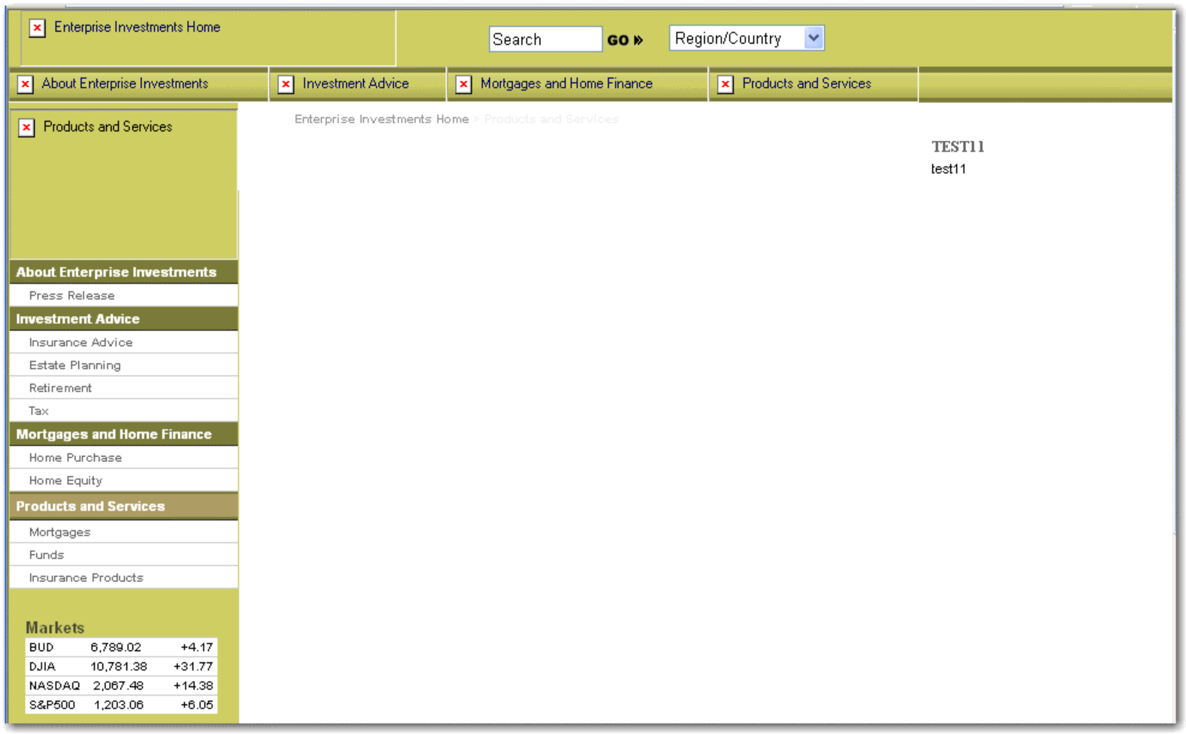
Since IIS could not resolve the server name, we set source to the actual name and instance instead of localhost:

```
con.open "PROVIDER=SQLOLEDB;DATA
SOURCE=MKTSERVER\MKTDATA;UID=sa;PWD=password;DATABASE=marketsdb"
```

See support.microsoft.com for information about similar errors of this type.

Missing static files

Although you have copied the static files to the publishing root on the Web application Server, some of them appear to be missing:



TEST11
test11

BUD	6,789.02	+4.17
DJIA	10,781.38	+31.77
NASDAQ	2,067.48	+14.38
S&P500	1,203.06	+6.05

Figure 58: Dynamic Preview with Missing Images

Possible causes and resolutions:

Navimages are missing because they are placed on a Site when the navigation components are published to the site.

To resolve:

Either display the previews without the navigation images, or manually copy the image files onto the Site, or publish the navigation components to the Site.

For additional errors relating to the publishing process, refer to the Troubleshooting section of Rhythmyx's online publishing documentation or the document *Implementing Publishing in Rhythmyx*.

Index