

# Converting a Rhythmyx Implementation to Use Version 6.5 Features

About Converting to Rhythmyx Version 6.5 Features .....	2
Assumptions .....	2
Reference Server .....	2
Preparing to Implement the Conversion.....	2
Converting Content Lists to Use JCR Queries .....	3
Implementing the Enterprise Investments Full Binary Content List .....	4
Implementing the Enterprise Investments Non-binary Content List.....	6
Implementing the Enterprise Investments Incremental Content List .....	7
Implementing the Enterprise Investments Unpublish Content List.....	7
Updating Editions.....	7
Handling Default Variants.....	9
Converting Location Schemes to Use JEXL .....	9
JEXL Variables Commonly Used in Location Schemes.....	11
JEXL Functions Commonly Used in Location Schemes .....	12
Converting to Velocity Assembly .....	14
Implementing Managed Navigation Templates in Velocity.....	14
Converting Global Templates to Velocity.....	14
Updating Slots .....	18
Converting Local Variants to Velocity Templates .....	18

## About Converting to Rhythmyx Version 6.5 Features

Once you complete the stabilization tasks described in *Upgrading to Rhythmyx Version 6.5*, you can operate your system without further intervention. Rhythmyx Version 6.5 includes three features that offer improvements in performance and maintainability, however, so you may want to consider converting your system to use these features:

- Content Lists based on Java Content Repository (JCR) queries rather than on Rhythmyx XML applications;
- Location Schemes based on Java Expression Languages (JEXL) expressions; and
- Assembly based on Velocity Templates rather than XSLT Stylesheets.

This document explains how to convert your implementation to use these features.

NOTE: To illustrate all procedures in this document, we will assume an upgrade of the standard FastForward installation on Rhythmyx Version 5.7 to Rhythmyx Version 6.5.

## Assumptions

The procedures in this document assume that:

- You have already upgraded your system from Rhythmyx Version 5.7 or earlier and have stabilized the upgraded system. If you have not already upgraded your system, see *Upgrading\_to\_Rhythmyx\_Version\_6\_0.pdf*, the upgrade and stabilize your system before proceeding.
- You publish to one site or a small group of sites.
- You publish HTML or XML.
- You use Site Folder Publishing. If you do not use Site Folder Publishing, you will need to re-implement your system in Rhythmyx Version 6.5 to use that feature rather than converting your existing implementation.

## Reference Server

Percussion Software recommends that you install a separate Rhythmyx Version 6.5 Server with FastForward to serve as a reference implementation. You can refer to the implementation of the Content Lists and Velocity Templates in the reference server as an example implementation of these features in Rhythmyx Version 6.5.

Percussion Software will provide an evaluation license of Rhythmyx Version 6.5 for your reference server. Contact your Percussion Software Sales Account Representative to request this license.

## Preparing to Implement the Conversion

Before starting the conversion implementation, attend training on Rhythmyx Version 6.5 to learn about the new features in this release. Then plan your conversion:

- Read this document to the end.
- Determine the resources that you will need to complete the conversion.
- Secure and allocate these resources.

Next, set up an implementation environment (if you did not already do so for the initial upgrade to Rhythmyx Version 6.5). Build a complete copy of your production environment:

- 1 Use a copy of your production server tree and a dump of your production database.
- 2 Set up a Web server to match your production environment
- 3 Test publishing and ensure that the system works as expected BEFORE beginning to implement the conversion.
  - All Content Editors work as designed.
  - All Variants preview correctly and all Snippets are assembled into Pages correctly.
  - Publishing runs are completed without errors, all expected content is published to the expected locations.

Implement your conversions in this development environment, then test them to ensure they are behaving as expected and designed. Once the converted implementation is behaving as desired, migrate your conversions to the production environment.

## Converting Content Lists to Use JCR Queries

In Rhythmyx Version 6.5, Content Lists extract Content Items from the Repository using JCR queries. For details about JCR queries, see “Writing Automated Slot Queries” in the *Rhythmyx Implementation Guide* or “Writing Content List Queries” in the Help for the Publishing tab of Content Explorer.

A Rhythmyx Version 6.5 Content List implemented using a JCR query has five key differences from a Content List implemented in Rhythmyx Version 5.7 and earlier:

- A Rhythmyx Version 6.5 Content List does not require a Rhythmyx application; the JCR query does the work of selecting the Content Items from the Repository.
- A Rhythmyx Version 6.5 Content List specifies the Rhythmyx Site Folder to be published directly in the JCR query. In Rhythmyx Version 5.7 and earlier, the Site Folder to be published is specified by a combination of the Edition definition and Site registration.
- A Rhythmyx Version 6.5 Content List can specify the specific Content Types to publish, which makes it much easier to separate the publishing of binary and non-binary Content Items into different Content Lists. In Rhythmyx Version 5.7, to control which Content Items were published, you had to place Content Items that required special attention in a specially flagged Folder; then add the `IncludeFolder HTML` parameter to the Content List URL to specify whether to publish Content Items in flagged or unflagged Folders.
- The Content List must specify a Template Expander. A Template Expander is an extension that defines how the system will determine the Variants or Templates that will be used to format published Content Items.
- The Content List must specify an Item Filter. An Item Filter is a collection of rules that filters a list of available Content Items. In Content Lists, an Item Filter substitutes for the Authorization Type (Auth Type), ensuring that only the desired Content Items (typically Public Content Items) are published.

The FastForward implementation in Rhythmyx Version 5.7 included four Content Lists, as illustrated in the following screenshot:

	Name	Url	Edition Type
✗	Site Root Full	/Rhythmyx/rx_Support_pub/folder_clist.xml?valid=y,i&delivery=filesystem&pubOp...	
✗	Site Root Incremental	/Rhythmyx/rx_Support_pub/folder_clist.xml?valid=y,i&inc=y&delivery=filesystem...	
✗	Site Root Unflagged	/Rhythmyx/rx_Support_pub/folder_clist.xml?valid=y,i&delivery=filesystem&pubOp...	
✗	Unpublish	/Rhythmyx/rx_Support_pub/unpub_clist.xml?delivery=filesystem	

*Content Lists in Rhythmyx Version 5.7*

When converting to use Content Lists based on JCR queries, we will need at least twice as many Content Lists. For each Site, we will need to define the following Content Lists:

- Full publish of non-binary Content Types
- Full publish of binary Content Types
- Incremental publish
- Unpublish

Note the slight difference from the organization under Version 5.7. In that Version, the Site Root Full Content List published all Content Items of all Content Types (binary and non-binary) on the Site. The Site Root Unflagged published only Content Items in Folders that were not flagged (which, in FastForward, meant only non-binary Content Types, since binary Content Types were stored in Folders that were flagged). In Version 6.5, we are defining unique Content Lists for binary and non-binary Content Types, and will include those Content Lists in different Editions to control whether binary Content Types are published.

Thus, the FastForward implementation for Rhythmyx Version 6.5 includes the following Content Lists to implement equivalent functionality to the FastForward implementation in Rhythmyx Version 5.7

- Corporate Investments Full Binary
- Corporate Investments Full Non-binary
- Corporate Investments Incremental
- Corporate Investments Unpublish
- Enterprise Investments Full Binary
- Enterprise Investments Full Non-binary
- Enterprise Investments Incremental
- Enterprise Investments Unpublish

**Implementing the Enterprise Investments Full Binary Content List**

The Content Lists for the two Sites do not differ significantly, so we will use the Enterprise Investment Content Lists to illustrate the implementation.

We will start by implementing the Enterprise Investments Full Binary Content List. This Content List illustrates all of the new features of JCR-query-based Content Lists with a relatively simple query.

The FastForward implementation includes the following binary Content Types:

- File
- Image
- NavImage

So we want to implement a JCR query similar to the following statement:

```
Select File, Image, and NavImage Content Items from the
Enterprise Investments Site
```

That statement translates to the following JCR query:

```
Select rx:sys_contentid, rx:sys_folderid from
rx:file,rx:image,rx:navimage where jcr path like
'\/Sites/EnterpriseInvestments%'
```

We will name the Content List *rffEiFullBinary*. This name conforms to the naming conventions devised for Rhythmyx Version 6.5. For details about design object naming conventions in Rhythmyx Version 6.5, see “Design Object Naming Conventions” in the *Rhythmyx Version 6.5 Implementation Guide*.

When you create a new Content List in Rhythmyx Version 6.5, a standard URL is created automatically. We will need to modify this URL with the name of the new Content List. In addition, if you use an Assembly Context, you will need to specify the Assembly Context in the URL of the Content List as the HTML parameter *sys\_assembly\_context*. For the purposes of this exercise, we will assume that an Assembly Context has been implemented with the ID *301*.

We will use the Variants associated with the Enterprise Investments Site, which has the ID *301* to format the published Content Items. Finally, we will need to specify an Item Filter. Since we want to publish only Public Content Items, we will use the *Public* Item Filter.

To implement the Enterprise Investments Full Binary Content List:

- 1 Start Content Explorer and go to the Publishing tab.
- 2 In the left navigation, under Content Lists, click the By Name link. Content Explorer displays the Content List editor.
- 3 Click the New Content List link. Content Explorer displays the Edit Content List page with default values:

Edit Content List		Errors		
Id	0-21-330			
Name	<input type="text" value="new_0-21-330"/>			
Description	<input type="text"/>			
Url	<input type="text" value="/Rhythmyx/contentlist?sys_deliverytype=filesystem&amp;sys_contentlist=new_0-21-330"/>			
Type	<input checked="" type="radio"/> Normal <input type="radio"/> Incremental			
Edition Type	Automatic ▾			
Generator	-- Unassigned -- ▾			
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody></tbody></table>	Name	Value	
Name	Value			
Template Expander	-- Unassigned -- ▾			
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody></tbody></table>	Name	Value	
Name	Value			
Item Filter	public ▾			
<input type="button" value="Save"/> <input type="button" value="Cancel"/>				

*Edit Content List page with default values*

- 4 In the Name field, enter *rffEiFullBinary*.
- 5 In the URL field:
  - Change the value of the `sys_contentlist` HTML parameter to *rffEiFullBinary*: `sys_contentlist=rffEiFullBinary`.
  - Add the `sys_assembly_context` HTML parameter with the value *301*: `sys_assembly_context=301`.
- 6 Leave the default values for the Type and Edition Type fields.
- 7 In the Generator drop list, choose the *sys\_SearchGenerator* (*Java/global/percussion/system/sys\_SearchGenerator*). When you select the Generator, the page will refresh and display the Generator parameters. Enter the following JCR select statement as the Value of the query parameter:
 

```
Select rx:sys_contentid, rx:sys_folderid from
rx:file,rx:image,rx:navimage where jcr path like
'//Sites/EnterpriseInvestments%'
```
- 8 In the Template Expander drop list, choose the *sys\_SiteTemplateExpander* (*Java/global/percussion/system/sys\_SiteTempalteExpander*). When you select the Template Expander, the page will refresh and displays the Template Expander parameters. Enter *301* as the Value of the `siteid` parameter. Leave the value of the `default_template` parameter blank.
- 9 In the Item Filter drop list, choose *public*. (This is the default option.).
- 10 Click the [Save] button to save the Content List.

## Implementing the Enterprise Investments Non-binary Content List

Other than the name, the only difference between the Enterprise Investments Non-binary Content List and the Enterprise Investments Binary Content List is the JCR query. For the Non-binary Content List, you will specify non-binary Content Types:

- AutoIndex
- Brief
- Calendar
- Contacts
- Event
- External Link
- Generic
- Generic Word
- Home
- Press Release

The JCR query would be:

```
select rx:sys_contentid, rx:sys_folderid from
rx:rffautoindex,rx:rffbrief,rx:rffcalendar,rx:rffcontacts,
rx:rffevent,rx:rffexternallink,rx:rffgenericword,rx:rffgeneric,
rx:rffhome,rx:rffpressrelease where jcr:path like
'//Sites/EnterpriseInvestments%'
```

## Implementing the Enterprise Investments Incremental Content List

When publishing an incremental Content List, you want to republish all Content Items that were modified since the last publish. JCR query language provides a special query option for the from clause to select all Content Types, `nt:base`. Thus, the query for an incremental Content List would resemble the following:

```
select rx:sys_contentid, rx:sys_folderid from nt:base where
jcr:path like '//Sites/EnterpriseInvestments%'
```

When defining the Content List, for the Type option, select the **Incremental** radio button. Otherwise, use the same procedure as used to implement the Enterprise Investment Binary Content List.

## Implementing the Enterprise Investments Unpublish Content List

When unpublishing Content, you want to select all Content Items that have expired since the last publish, so you will use the same query as was described in “Implementing the Enterprise Investments Incremental Content List”.

You will also want to add the `sys_publish` parameter to the URL with a value of *unpublish*:

```
/Rhythmyx/contentlist?sys_deliverytype=filesystem&sys_publish=
unpublish&sys_assembly_context=301&sys_contentlist=rffEiUnpublish
```

Since you want to unpublish all Content Items on the Enterprise Investments Site that have expired, you would use the same JCR query as the incremental Content List.

Finally, in the Item Filter field, select the *Unpublish* Item Filter. This item filter selects only Content Items that have already been published.

## Updating Editions

Once you have defined the new Content Lists, you can remove the existing application-based Content Lists from your Editions and add the new JCR-query-based Content Lists. For example, in Version 5.7, FastForward was implemented with the following Editions

<b>Edition</b>	<b>Included Content Lists</b>
Full Enterprise Investments	Site Root Full
Incremental Enterprise Investments	Site Root Incremental
Enterprise Investments Unflagged	Site Root Unflagged
Unpublish Enterprise Investments	Unpublish

After conversion, the Enterprise Investments Editions would include the following Content Lists:

<b>Edition</b>	<b>Included Content Lists</b>
Full Enterprise Investments	rffEiUnpublish rffEiFullBinary rffEiFullNonBinary
Incremental Enterprise Investments	rffEiUnpublish rffEiIncremental

<b>Edition</b>	<b>Included Content Lists</b>
Enterprise Investments Unflagged	rffEiFullNonBinary NOTE: The Name of this Edition could be changed to something like “Enterprise Investments Non-binary” or something similar to indicate the Content Types that are actually being published.
Unpublish Enterprise Investments	rffEiUnpublish NOTE: This Edition could also be removed.



## Handling Default Variants

If you use Default Variants, you will need to add a temporary Dispatch Template to ensure that Default Variants are Published correctly. If you do not add this Dispatch Template, Rhythmyx will publish all Variants rather than publishing Default Variants when specified.

You can give the Dispatch Template any name you like. It must have one binding. The binding Variable can have any name, but the value must be:

```
$sys.item.getProperty('rx:default_variantid').long
```

For details about creating a Dispatch Template, see “Dispatch Templates” in the *Rhythmyx Implementation Guide*.

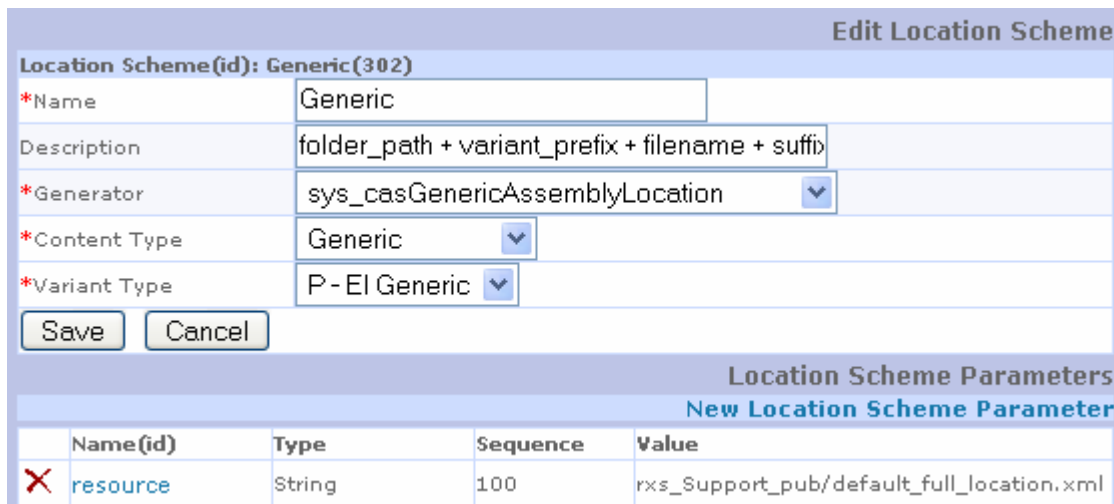
## Converting Location Schemes to Use JEXL

In Rhythmyx Version 6.5, you can use Java Expression Language (JEXL) to generate locations. For details about JEXL, see (JEXL URL). (The currently supported version of JEXL is JEXL 1.0.) For details about writing JEXL expressions, see “Bindings” in the *Rhythmyx Implementation Guide*. For details about implementing Location Schemes using JEXL, see “Defining Contexts and Location Schemes”.

While in Rhythmyx Version 5.7 and earlier, most Location Schemes were implemented using the `sys_casGenericAssemblyLocation` generator. This Location Scheme generator requires a supporting Rhythmyx application to generate the location data. In Rhythmyx Version 6.5, you can achieve the same effect with a simple JEXL expression that you can write directly in the Location Scheme parameter.

To illustrate the conversion of Location Schemes, we will examine the Generic Location Scheme in the Publish Context of the FastForward implementation in Rhythmyx Version 5.7 and the JEXL-based Location Scheme used in Rhythmyx Version 6.5.

In Rhythmyx Version 5.7, the Generic Location Scheme used the `sys_casGenericAssemblyLocation` generator.



**Edit Location Scheme**

**Location Scheme(id): Generic(302)**

\*Name: Generic

Description: folder\_path + variant\_prefix + filename + suffix

\*Generator: sys\_casGenericAssemblyLocation

\*Content Type: Generic

\*Variant Type: P - El Generic

Save Cancel

**Location Scheme Parameters**

New Location Scheme Parameter				
	Name(id)	Type	Sequence	Value
X	resource	String	100	rxs_Support_pub/default_full_location.xml

*Generic Location Scheme in Rhythmyx Version 5.7*

The `resource` parameter specifies the resource that generates the data used to generate the location. In this case, the resource is in the `default_full_location` resource in the application `rxs_SupportPub`. The following graphic illustrates the mapping of this resource:

Back-end	XML	C
<code>sys_casGetSiteBaseUrl(, yes)</code>	<code>PSXXmlField/location/baseurl</code>	<input type="checkbox"/>
<code>rxs_SieFolderAssembly()</code>	<code>PSXXmlField/location/path</code>	<input type="checkbox"/>
<code>CONTENTVARIANTS.LOCATIONPREFIX</code>	<code>PSXXmlField/location/variant_prefix</code>	<input type="checkbox"/>
<code>RXS_CT_SHARED.FILENAME</code>	<code>PSXXmlField/location/filename</code>	<input checked="" type="checkbox"/>
<code>sys_Concat(page, CONTENTSTATUS.CONT...</code>	<code>PSXXmlField/location/filename</code>	<input checked="" type="checkbox"/>
<code>CONTENTVARIANTS.LOCATIONSUFFIX</code>	<code>PSXXmlField/location/variant_suffix</code>	<input type="checkbox"/>
<code>CONTENTSTATUS.CONTENTSSUFFIX</code>	<code>PSXXmlField/location/suffix</code>	<input type="checkbox"/>
		<input type="checkbox"/>
		<input type="checkbox"/>

*Mapping of the default\_full\_location resource*

In this resource:

- The `sys_casGetSiteBaseUrl` UDF retrieves the base URL of the Site from the Site registration.
- The `rxs_SieFolderAssembly` UDF retrieves the Content Explorer Folder path to the Content Item.
- If the Variant being published specifies a prefix or a suffix, they are mapped directly from the Repository to the output XML document.
- If the Content Item being published has a suffix, it is mapped directly from the Repository to the output XML document.
- If the Content Item being published has a filename field and that field has a value, that value is mapped directly to the output XML document; otherwise, the published filename is generated by concatenating the string “page” with the Content Item’s ID.

In Rhythmyx Version 6.5, a JEXL expression achieves the same result:

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

To use a JEXL expression, you must specify the `sys_JexlAssemblyLocation` generator.

In the example expression:

- The variable `$sys.pub_path` generates the publication path of the Content Item. This one JEXL variable accomplishes the same goal as the `sys_casGetSiteBaseUrl` and `rxs_SiteFolderAssembly` UDFs in the Version 5.7 Location Scheme. (Note that the variable `$sys.path` could also be used. In general, `$sys.pub_path` is preferable because it will output the value of the custom property `sys_pubFilename` while `$sys.path` outputs only Foldername.)
- The variable `$sys.template.prefix` retrieves the value of the **Prefix** field in the Template being published, if that field has a value. This value was mapped directly in the resource in the Version 5.7 Location Scheme. Note that this expression does not use the **Suffix** field of the Template, but that could be included in the expression using `$sys.template.suffix`.
- To generate the file name, the string “item” is concatenated with the Content Item ID (which is converted to a string): `...'item' + $sys.item.getProperty('rx:sys_contentid').String...`
- To determine the extension of the output file, the function `$rx.location.getFirstDefned` is used. The first parameter of this function is the Content Item being published (`$sys.item`). The last parameter is a default used if none of the other parameters result in a value. The remaining parameters are the Content Item fields to check for a value. The valued used is the value in the first field in the list that contains a value. In this expression, the fields `activeimage_ext` and `sys_suffix` are specified. If the field `activeimage_ext` contains a value, that value will be used. If the field `sys_suffix` contains a value, that value will be used. If niether of these fields contains a value, the default value (`.html`) is used.

## JEXL Variables Commonly Used in Location Schemes

The following JEXL variables are commonly used when defining a Location Scheme

Variable	Type	Description
<code>\$sys.item</code>	<code>javax.jcr.Node</code>	Contains the fields and children of the current Content Item. Each field, and each child, is considered a property of the <code>\$sys.item</code> variable.  Use the <code>getProperty</code> method to refer to a field of the Content Item. Usually, you will use the <code>getString</code> method to return a string, although you can also return other Java data types. For example, <code>\$sys.item.getProperty(displaytitle).getString</code> refers to the value of the <code>displaytitle</code> field of the current Content Item as a string value. For details see the JavaDoc for <code>javax.jcr.Property</code> in the JSR-170 JavaDoc.
<code>\$sys.crossSiteLink</code>	Booelan	<i>True</i> if the Dependent Content Item in the ActiveAssembly Relationship is on a different Site. Generally useful when generating a URL rather than a delivery location.
<code>\$sys.path</code>	String	The path of the Folder to the Site root Folder. Outputs only the name of the Folder, not the Published name if the two are different. See also <code>\$sys.pub_path</code> .
<code>\$sys.pub_path</code>	String	The publication path of the Content Item. Is essentially the same as the <code>\$sys.path</code> variable, but outputs the published name of any Folder that has a value specified for the <code>sys_pubFilename</code>

Variable	Type	Description
		custom parameter.
\$sys.site.id	IPSGuid	ID of the Site, as a GUID
\$sys.site.path	String	path in the Folder tree from the current Content Item to the Site Folder
\$sys.site.url	String	URL of the Site defined in the <b>Site Address</b> field in the Site registration.
\$sys.site.globalTemplate	String	Name of the Global Template of the Site. If the Global Template is undefined, returns null.
\$sys.template.prefix	String	Location prefix from the Template definition.
\$sys.template.suffix	String	Location suffix from the Template definition.
\$sys.variables	Map<String,String>[]	Provides access to context variables for the current Site.

## JEXL Functions Commonly Used in Location Schemes

The JEXL functions most commonly used when defining a location are the functions of `$rx.location`:

### **\$rx.location.generate**

**Returns** String

Generates the target URL for the assembly item using the default Template.

Can be specified with the template parameter, in which case the link will be generated to the specified Page Template. The value of the Template parameter must be the name of a page Template.

Name	Type	Description
targetitem	<a href="#">IPSAsemblyItem</a>	The target Content Item to which the URL will point.
targetTemplate	String	The specific Template to which the URL will point.

### **\$rx.location.generate**

**Returns** String

Additional signature of the `$rx.location.generate` function that allows the user to specify the data used to generate the URL.

Name	Type	Description
templateinfo	String	The Page Template to which the URL will point.
item	Node	The target Content Item to which the URL will point.
folderPath	String	The Folder location to which the URL will point.
filter	String	The Item Filter to use when generating the URL.
siteid	Number	The ID of the Site to which the URL will point.
context	Number	The publishing Context for which to generate the URL.

**\$rx.location.getFirstDefined**

Returns String

Looks through the set of fields specified to find a field that contains a value. The first field in the list specified that contains a value will be used. If none of the specified fields contains a value, the default value will be used.

Name	Type	Description
item	Node	The Content Item whose fields to search.
listofproperties	String	Comma-separated list of fields to search for a value.
defaultvalue	String	default value to use if none of the specified fields contains a value.

**\$rx.location.siteBase**

Returns String

Returns the base URL for the specified Site. Can include the modify parameter. If the modify parameters is set to yes [`siteBase($sys.assemblyitem, yes)`], the protocol, host, and port are not included in the returned URL.

Name	Type	Description
siteid	String	The Site whose base URL to return
modify	String	If the value of this parameter is "yes", the protocol, host, and port will be stripped from the base URL. Otherwise, the protocol, host, and port are included. (Optional)

You can also used Velocity tools functions (\$tools); for details see see the Velocity tools documentation (<http://jakarta.apache.org/velocity/tools/index.htm>).

The following tools are available:

Tool	Class
\$tools.alternator	org.apache.velocity.tools.generic.AlternatorTool
\$tools.date	org.apache.velocity.tools.generic.DateTool
\$tools.esc	org.apache.velocity.tools.generic.EscapeTool
\$tools.mill	org.apache.velocity.tools.generic.IteratorTool
\$tools.list	org.apache.velocity.tools.generic.ListTool
\$tools.math	org.apache.velocity.tools.generic.MathTool
\$tools.number	org.apache.velocity.tools.generic.NumberTool
\$tools.render	org.apache.velocity.tools.generic.RenderTool
\$tools.sorter	org.apache.velocity.tools.generic.SortTool
\$tools.parser	org.apache.velocity.tools.generic.ValueParser

You may also define customg JEXL functions, which begin with the prefix \$rx. For details about developing custom JEXL functions, see the *Rhythmyx Technical Reference*.

## Converting to Velocity Assembly

Converting assembly to use Velocity Templates is a three-step process:

- 1 Implement new Managed Navigation Templates using Velocity.
- 2 Convert Global Templates to Velocity.
- 3 Convert Local Templates to Velocity.

Note that Steps 1 and 2 must be done together. Once Managed Navigation and Global Templates are converted, you can begin publishing using these Templates while continuing to use XSL Variants. You can then convert XSL Variants at your leisure.

### Implementing Managed Navigation Templates in Velocity

Before implementing new Managed Navigation Templates, create a new Managed Navigation Slot. A Managed Navigation Slot uses the `sys_ManagedNavContentFinder`. For details about creating a Slot, see “Creating Slots” in the *Rhythmyx Version 6.5 Implementation Guide*. For details about Managed Navigation Slots, see “Managed Navigation Slot” in the *Rhythmyx Version 6.5 Implementation Guide*. You can also use the Managed Navigation Slot in the Version 6.5 reference server as a model.

For details about implementing Managed Navigation Templates, see “Creating Managed Navigation Templates in the *Rhythmyx Version 6.5 Implementation Guide*. In the Rhythmyx Workbench, model Templates for the following common Managed Navigation Templates are provided:

- breadcrumbs (breadcrumb nav)
- top navigation (top nav)
- left navigation (left nav)

You can base your own Managed Navigation Templates on these models.

### Converting Global Templates to Velocity

Converting Global Templates to Velocity is a two-step process:

- 1 Create new Global Template objects in the Rhythmyx Workbench. When creating the Global Template objects, specify the existing Global Template HTML as the source for the Template. For example, when converting the Enterprise Investments Global Template, we would specify `<Rhythmyxroot>/rxs_GlobalTemplates`. For details about creating Global Template objects, see “Creating Global Templates” in the *Rhythmyx Version 6.5 Implementation Guide*.

- 2** In the Template object in the Workbench, on the Source tab, convert XSpLit markup to Velocity markup:
- a) For local content, convert XSpLit field markup [psx-fieldname] to Velocity macro markup [#field ("fieldname")]. In Global Templates, the most common local content is the title, which generally uses the #displayfield macro. For example, in the Enterprise Investments Global Template, the title tag is converted from <title>psx-shared/displaytitle</title> to <title>#displayfield("displaytitle")</title>.

---

NOTE: The #displayfield macro used in this example is used for fields that should not be modified in Active Assembly. The #field or #field\_if\_defined macros are used for fields that are eligible for Active Assembly. For additional details about Velocity markup, see “Creating Templates” in the *Rhythmyx Implementation Guide*.

---

- b) Slot markup is converted from XSpLit formatting

```
<!-- start slot slotname -->
<!-- start snippet wrapper -->
<span psxeditslot="no" slotname="slotname">Slot
Name</span>
<!-- end snippet wrapper -->
<!-- end slot slotname -->
```

to Velocity markup:

```
#slot("SlotName" "" "" "" ""
"template=NavTemplateName")
```

For example, in the Enterprise Investments Global Template, the Top Navigation is converted from

```
<!-- start slot nav_top -->
<!-- start snippet wrapper -->
<span psxeditslot="no" slotname="nav_top">Top
Nav</span>
<!-- end snippet wrapper -->
<!-- end slot nav_top -->
```

to

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

- c) Convert references to static files (such as links to Cascading Stylesheet or JavaScript files) to use Template Bindings. For details about Bindings, see “Bindings” in the *Rhythmyx Version 6.5 Implementation Guide*. For details about the use of Bindings to implement references to static files, see “Converting References to Static Files”. You can continue to use the existing variables, but must define those variables in the Bindings of the Global Template.

The following code snippets show key sections of the Enterprise Investments Global Template before conversion and after:

### Enterprise Investments Global Template Before Conversion

```

<html lang="en">
  <head>
    <title>psx-shared/displaytitle</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    <meta name="keywords" content="Key Words" />
    <meta name="description" content="Description" />
    <link rel="stylesheet" psx-
href="$rxs_navbase/css/rxs_styles.css"
href="web_resources/css/rxs_styles.css" type="text/css" />
    <script psx-src="$rxs_navbase/js/mouseover.js"
language="javascript" type="text/javascript"></script>
  </head>
  <body>
    <!-- start slot nav_preload -->
    <!-- start snippet wrapper -->
    <div slotname="nav_preload" psxeditslot="no" />
    <!-- end snippet wrapper -->
    <!-- end slot nav_preload -->
    <div id="EIPage">
      <div id="TopPortion">
        ...
        <!-- ===== HORIZONTAL NAV STARTS HERE ===== -->
      >
      <div id="horizontal_nav">
        <!-- start slot nav_top -->
        <!-- start snippet wrapper -->
        <span psxeditslot="no" slotname="nav_top">Top Nav</span>
        <!-- end snippet wrapper -->
        <!-- end slot nav_top -->
      </div>
    </div>
    <div id="MainPortion">
      <div id="LeftSide">
        <div id="LeftNav">
          <!-- start slot nav_left -->
          <!-- start snippet wrapper -->
          <span psxeditslot="no" slotname="nav_left">nav
left</span>
          <br />
          <!-- end snippet wrapper -->
          <!-- end slot nav_left -->
        </div>
        ...
      </div>
      <div id="MainBody" psx-localtemplate="yes">Local Body
Template Here</div>
      <div id="SiteFooter">
        <!-- start slot nav_bottom -->
        <div id="nav_lbottom">
          <!-- start snippet wrapper -->

```



```

        <span slotname="nav_bottom" psxeditslot="no">Bottom
Nav</span>
        <!-- end snippet wrapper -->
    </div>
    <!-- end slot nav_bottom -->
</div>
</div>
</body>
</html>

```

### Enterprise Investments Global Template after Conversion

```

<html lang="en">
  <head>
    <title>#displayfield("displaytitle")</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
    <meta name="keywords" content="#displayfield('keywords')" />
    <meta name="description"
content="#displayfield('description')" />
    <meta content="Percussion Rhythmyx" name="generator"/>
    <link rel="stylesheet" href="$rxs_navbase/css/rxs_styles.css"
type="text/css" />
    <script src="$rxs_navbase/js/mouseover.js"
    language="javascript" type="text/javascript"></script>
  </head>
  <body>
    #slot("rffNav" "" "" "" "" "template=rffSnNavPreload")
    <div id="EIPage">
      <div id="TopPortion">
...

    <div id="horizontal_nav">
      #slot("rffNav" "" "" "" "" "template=rffSnEiTop")
    </div>
    <div id="MainPortion">
  <div id="LeftSide">
    <div id="LeftNav">
      #slot("rffNav" "" "" "" "" "template=rffSnEiNavLeft")
      <br />
    </div>
...
  </div>
  <div id="MainBody">
    #inner()
  </div>
  <div id="SiteFooter">
    #slot("rffNav" '<div id="nav_lbottm">' "" "" '</div>'
"template=rffSnEiNavBottom")
  </div>
  </div>
</body>
</html>

```

## Updating Slots

Before converting Local Variants, you should update the Slots in the Workbench. When using Velocity to assemble Content, each Slot must include a Content Finder. A Content Finder is an extension that retrieves the Content Items to be included in the Slot. For most Slots, you should specify the `sys_RelationshipContentFinder`, but there are some exceptions:

- For the Managed Navigation Slot, specify the `sys_ManagedNavContentFinder`
- For Automated Slots, specify either the `sys_AutoSlotContentFinder` (if you want to use a JCR query to populate the Slot) or the `sys_LegacyAutoSlotContentFinder` (if you want to use a Rhythmyx query resource to populate the Slot).
- If you have Sites in multiple languages, use the `sys_TranslationContentFinder` to generate automatic links to translations of Content Items in different Languages.

## Converting Local Variants to Velocity Templates

Rhythmyx Version 6.5 includes a Variant to Template Migration Tool that creates new Velocity Templates based on existing XSL Variants.

Note that in Rhythmyx Version 5.7 and earlier, even though multiple Variants could use the same assembly resource, each combination of Content Type and Variant required a unique Variant registration. Multiple registrations are no longer necessary in Rhythmyx Version 6.5. Multiple Content Types can be associated with a single Template. Thus, the Variant to Template Migration Tool will create only one Velocity Template for each assembly resource; each Content Type that has a Variant associated with that resource will automatically be associated with the new Template.

---

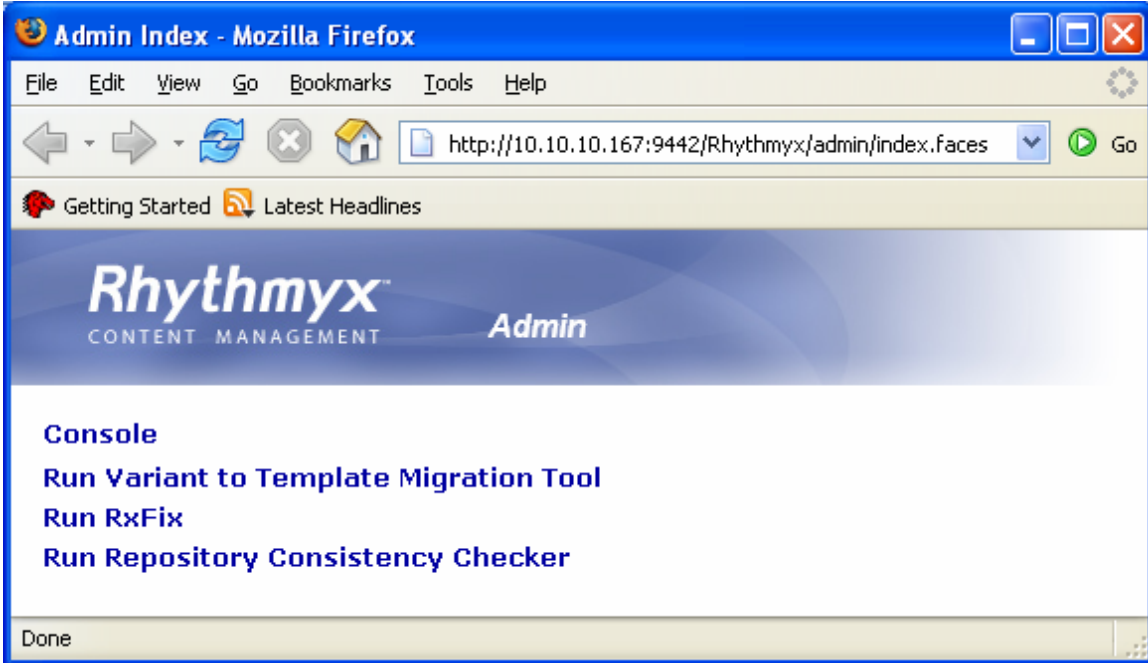
NOTE: While Managed Navigation Variants are listed in the Variant to Template Migration Tool, these Variants should not be converted because Managed Navigation works differently under Velocity than it does under XSL. Managed Navigation should be reimplemented in Velocity as discussed in “Implementing Managed Navigation Templates in Velocity” on page 14, rather than converted.

---

The Variant to Template Migration Tool imports the source HTML for the Variant into the Template and displays it on the Source tab of the Template editor.

To run the Variant to Template Migration Tool:

- 1 Start a browser and enter the URL of the Rhythmyx Admin page:  
<http://host:port/Rhythmyx/admin>. where `host` is the name or IP address of the machine where Rhythmyx resides and `port` is the port on which the Rhythmyx server listens. You will have to log in to get to the admin page.



*Rhythmyx admin page*

- 2 Click the Run Variant to Template Migration Tool link.  
 Rhythmyx displays the Variant to Template Migration Tool page.

Choose from which variant sets to create templates. You can accept or change the names of the created templates here as well.

Selected	Variant Name	Template Name	Resource
<input type="checkbox"/>	NavTreeLink	NavTreeLink_v	../rxs_NavTree_cas/navtree.html
<input type="checkbox"/>	NavonLink	NavonLink_v	../rxs_Navon_cas/navon.html
<input type="checkbox"/>	P_CI_Category_List	P_CI_Category_List_v	../rxs_Category_cas/p_category.html
<input type="checkbox"/>	P_CI_Event	P_CI_Event_v	../rxs_Event_cas/page.html
<input type="checkbox"/>	P_CI_Generic	P_CI_Generic_v	../rxs_Shared_cas/p_shared.html?shared_variantid=1000
<input type="checkbox"/>	P_CI_Generic1	P_CI_Generic1_v	../rxs_GenericWord_cas/p_shared.html
<input type="checkbox"/>	P_CI_Home	P_CI_Home_v	../rxs_Home_cas/page.html
<input type="checkbox"/>	P_CI_Press_Release	P_CI_Press_Release_v	../rxs_PressRelease_cas/page.html
<input type="checkbox"/>	P_EI_Category_List	P_EI_Category_List_v	../rxs_Category_cas/p_category.html
<input type="checkbox"/>	P_EI_Event	P_EI_Event_v	../rxs_Event_cas/page.html
<input type="checkbox"/>	P_EI_Generic	P_EI_Generic_v	../rxs_Shared_cas/p_shared.html?shared_variantid=1

*Variant Converter page*

### Field Descriptions

**Selected** Checkbox. Check this box to select an XSL Variant to convert to a Velocity Template.

**Variant Name** Read-only. Name of the XSL Variant. If multiple XSL Variants use the same Rhythmyx assembly resource, all will be listed in this field.

**Template Name** Name of the Velocity Template that will be created in the Rhythmyx Version 6.5 Workbench. The name defaults to the name of the XSL Variant with the suffix “\_v”. You can change the name of the Template. (NOTE: Do not change the name of the Template to match the name of the Variant. Multi-Server Manager will not package your Templates and Variants properly if the names of Templates match the names of Variants. Also, do not change the name of the Variants. Changing the name of your Variants could also cause errors when packaging Multi-Server Manager archives.)

**Resource** URL of the assembly resource specified by the XSL Variant

### Converting Variants

To convert Variants

- 1 Start the Variant to Template Migration Tool.
- 2 Check the Selected box in the row of the Variants you want to convert.  
NOTE: Percussion Software recommends that you do not convert Managed Navigation Variants
- 3 Optionally, change the default name of one or more Templates. For example, the default name for the P\_EI\_Generic Template created from the P-EI Generic Variant is *P\_EI\_Generic\_v*. To conform to the new naming conventions, the name of the Template could be changed to *rffPgEIGeneric*.

---

(NOTE: Do not change the name of the Template to match the name of the Variant. Multi-Server Manager will not package your Templates and Variants properly if the names of Templates match the names of Variants. Also, do not change the name of the Variants. Changing the name of your Variants could also cause errors when packaging Multi-Server Manager archives.)

---

- 4 Click the [**Convert**] button to launch the conversion process.

- When the this process is complete, Rhythmyx returns the results page. This page displays the results of the conversion and any errors that occurred during the conversion process.

Results from converting selected variants to templates.

Name	Resource	New Template Name
P_EI_Category_List	../rxs_Category_cas/p_category.html	P_EI_Category_List_v
P_EI_Event	../rxs_Event_cas/page.html	rffPEIGeneric
P_EI_Generic	../rxs_Shared_cas/p_shared.html?shared_variantid=1	P_EI_Generic_v
P_EI_Generic1	../rxs_GenericWord_cas/p_shared.html	P_EI_Generic1_v
P_EI_Home	../rxs_Home_cas/page.html	P_EI_Home_v
P_EI_Press_Release	../rxs_PressRelease_cas/page.html	P_EI_Press_Release_v
P_Monthly	../rxs_Calendar_cas/p_month.html	P_Monthly_v
S_AutoList	../rxs_AutoIndex_cas/s_autoindex.html	S_AutoList_v
S_Auto_Bullet_List	../rxs_AutoIndex_cas/s_autoindex.html	S_Auto_Bullet_List_v
S_Bulleted_List	../rxs_Category_cas/s_categoryNonAuto.html	S_Bulleted_List_v
S_Category_Summary	../rxs_Category_cas/s_category.html	S_Category_Summary_v
S_Date_Range	../rxs_Event_cas/snip.html	S_Date_Range_v

*Variant Converter results page*

- Next, update the HTML in the Source tab of each Template, substituting Velocity markup for XSpLit markup. You will also need to add Bindings to the Template to support image and hypertext links and other processing. For details about Velocity markup, see the following topics in the *Rhythmyx Implementation Guide*:

For information about...	See this topic
Implementing Snippets	“Implementing Snippet Templates”
Implementing Pages	“Creating Page Templates”
Implementing hypertext links	“Implementing Complex Snippets”
Implementing image links	“Implementing a Binary Template”
Bindings in general	“Bindings”
Automated Slots	“Creating an Automated Slot”
Including Slots on a Template	“Creating Page Templates”
Default Templates (implemented in Rhythmyx Version 6.5 using Dispatch Templates)	“Dispatch Templates”

When marking up the Template HTML with Velocity code, be sure to check the Mapper on the original Variant resource for processing logic, as well as the XSL. If you find additional processing logic, you will need to re-implement it in one of the following ways

- develop Bindings;
- develop new Velocity macros;
- develop additional inline Velocity.

You will need to assess the processing logic to determine which option is best suited to produce the desired results.

Also, when marking up Template HTML with Velocity, consider implementing custom Velocity macros to reduce the code in each Template. For example, suppose your Page Templates share a common HTML header:

```
<head>
  <title>#displayfield("displaytitle")</title>
  <link href="$sys.variables.css/Public.css"
    rel="stylesheet" type="text/css" />
</head>
```

You could create a macro such as the following:

```
## Standard Header used in regular Snippets
#macro(snippet_head $field_parm)
  <head>
    <title>#displayfield("$field_parm")</title>
    <link href="$sys.variables.css/Public.css"
      rel="stylesheet" type="text/css" />
  </head>
#end
```

You could then use the following formatting in the HTML:

```
<html>
  #snippet_head("displaytitle")
  <body>
  ...
```

7 The final step in the process is to repoint ActiveAssembly Relationships from the Variant to the new Template. Use the RhythmyxVariantConverter tool (<Rhythmyxroot>\VariantConverter\RhythmyxVariantConverter.exe or <Rhythmyxroot>/VariantConverter/RhythmyxVariantConverter.sh) to complete this conversion. This tool uses a properties file. To run the conversion:

a) Open the file <Rhythmyxroot>/VariantConverter/variantsconverter.properties in a simple text editor.

b) Update the Login Account Information:

```

hostName=name or IP address of Rhythmyx machine
port=Rhythmyx port
loginId=username (specified user must have design
rights)
loginPw=password of the specified user
useSSL=false (default; change to true if you want to use
SSL)
serverRoot=Rhythmyx (this property should not be
modified)
communityId=ID of the Community of the Content Items you
want to convert. This property is optional. If not
specified, the last Community logged by the specified
user will be used.
    
```

c) Specify the Workflows and Transitions used by when repointing Content Items in the Public State. (Content Items that are not Public do not have to be Transitioned to be converted.) Use the following pattern:

```

Workflow_Name=TransitionfromPublic;TransitiontoPublic
where
    
```

Workflow\_Name is the name of a Workflow used by Content Items whose Active Assembly Relationships you want to repoint

TransitionfromPublic is the name of the Transition to use to move the Content Item from the Public State to an editable State (typically, QuickEdit is used for this purpose).

TransitiontoPublic is the name of the Transition to use to return the Content Item to the Public State after processing (typically, ReturnToPublic is used for this purpose).

d) In the variantMap property, specify which Variants should point to which Template. Use the pipe character (|) to separate the Variants from the Templates. Use commas to separate each Variant/Template pair. For example, the default mapping property:

```

variantMap=301|501,302|502
    
```

repoints Active Assembly Relationships that refer to Variant ID 301 to use Template ID 501 and repoints those that refer to VariantID 302 to use Template ID 502. Note that the specified Templates must be Page Templates.

---

NOTE: Use the Properties view in the Rhythmyx Workbench to access these IDs. The Properties view is available by default under the Navigation view, next to the Snippet Drawer. Select the Variant or Template whose ID you need in the Navigation view, then in Properties view, select the ID property. Right-click and choose *Copy* from the popup menu. You can then paste the ID into the variantsconverter.properties file. The entire property will be pasted:

```

ID 0-4-343 (17179869527)
    
```

---

---

Only the three-digit ID number is required. The rest of the text must be removed or the RhythmyxVariantsConverter will fail.

In the example above, the ID is 343. The rest of the text must be removed.

---

- e) You can also specify specific Content Items whose ActiveAssembly Relationships you want to repoint. Specify the ID of these Content Items in the contentId property of the variantsconverter.properties file. Use commas to separate multiple IDs. You can access Content Item IDs by viewing the Content Item properties in Content Explorer.
- f) After you have defined all of the conversion properties, run RhythmyxVariantsConverter.exe or RhythmyxVariantsConverter.sh.