

Rational. Publishing Engine



User Guide

Rational Publishing Engine
User Guide
Release 1.1

Before using this information, be sure to read the general information under Appendix, [“Notices” on page 285](#).

This edition applies to **VERSION 1.1, Rational Publishing Engine** and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2009**

US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

Rational Publishing Engine	1
Introduction	1
Software requirements	1
RPE and DocExpress	1
RPE and WEXP	2
Key differences	2
RPE Resources	2
Concepts	3
Document template	3
Data source schema	3
Creating data source schemas	4
Document specification	4
Concrete data sources	5
Output	7
Workflow	8
Publishing documents with RPE from a data source	8
Data	8
Schema display	8
Queries and contexts	10
Filtering data	12
Sorting data	13
Data Source pre-processing	13
Dynamic Data sources	14
Defining a “Data Source Configuration”	14
Dynamic data	16
Inherited Data Configuration	16
Template flow	18
DOORS Data	18
Configuring the data source	19
Configuration Wizard	20
DOORS Schema	26
Queries and attributes	28

Images	30
OLEs	31
Tables	32
DOORS External Links	32
DOORS Links	33
Recursive retrieval	33
Filtering	33
Filtering by direction	33
Filtering by link module	34
Filtering by target module	34
Combining filters	34
What cannot be extracted	34
DOORS Database Structure	35
Configuring the data source	35
DOORS Database Structure Schema	35
Database	36
Folder and Project	36
Module	37
Baselines and views	37
Filtering	37
Sorting	37
Retrieving information recursively	37
Tau Data	38
Configuring the data source	38
Tau Schema	41
Queries	41
Nested queries	42
How RPE builds queries	43
Filtering	43
Native Filter	43
RPE filter	44
Sorting Tau data	44
Type casting	44
Cast vs. Filter	48
Attributes	48
Special attributes	48
GUID	48
_image	48
stringRepresentation	49

The query element	49
XML	50
Configuring the data source	50
What can be extracted	52
REST Data	52
Obtaining a schema	52
Configuring a REST Data Source	52
Resource Discovery Wizard	54
Filtering	64
Native filtering	65
RPE Output	66
Stylesheets	66
Word	66
Defining a Word output	66
Word 2003 and 2007 formats	67
Output	67
Stylesheets	67
Unicode data	67
OLEs	68
Post-processing	68
Word macros	68
updateTOCs	68
updateFields	69
updateTablesOfFigures	69
insertOLEs	69
RPE	69
Self contained Word document	69
HTML	69
Defining an HTML output	69
Output	70
Stylesheets	70
Unicode data	70
PDF	70
Defining a PDF output	70
Stylesheets	70
Unicode data	71

XSL-FO	71
Defining an XSL-FO output	71
Output	71
Stylesheets	71
Unicode data	71
Post-processing	72
RPE Launcher	72
Defining a document specification	73
Metadata	73
Multi - template document specifications	74
Functions	75
Preferences	75
Engine preferences	76
Shared template library	77
Preview preferences	78
Synchronizing document specifications with templates	79
Generating a document	79
Pause and abort	80
Results dialog	81
Local engine installation	81
Remote engine results	83
Command line switches	83
RPE Publish Wizard	83
RPE Document Studio	91
Template elements	93
Components	96
Palette	96
Schema view	97
Editor area	98
Outline	98
Properties	98
Content	98
Adding elements	99
Naming elements	99
Editing element properties	100
Formatting properties	101
Calculated properties	103

Create New Template Wizard	105
Data	115
Add Data Source Schema Wizard.	115
Schema discovery	120
Schema view	121
Queries	122
Using data attributes in template elements	124
Content editor	126
Condition editor	129
Filter	130
Sort	133
Data Contexts	135
Nested queries	136
Advanced usage of data attributes	138
Calculated values	138
Conditions	139
Filter vs. Conditions	140
Conditional formatting	141
Styles	144
Creating a style	144
Editing a style	150
Changing style properties	151
Applying a style	152
Using external styles	154
Style precedence	154
Style inheritance	154
Reserved style names	154
Master pages	154
Adding a master page	154
Using master pages	157
Data attributes in master pages	158

Advanced template concepts	161
Bookmarks	161
Name mangling	161
Hyperlinks	161
Internal hyperlinks	161
Variables	161
User variables	162
RPE Variables	162
Variable assignments	162
RPE Predefined Template	165
RPE launcher integration	165
Synchronized mode	166
Generate & preview	167
Unsynchronized mode	167
Schema Discovery	167
DOORS Schema Discovery	167
Schema discovery for current module	171
Linked object attributes discovery	177
Tau Schema Discovery	182
DOORS Addin	187
Installation	188
Usage	188
Running from Database Menu	188
Running from Module menu	188
Tau Addin	188
Installation	188
Usage	189
Deployment scenarios	189
Local engine	190
Remote engine	190

How to	190
Tips & tricks	190
Editable elements	190
Table of Contents in Word documents	190
Figure captions in Word documents	190
Included files	190
Heading styles	191
Formatting properties vs. Styles	191
RPE Styles vs. External Styles	191
Numbering headings for Microsoft Word	191
Dynamic images	191
Dynamic included files	191
Unicode data in output	191
Moving an element outside the visible editor region	192
Table fit to window	192
PDF Tables	192
Table merge	192
Once per table	192
Frequently asked questions	193
Troubleshooting RPE	194
Common problems	195
RPE Core Logging	195
RPE UI Logging	196
RPE Core Debug Mode	196
Customer feedback module	197
Examples	199
A template for DOORS data	199
Basic setup	199
Advanced template options	209
Using DOORS Link	224
Generate the document	229
Configure the Data source	230
Generate the document	232
View the results	234

A template for Tau data	234
Define the template content	238
Generate the document	251
An example for REST Data	254
Obtain the schema	255
Create the template content	263
Configuring the data source	277
Generate the document	278
Annex	281
Library definition file	281
Known problems & limitations	282
Appendix A: Notices	285
Trademarks	288

Rational Publishing Engine

Introduction

Documents continue to be the preferred and most effective way to communicate necessary information across all disciplines, both internally and across contractual boundaries. Creating documents and keeping them up-to-date can be time consuming and error prone. Documents can even become outdated by the time you have finished writing them.

Rational Publishing Engine automates documentation generation from Rational products and select third party applications, helping organizations generate documents for ad-hoc use, formal reviews, contractual obligations, or compliance with standards.

With Rational Publishing Engine you spend less time on the tedious and manual task of creating documents and keeping them up to date.

Benefits

- Reduce Risk of Errors through systematic creation of documents
- Enhance Customer Satisfaction with more effective communication
- Improve Regulatory Compliance by automatically generating required documents

Features

- Built-in extractors support many data sources
- Multiple output formats with complete flexibility in appearance
- Concurrent document generation to multiple target formats from a single template
- Composite reports containing data from multiple sources
- Predefined templates included out-of-the-box for rapid adoption

Software requirements

RPE requires Java Runtime Environment 1.6 or later. (IBM and Sun JREs supported)

For DOORS document generation a DOORS 9.1 or newer client installation is required.

For Tau document generation a Tau 4.2.0.1 or newer client installation is required.

RPE 1.1 is available on the platforms listed on the readme notes.

NOTE For installing and configuring RPE please consult the installation guide.

RPE and DocExpress

While serving the same purpose, RPE and DocExpress are fundamentally different in their approach to data generation. RPE borrows some DocExpress concepts that proved valid, such as embedding document generation capabilities within the data source application GUI, but its architecture, design and implementation are completely new and different from DocExpress.

To minimize the learning curve it is important to view RPE as a brand new product.

RPE and WEXP

Like in the case of DocExpress, RPE and WEXP are fundamentally different in their approach to data generation. RPE borrows some WEXP concepts that proved valid, such as allowing non linear document flows, the ability to define the target data source at runtime but its architecture, design and implementation are completely new and different from WEXP.

Key differences

RPE	DocExpress	WEXP
Template driven generation	Data driven generation	Data driven generation
Does not alter the data source. For the same data source more than 1 document template can be defined.	Requires changes in the data source	Requires changes in the data source
Supports DOORS, Tau, RESTfull data sources and generic XML	DOORS, Tau and XML.	Limited to DOORS
Supports multiple data sources and data source types in the same document generation	Data from different data sources can be generated in the same document but not interleaved.	
Word, PDF, HTML, XSL-FO	Word output, Frame Maker and limited HTML.	Word Output
Supported on Windows and Red hat Enterprise Edition 5	Windows only	Windows only
Does not use clipboard for document generation	Uses clipboard for data transfer	
Does not require Microsoft Word	Requires Microsoft Word	Requires Microsoft Word
Can generate HTML, PDF		
Improved performance		
Improved formatting capabilities		

Table 1 DocExpress, WEXP, RPE comparison

RPE Resources

In addition to the RPE documentation there are a host of other sources of information for RPE.

RPE Support Site <http://www-01.ibm.com/software/awdtools/pubengine/support>

RPE Communities <http://www-01.ibm.com/software/awdtools/pubengine/support/community.html>

RPE Community Wiki – contains additional documentation, examples, tips and tricks
RPE Developer Works Forum

Concepts

RPE relies on several key concepts, described in the following sections:

- Document Template
 - Data source schema
 - Template content
 - Master pages
 - Variables
- Document Specification
 - Concrete data source
 - Document output

Document template

This is the *blueprint* for the document generation. A document template defines what data is to be extracted from the data source (*queries*) and how to format all the information (*formatting information*).

A template is built of static and dynamic content. The static content is defined by the data such as texts and images provided when the template is designed. The dynamic content is represented by data obtained from the data sources at document generation time.

Formatting information can be defined in the template, while for some data sources; formatting information embedded in the data can also be retained.

NOTE A document template is self contained. It is stored as an archive file with the extension *.dta* (Document Template Archive). You can share/move/copy a template freely with no implications.

NOTE RPE 1.1 is backward compatible with document templates however document templates created with RPE 1.1 cannot be used in previous versions.

Data source schema

A document template does not make references to concrete data sources. When defining the template you will be using the definition of your data's structure, its schema. This allows for the template to be applied to any data source with a structure matching the one used when defining the template.

A RPE template can contain any number of data source schemas.

RPE uses standard XML Schema Definition for data source schemas. <http://www.w3.org/XML/Schema>

NOTE A single data schema can be used for any number of *matching* concrete data sources.

NOTE When a data source does not perfectly match the data schema, RPE will process the elements for which a match is found, and ignore these non-critical errors. For example if a DOORS Attribute named “status” is used in the template and the actual data source (a DOORS module) does not have this attribute, then the document generation will (by default) continue.

Creating data source schemas

RPE provides the mechanisms for creating or obtaining schemas for several data source types such as DOORS, Tau and Rational RESTfull Data Sources. For generic XML files you need to provide the schema yourself or use a 3rd party application for inferring the schema from the XML data.

For details on how schemas are obtained for DOORS, Tau and Rational RESTfull data sources please consult the Schema Discovery discussion in the Document Studio section.

Document specification

Document templates define the workflow and content of the document while the document specification defines the document templates to be used, the concrete data sources assigned to each template data source schema and the output formats.

NOTE A single document specification can use any number of templates.

NOTE The document specification does NOT package any of the templates it uses or other resources. If you copy or move a document specification to a different location/machine then you need to copy all of the templates/resources or ensure that the templates are located on a shared location visible from the new document specification location.

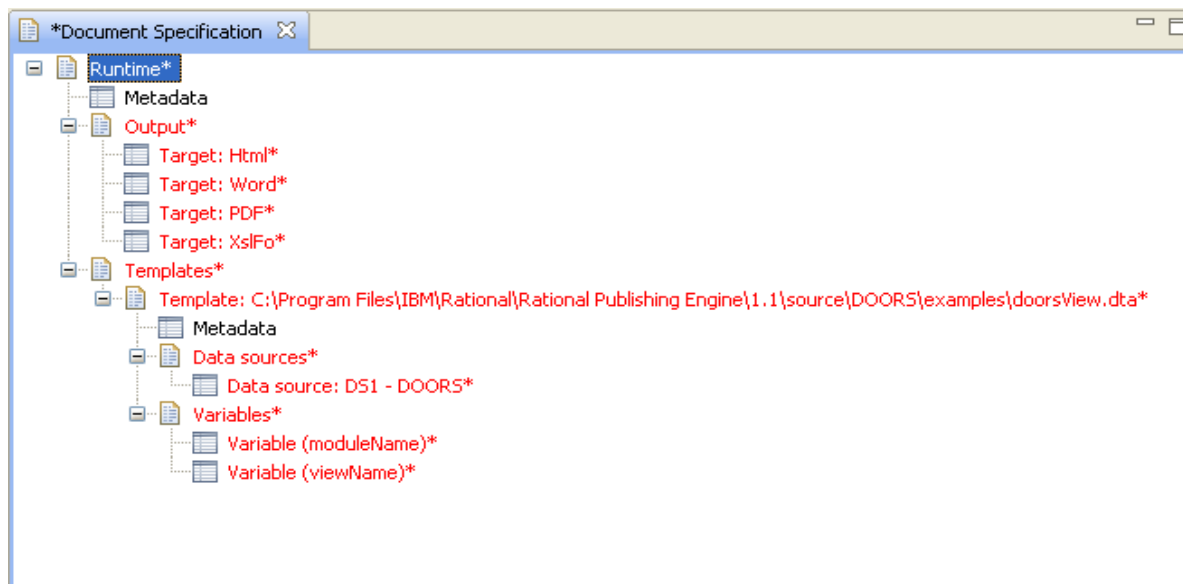


Figure 1 Document Specification with template from local file system

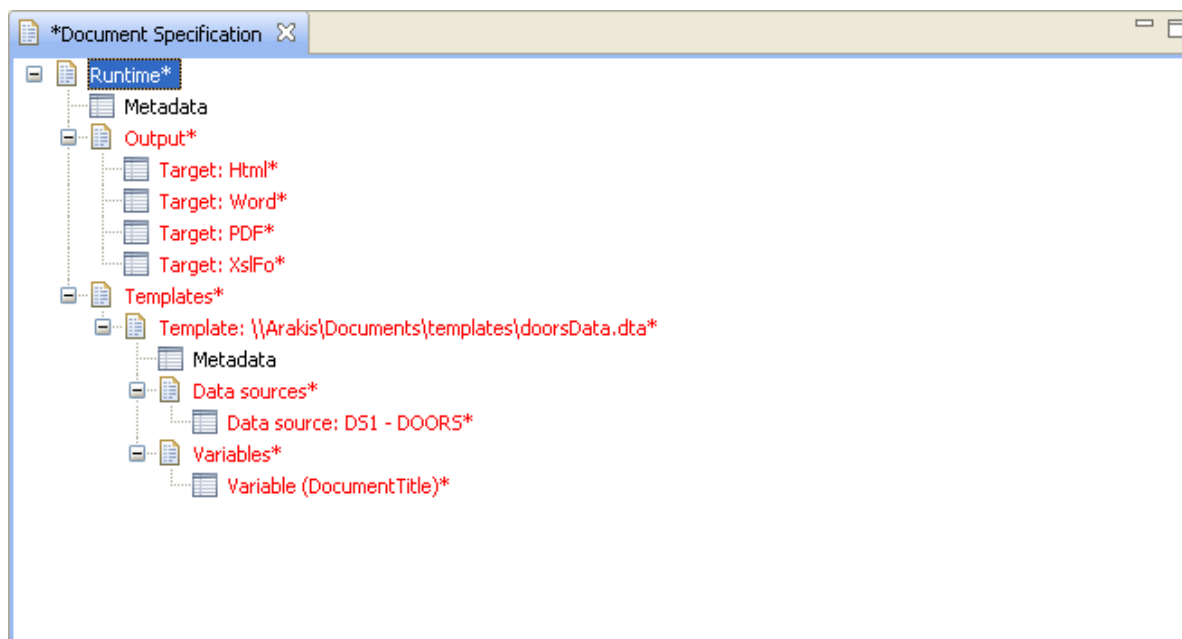


Figure 2 Document Specification with shared template

Concrete data sources

Configuring a data source means providing RPE with the details needed to locate and access the actual data source. Currently RPE supports the following data source types:

- DOORS
- Tau
- Generic XML
- Rational tools implementing the Rational REST Get Specification

Each type of data source has different properties that need to be configured. See the Input section for details on the specific properties.

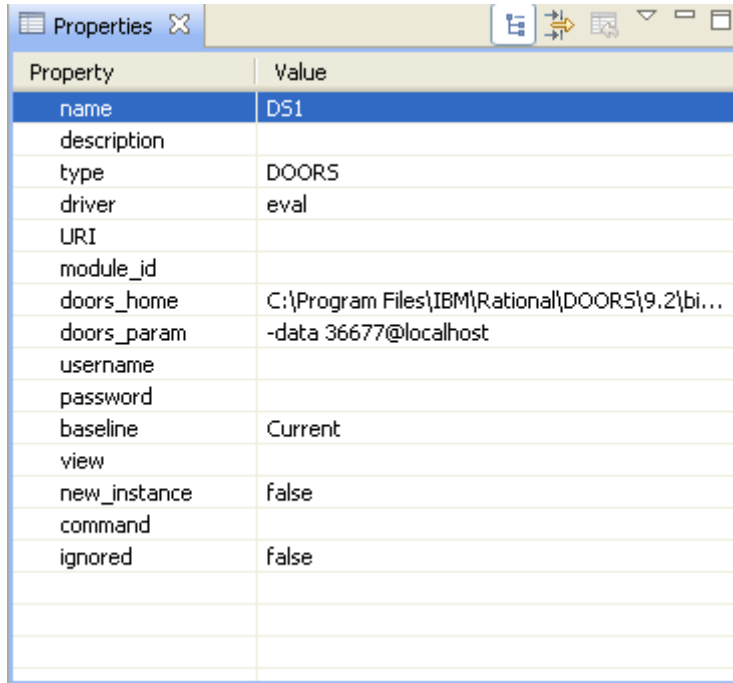


Figure 3 A Concrete DOORS Data source

The properties shared by all data sources are listed below:

Property	Description
name	The name of the data source as referred in the document template. <i>Read only</i>
description	The description provided in the document template for the data source. <i>Read only</i>
type	The type of the data source. <i>Read only</i>
driver	Property maintained and used internally by RPE. Must not be modified.
URI	The URI of the data source. The form and shape of the URI depends on the actual data source. For some data sources the URI is a URL.
command	A system command to be executed before RPE starts loading the data.

Ignored	Possible values: true/false. Default value: false If set to true, RPE will ignore all the template queries using the data source.
---------	---

Output

RPE 1.1 supports the following output formats:

Microsoft Word (97-2003 and 2007 format)

HTML

PDF

XSL-FO (see <http://www.w3.org/TR/xsl11/> for details)

Configuring an output target means providing RPE with all the information needed to produce the output.

NOTE Not all output formats have the same set of capabilities. For a detailed description of formatting capabilities check the Palette Reference Guide.

Each type of output format has different properties that need to be configured. See the Output section for details on specific properties.

Property	Value
type	Word
driver	Telelogic.Word.Driver
path	
stylesheet	
macro	

Figure 4 Word output properties

The properties shared by all output formats are listed below:

Property	Description
Type	The output type. Can be Word, HTML, PDF and XSL-FO. Read-only
Driver	The output driver used to create the output. Read-only
Path	The location of the output file. If empty RPE will generate a unique name in the user's temp folder.

Workflow

The steps needed to produce a document from a supported data source are:

1. Create a document template
 - Create a new document template using RPE Document Studio
 - Define the schema for your data source. For some data types, such as DOORS, RPE provides assistance for building the schema.
 - Insert the data source schema into the template
 - Define the template's content
 - Save the template
2. Generate the output
 - Create a document specification in Launcher or Document Studio
 - Add a template previously defined to the document specification. RPE will detect the data source schemas used by the template and will generate the required fields in the Document Specification that need to be completed with information about the actual data source
 - Configure the actual data sources by filling in the required fields
 - (optional) Provide values for the template variables as needed
 - (optional) Configure the output formats required by filling in the required fields

Publishing documents with RPE from a data source

In this scenario, a template or document specification can be selected via wizards in the data source application.

The wizard takes the user through a series of steps collecting required information such as actual data sources, output formats etc. In many cases, the selection of actual data sources is sensitive to context and automated (e.g. when running from a DOORS Formal Module, the current Module is automatically selected as the actual data source).

Data

RPE currently supports DOORS, Tau, generic XML data sources and Rational sources exposing the Rational REST Get Specification.

Schema display

The data source schemas are used in the Document Studio. When displaying a data source schema, RPE will use the "label" annotation if present. As such it is possible that the names displayed in RPE's browser.

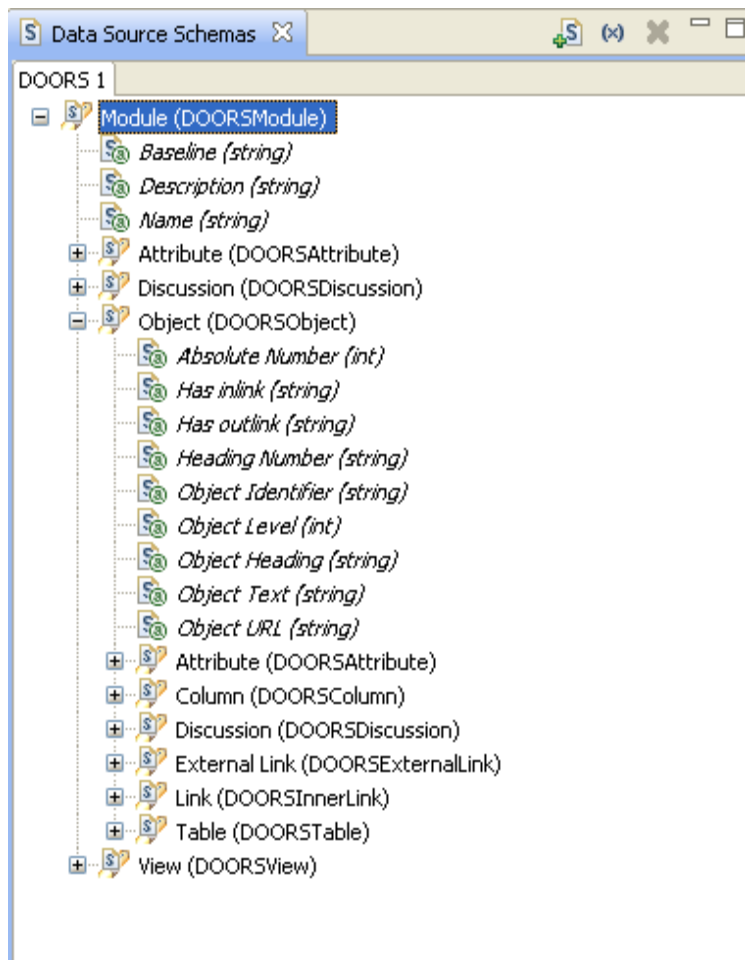


Figure 5 Annotated DOORS Schema view

You can configure RPE to show the raw schema from the Document Studio preference page. The above schema would look as in the picture below. Note the names of the attributes and elements.

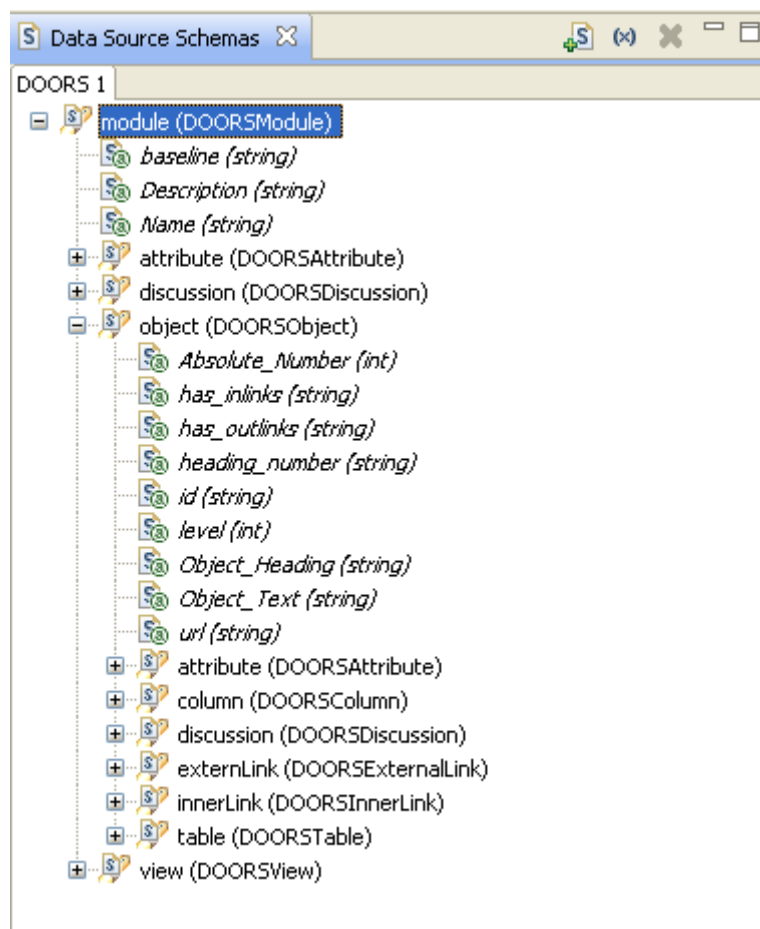


Figure 6 Raw DOORS schema view

Queries and contexts

As previously mentioned, a RPE template specifies the data to be extracted using *queries*. A *query* is a path in the data source schema. Ex. Module.Object.Attribute.Name. This path starts with the root element (module), continues with its child element Object than with the Object's child attribute and so on.

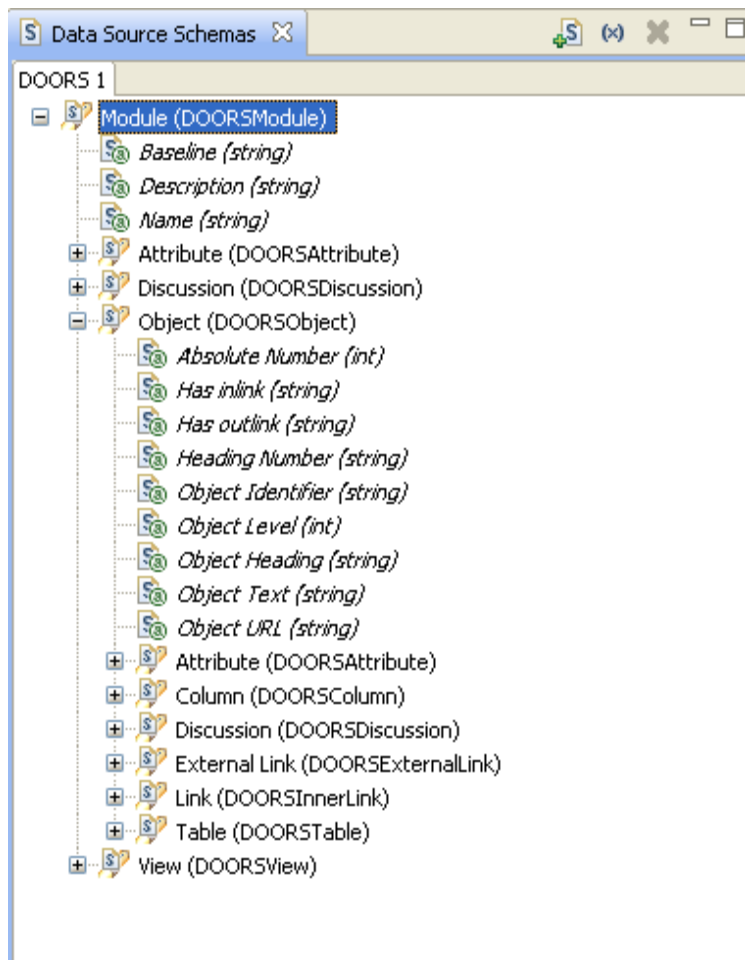


Figure 7 Sample DOORS Data Schema (annotated view)

NOTE For easier identification attributes are rendered in italic.

The Document Studio shields users from much of the complexity of manually writing queries with features such as drag and drop of schema elements. Nevertheless, it is still useful for template authors to understand the concepts of schema and queries and how they are constructed.

In the above data source schema some valid queries include:

Query	Description
module	returns a single result, the source module
module.object	Returns all the objects in the source module, as filtered/ sorted by the source view

module.object.attribute	If used in a module.object context returns all the attributes for the current object.
-------------------------	---

A query is attached to a template element. The template element and all its children can use the attributes of the entities returned by the current query and the attributes of all the queries from parent elements.

In the above example, once the query module.object is applied, then any of the Object's schema element attributes can be used: Absolute Number, Has Inlink, Object Text etc

RPE template elements can be nested. Setting queries on elements and their children elements creates nested contexts. The query in the child element will be performed in the context of the parent's element query results.

Example:

Element 1: module.object

Element 1.1 (child of Element 1): module.object.attribute

The second query will yield a list of attributes for the current object returned by the query of element 1. In element 1 only the attributes of DOORS Objects can be used while in element 1.1 the attributes of DOORS Object attributes can be used (i.e. the names of those Object attributes).

NOTE RPE Studio fully assists the user in building the queries and assigning the appropriate contexts. At no time you will be required to manually type a query.

Filtering data

There are situations when not all the data is needed. In these cases you can limit the amount of processed data by setting a *filter* on the query. There are two ways for specifying a filter:

- *RPE Filter* – JavaScript expression using the data attributes of the entities returned by the query
- *Native filter* – plain text in a format specific to each data source type.

When the query is performed, only the data entities matching the filter will be included in the output.

NOTE Not all data source types support native filtering.

NOTE For data sources that support native filtering it is mandatory for the native filter to be a valid as defined by the data source. RPE cannot and will not perform any validation on the native filter. Providing an invalid native filter can have results ranging from incorrect data in the output to the tool crashing.

NOTE For data sources accessed through RPE's Generic XML input driver it is not possible to define native filtering. The only exception to the rule is for the data sources where the filtering can be specified in the URL.

TIP Native filters are usually more effective and yield faster document generation times than RPE filters. Whenever possible try using native filters.

Sorting data

Data can be sorted in RPE. Sorts can be specified in two ways:

- *RPE Sort* – consists of the list of attributes and the sort direction (ascending/ descending)
- *Native sort* – plain text in a format specific to each data source type. For DOORS this text must be in the format of the DOORS Sort

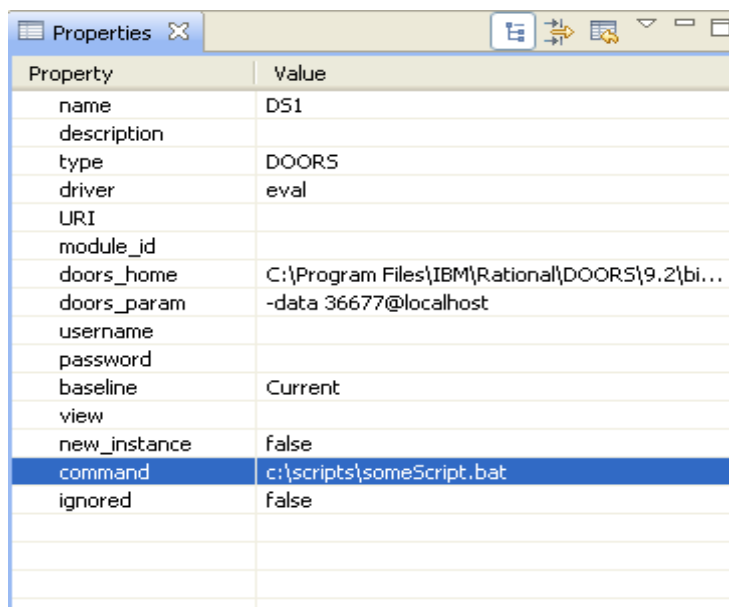
When the query is performed, the elements will be inserted sorted in the output document.

NOTE Not all data sources support native sorting.

TIP Native sorts are usually more effective and yield faster document generation times than RPE sorts. Whenever possible try using native sorts.

Data Source pre-processing

With version 1.1 RPE offers the possibility to preprocess the data source. The pre-processing mechanism is generic and consists of a command that will be executed by RPE before data is retrieved from the data source. The command is set in the Document Specification. RPE waits for the command to complete but will not check its return code or handle errors in any way.



Property	Value
name	DS1
description	
type	DOORS
driver	eval
URI	
module_id	
doors_home	C:\Program Files\IBM\Rational\DOORS\9.2\bi...
doors_param	-data 36677@localhost
username	
password	
baseline	Current
view	
new_instance	false
command	c:\scripts\someScript.bat
ignored	false

Figure 8 Pre-processing command

Dynamic Data sources

With RPE 1.1 it is possible to configure/reconfigure a data source at runtime. (Re)Configuring a data source means changing the properties used by RPE when extracting data: URI, user name, password and any other data source specific properties.

The (re)configuration of a data source is done through a new template element named “Dynamic Datasource Configuration”. When the RPE core encounters this element during document generation it will initialize/reinitialize the data source according to the information provided.

NOTE The Dynamic Data Source element has, depending on the data source it configures, the exact same properties as the ones displayed in the Document Specification.

NOTE A “Data Source Configuration” can be seen as a Document Specification element included in a Document Template.

Defining a “Data Source Configuration”

A “Data Source Configuration” element is used just as any other template element. It has to be dragged in the template area.

NOTE Data Source Configuration elements cannot be used in master pages.

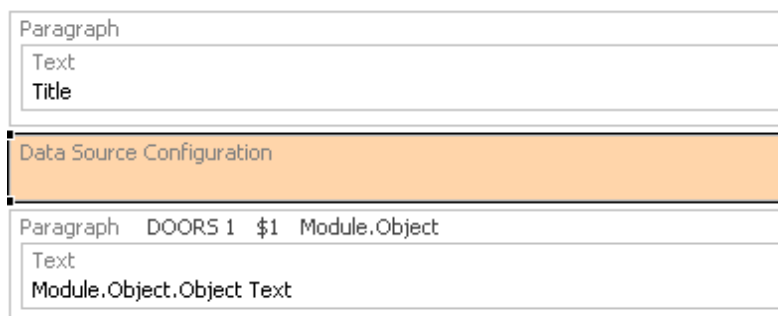


Figure 9 Data Source Configuration

A Data Source Configuration by itself has no value. To serve its purpose the Data Source Configuration must be connected to a Data Source Schema used in the template. The connection is done by dragging the Data Source Schema from the Outline View over the Data Source Configuration or by using the properties view.

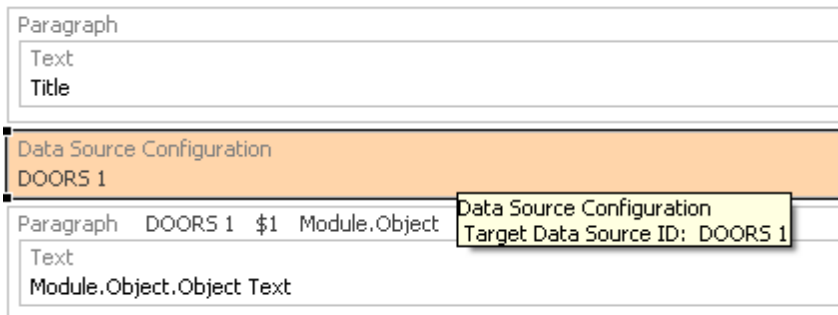
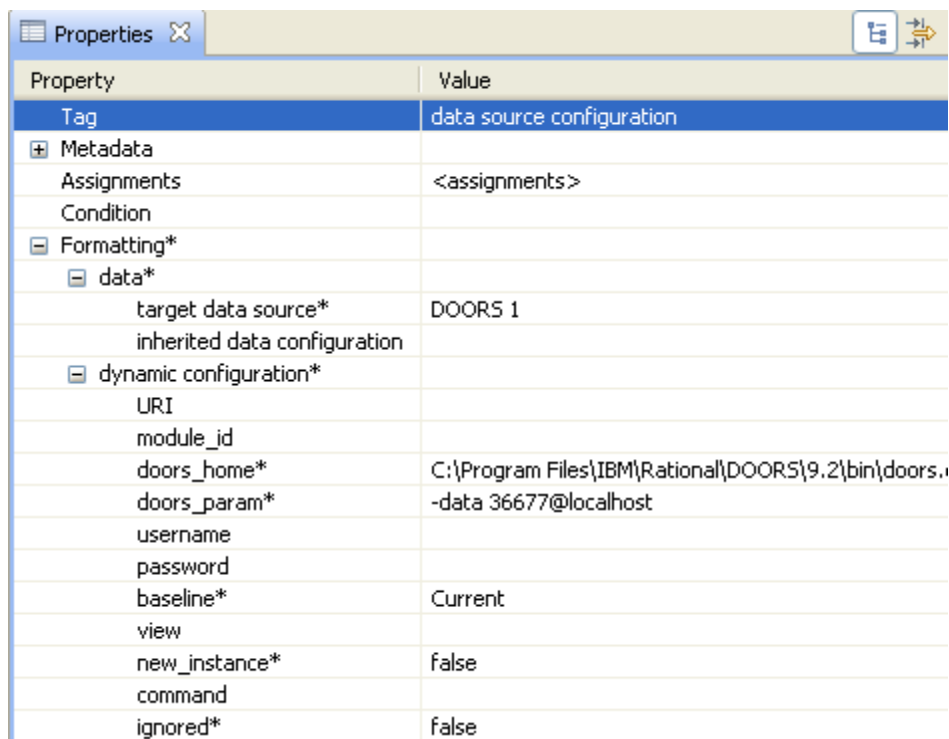


Figure 10 A Data Source Configuration element connected to a data source

Property	Value
Tag	data source configuration
Metadata	
Assignments	<assignments>
Condition	
Formatting*	
data*	
target data source*	DOORS 1
inherited data configuratio	
dynamic configuration*	

Figure 11

Once a Data Source Configuration is connected to a Data Source Schema, its formatting properties are updated to reflect the runtime properties of the Data Source Schema type.



Property	Value
Tag	data source configuration
Metadata	
Assignments	<assignments>
Condition	
Formatting*	
data*	
target data source*	DOORS 1
inherited data configuration	
dynamic configuration*	
URI	
module_id	
doors_home*	C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors.i
doors_param*	-data 36677@localhost
username	
password	
baseline*	Current
view	
new_instance*	false
command	
ignored*	false

Figure 12 Data Source Configuration connected to a DOORS Schema

As seen in the above image the properties depend on the type of the schema.

Dynamic data

A “Data Source Configuration” element has the same properties as the Data Source element in Document Specification with one major difference: the properties from the Document Specification have static values while the properties from the “Data Source Configuration” element can use any data available in their context. In other words you can configure a data source using values read from another data source.

NOTE For an example please consult the *RPE_DOORSMultiModuleGeneration* template provided with the installation. In that example a DOORS Data source is configured using module URI, view and version read from an XML data source.

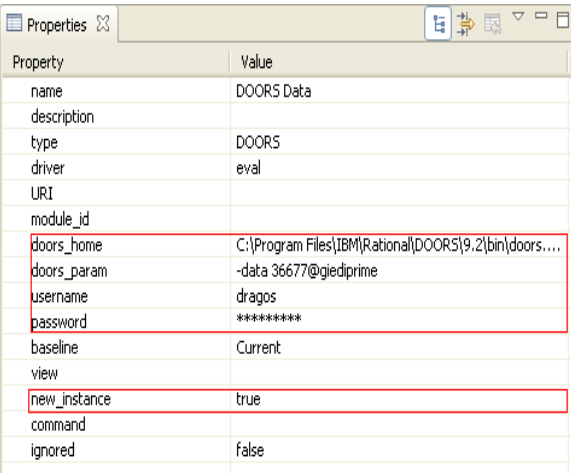
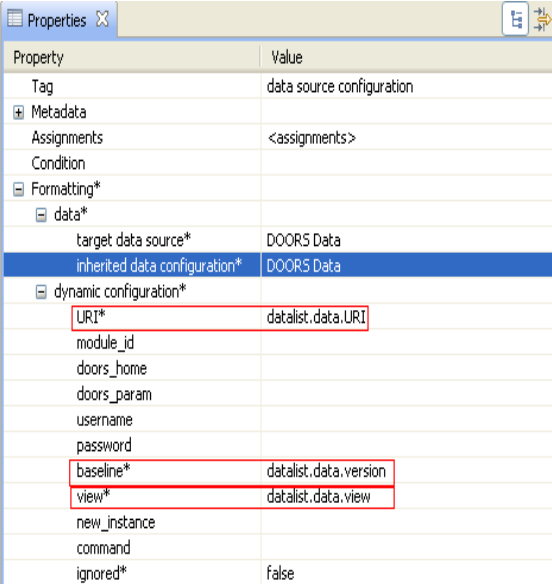
Inherited Data Configuration

Besides the “target data source” a configuration element has a property named “inherited data source”. The value for that property is empty by default but can be changed to the name of any Data Source Schema used in the template.

If this property is used, RPE will try to configure the “target data source” using the settings taken from the “inherited data source” combined with any properties defined in the “Data Source Configuration”.

The purpose of this property is to avoid having to enter all the configuration properties in the “Data Source Configuration” element. For example one could only specify the URI property and let the other properties (such as DOORS home, user name and password, new_instance etc) to be inherited from another DOORS Source used in the template.

NOTE The “target data source” and “inherited data source” can be one and the same data source. RPE will combine the properties defined in the Data Source Configuration element with those defined in the Document Specification in the following manner: unless the data source configuration property is empty it takes precedence over the document specification property See the example below.

Document Specification Properties	Data Source Configuration Properties																																																																										
 <table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>name</td><td>DOORS Data</td></tr> <tr><td>description</td><td></td></tr> <tr><td>type</td><td>DOORS</td></tr> <tr><td>driver</td><td>eval</td></tr> <tr><td>URI</td><td></td></tr> <tr><td>module_id</td><td></td></tr> <tr><td>doors_home</td><td>C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors....</td></tr> <tr><td>doors_param</td><td>-data 36677@giediprime</td></tr> <tr><td>username</td><td>dragos</td></tr> <tr><td>password</td><td>*****</td></tr> <tr><td>baseline</td><td>Current</td></tr> <tr><td>view</td><td></td></tr> <tr><td>new_instance</td><td>true</td></tr> <tr><td>command</td><td></td></tr> <tr><td>ignored</td><td>false</td></tr> </tbody> </table>	Property	Value	name	DOORS Data	description		type	DOORS	driver	eval	URI		module_id		doors_home	C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors....	doors_param	-data 36677@giediprime	username	dragos	password	*****	baseline	Current	view		new_instance	true	command		ignored	false	 <table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>Tag</td><td>data source configuration</td></tr> <tr><td>Metadata</td><td></td></tr> <tr><td>Assignments</td><td><assignments></td></tr> <tr><td>Condition</td><td></td></tr> <tr><td>Formatting*</td><td></td></tr> <tr><td>data*</td><td></td></tr> <tr><td>target data source*</td><td>DOORS Data</td></tr> <tr><td>inherited data configuration*</td><td>DOORS Data</td></tr> <tr><td>dynamic configuration*</td><td></td></tr> <tr><td>URI*</td><td>datalist.data.URI</td></tr> <tr><td>module_id</td><td></td></tr> <tr><td>doors_home</td><td></td></tr> <tr><td>doors_param</td><td></td></tr> <tr><td>username</td><td></td></tr> <tr><td>password</td><td></td></tr> <tr><td>baseline*</td><td>datalist.data.version</td></tr> <tr><td>view*</td><td>datalist.data.view</td></tr> <tr><td>new_instance</td><td></td></tr> <tr><td>command</td><td></td></tr> <tr><td>ignored*</td><td>false</td></tr> </tbody> </table>	Property	Value	Tag	data source configuration	Metadata		Assignments	<assignments>	Condition		Formatting*		data*		target data source*	DOORS Data	inherited data configuration*	DOORS Data	dynamic configuration*		URI*	datalist.data.URI	module_id		doors_home		doors_param		username		password		baseline*	datalist.data.version	view*	datalist.data.view	new_instance		command		ignored*	false
Property	Value																																																																										
name	DOORS Data																																																																										
description																																																																											
type	DOORS																																																																										
driver	eval																																																																										
URI																																																																											
module_id																																																																											
doors_home	C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors....																																																																										
doors_param	-data 36677@giediprime																																																																										
username	dragos																																																																										
password	*****																																																																										
baseline	Current																																																																										
view																																																																											
new_instance	true																																																																										
command																																																																											
ignored	false																																																																										
Property	Value																																																																										
Tag	data source configuration																																																																										
Metadata																																																																											
Assignments	<assignments>																																																																										
Condition																																																																											
Formatting*																																																																											
data*																																																																											
target data source*	DOORS Data																																																																										
inherited data configuration*	DOORS Data																																																																										
dynamic configuration*																																																																											
URI*	datalist.data.URI																																																																										
module_id																																																																											
doors_home																																																																											
doors_param																																																																											
username																																																																											
password																																																																											
baseline*	datalist.data.version																																																																										
view*	datalist.data.view																																																																										
new_instance																																																																											
command																																																																											
ignored*	false																																																																										

In the above images the properties marked with red are the ones used. The Data Source Configuration element defines the URI, baseline and view while the Document Specification entry defines the DOORS DB to use, user name, password etc.

NOTE It is recommended to use the “Data Source Configuration” element to define only properties that are not known at runtime. Using the “Data Source Configuration” element for all properties, including user names and passwords, location of exe files etc will make the templates less portable.

Template flow

The “Data Source Configuration” element must be placed in the document template before any element containing queries for the configured data source.

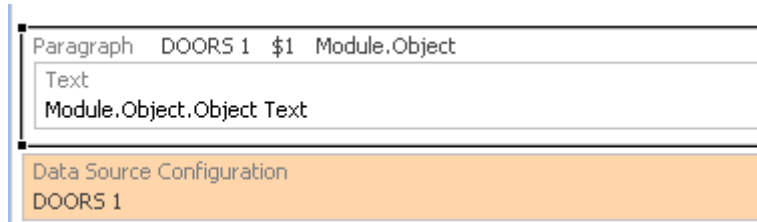


Figure 13

The paragraph in the above figure is not affected by the Data Source Configuration element. The paragraph will use “DOORS 1” data source with the existing settings.

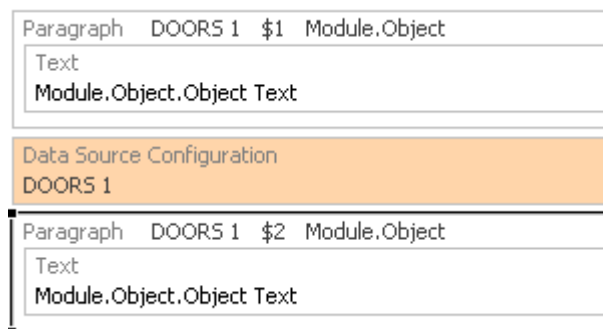


Figure 14

The second paragraph will use DOORS 1 with the properties defined by the Data Source Configuration element.

DOORS Data

A **concrete** DOORS Data source is defined by a **view** from a **version** (i.e. Current version or a baseline) of a DOORS **module**. In this context the View is only used to define the filtered and sorted subset of data to use. If no view is specified, the default view for the DOORS user* is used.

The DOORS user is the user whose credentials are used to log into DOORS when a headless DOORS instance is used (*new_instance=true*) or the user currently logged in DOORS for when an existing DOORS instance is used (*new_instance=false*).

RPE can extract data from a DOORS database as long as a DOORS 9.1 Client or later version is installed on the same machine. DOORS data can be extracted in two ways:

- Using a headless DOORS client run in the background

- Using an already running DOORS instance

The first method has the advantage of allowing unhindered use of any already running DOORS instances. The second method is slightly faster as the overhead of starting DOORS does not exist. The run mode is specified in the Document Specification using the RPE Launcher, by setting the *new_instance* property for each DOORS data source defined in the template.

NOTE RPE opens all the required modules in read-only access mode.

NOTE The data is extracted using DOORS DXL. On average the DXL execution time accounts for 50% -90% of the document generation time.

NOTE If the interactive run mode (using an existing DOORS instance) is set for a DOORS data source and no DOORS instance is running the data extraction will fail for that data source.

Configuring the data source

When a DOORS data source is present in a Document template you need to define the following properties for the concrete data source:

Property	Description	Interactive DOORS Client	Headless DOORS Client
URI	The absolute path of the DOORS module in the database. Case sensitive	required	required
module_id	The module's unique ID. Used if the URI is not specified, ignored otherwise.	optional	optional
doors_home	The absolute file path of doors.exe	not used	required
doors_param	The database to connect to and any other valid DOORS command line switch. Default: <i>-data 36677@localhost</i>	not used	required
username	The DOORS account name to use for data extraction	not used	optional
password	The DOORS account password (encrypted)	not used	optional
baseline	The module version to use. Case sensitive Default: <i>Current</i>	required	optional

view	The view to use for filtering/sorting. Case sensitive Default: <i>empty</i> . Will use the logged user's default view for the module.	required	optional
command	Pre-processing instructions.	optional	optional
new_instance	If set to true a headless DOORS client is used otherwise RPE will attempt to use an existing DOORS instance. Values: <i>true/false</i> Default: <i>true</i> NOTE if multiple DOORS Instances are available, the DOORS client which had been open the longest is used.	-	-

NOTE Providing an incorrect value for any field marked as required (except *view* and *baseline*) will result in the output not being generated.

Providing an incorrect value *view* or *baseline* will result in the output being generated from the default view of the current module version.

Configuration Wizard

A DOORS data source can be configured using the “Configure data source” function in launcher:

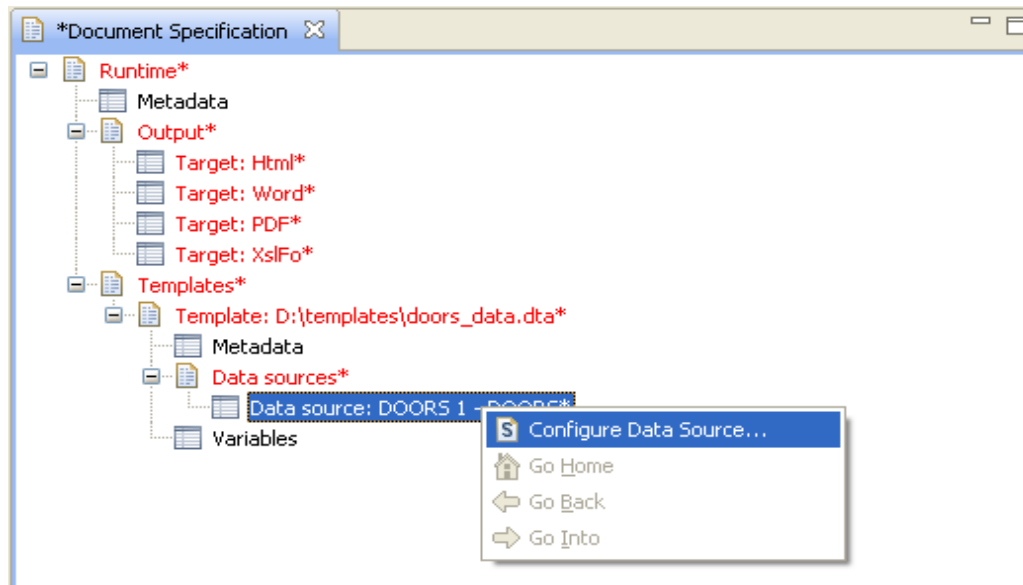


Figure 15 Configure data source

If this function is used RPE will display a wizard that allows selecting the DOORS parameters:

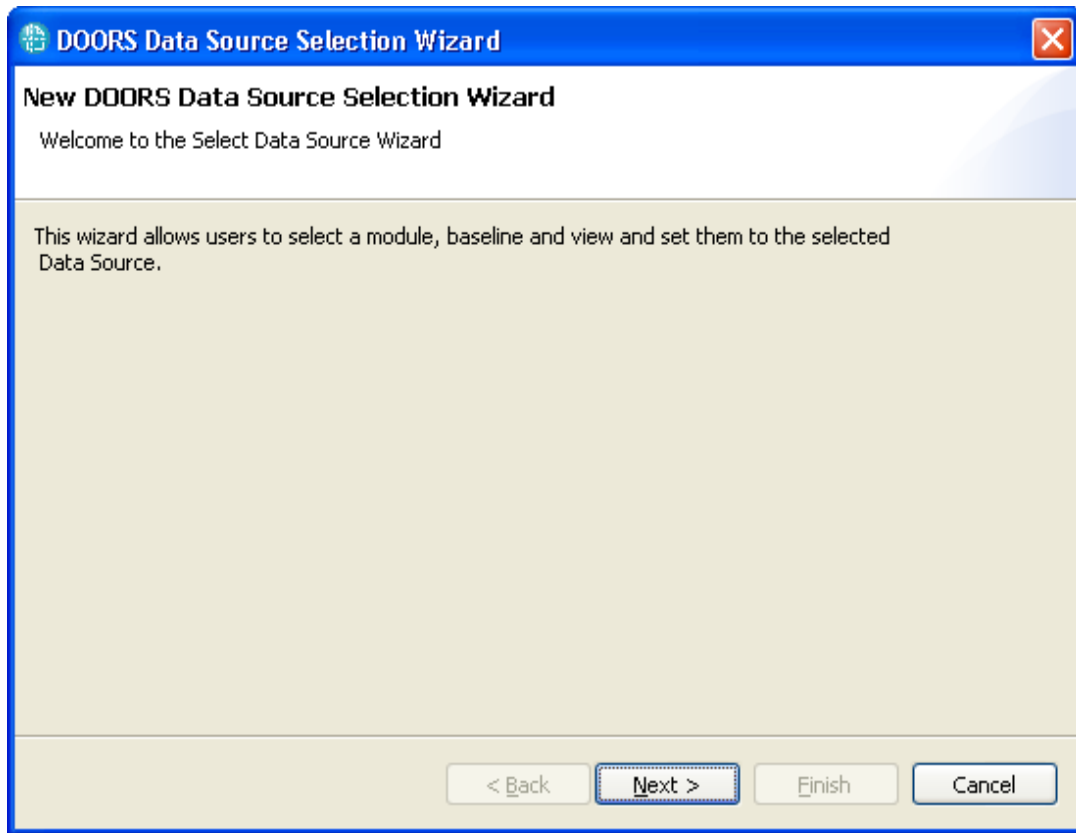


Figure 16 DOORS Configuration Wizard - Welcome screen

In the next step of the wizard you need to provide information to RPE about how to connect to DOORS.

The screenshot shows a Windows-style dialog box titled "DOORS Data Source Selection Wizard" with a close button in the top right corner. The main heading is "DOORS Connection options" followed by the instruction "Fill the information below". Below this, there are two radio button options: "Use running DOORS instance" (which is selected) and "Run a new background DOORS process". Underneath are four text input fields: "Username" containing "spurlos", "Password" (empty), "Database" containing "36677@giediprime", and "Path to doors.exe" containing "C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors.exe" with a "Browse..." button to its right. At the bottom of the dialog are four buttons: "< Back", "Next >" (highlighted with a dashed border), "Finish", and "Cancel".

Figure 17 Configuration Wizard - Connection options

The preferred (fastest) way to do this is via an existing DOORS instance.

If you specified the correct arguments to the wizard you are taken to a screen displaying the top level structure of the DOORS Database.

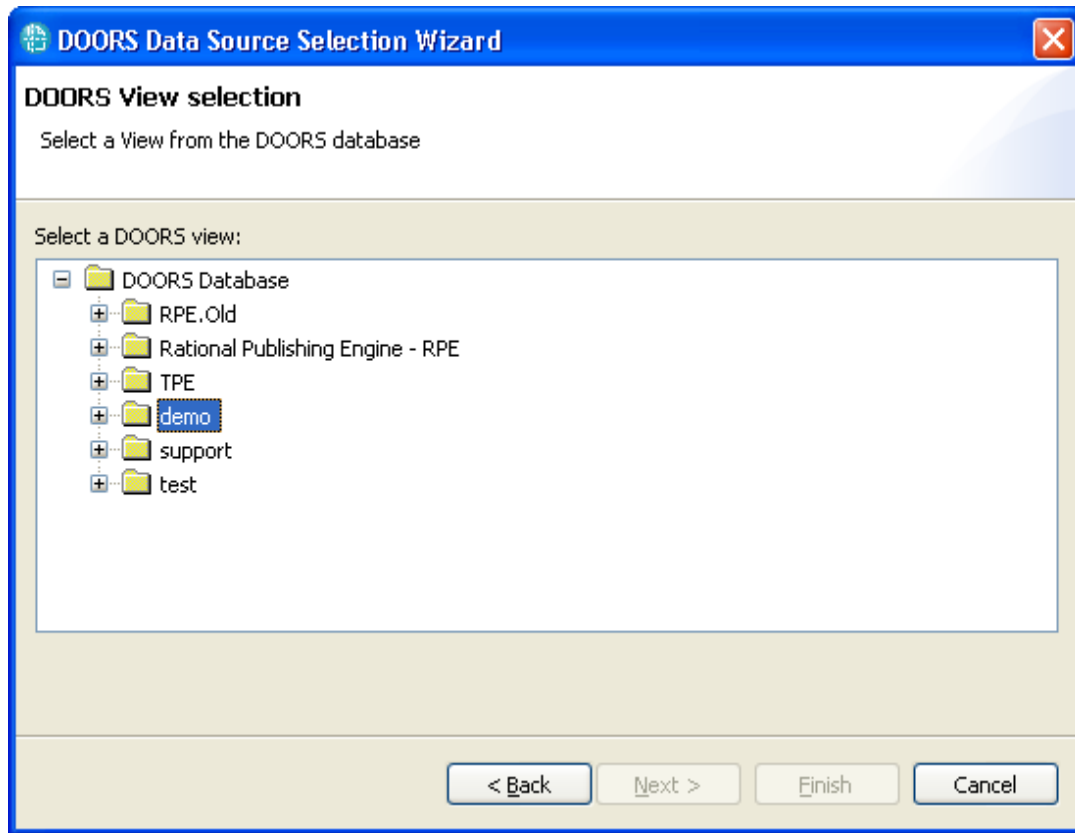


Figure 18 Database structure

At this point you have to navigate this structure and select a module – version- view.

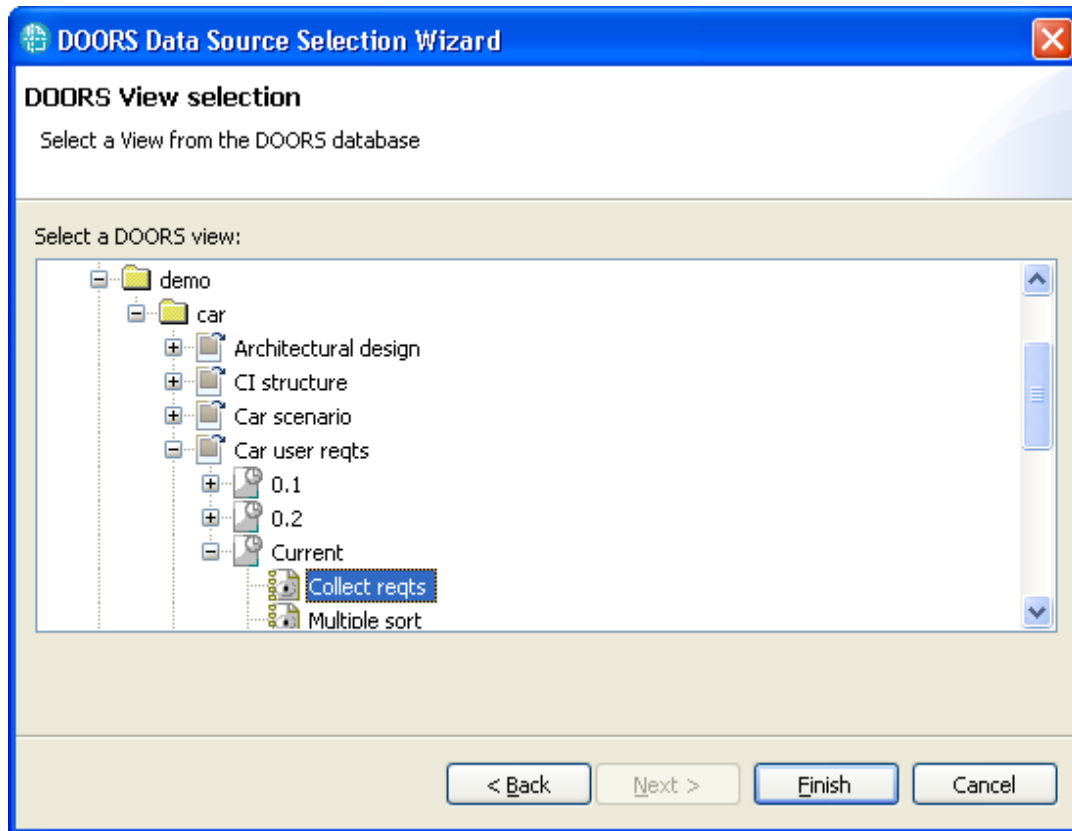
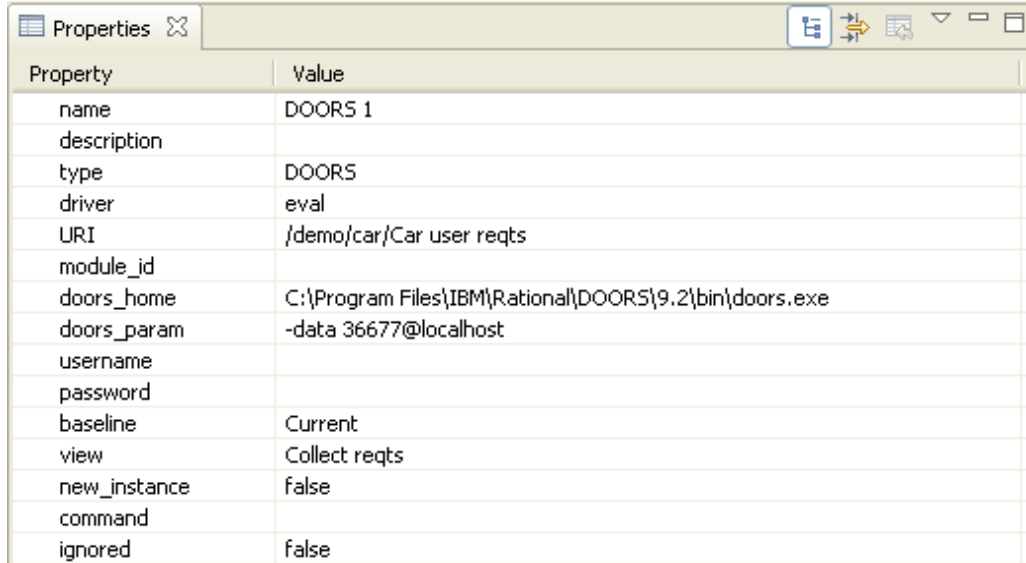


Figure 19 selecting the module - version – view

Only when you have selected a view will the Finish button activate.

The information you have provided is used to configure the data source:



Property	Value
name	DOORS 1
description	
type	DOORS
driver	eval
URI	/demo/car/Car user reqts
module_id	
doors_home	C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors.exe
doors_param	-data 36677@localhost
username	
password	
baseline	Current
view	Collect reqts
new_instance	false
command	
ignored	false

Figure 20 Configured DOORS Data Source

NOTE RPE will use all the information you have entered excepting the user name and password you used. You need to enter those values manually in the property page.

DOORS Schema

The DOORS schema was designed to be simple to use and to match closely the DOORS module structure. RPE comes with a predefined DOORS schema that is generic and valid for all DOORS Formal Modules. New DOORS schemas can be created using the “Schema Discovery” wizard in RPE Studio.

NOTE The predefined DOORS schema is most suited when you do not need to extract particular user defined attributes from DOORS but rather the whole content of a view or just predefined attributes. When particular user defined attributes are required it is recommended to use the Schema Discovery wizard to generate a schema on your specific data.

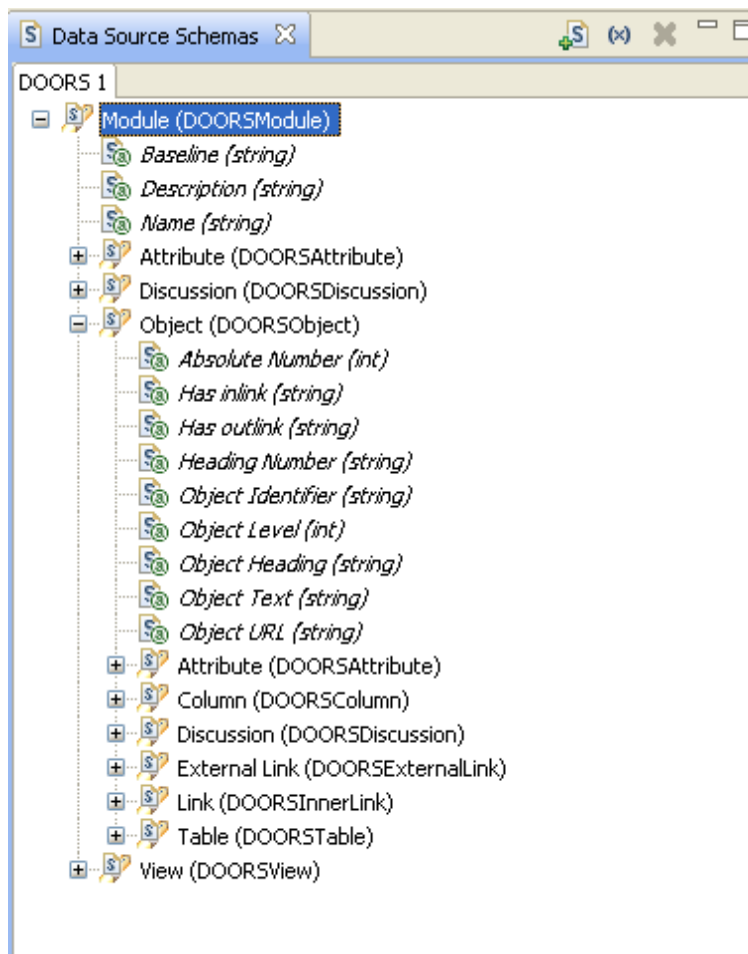


Figure 21 Predefined DOORS Schema

Queries and attributes

The following queries and attributes are from the default DOORS schema provided by RPE. User defined schemas will contain additional queries and/or attributes.

Query	Results	Attributes
Module	a single entity, the Module specified in the data source configuration	<ul style="list-style-type: none"> • <i>Name</i> - the Module's name • <i>Description</i> – the Module's description • <i>Baseline</i> – the Module version (baseline) used
Module.Attribute	the list of Module level attributes for the current Module	<ul style="list-style-type: none"> • <i>Name</i> – the attribute's name • <i>_value</i> – the attribute's value
Module.Discussion	The list of discussions for the module.	
Module.Discussion.Attribute	The list of attributes for a discussion	
Module.Discussion.Comment	The list of comments for a discussion	
Module.View	a single result, the View defined for each DOORS data source in the document specification	<ul style="list-style-type: none"> • <i>Name</i> – the name of the View
Module.View.Column	<p>the list of columns for the selected View</p> <p>NOTE The purpose of the <i>module.view.column</i> query is to provide a list of the column names without having to iterate the Module Objects. The result does not contain column data.</p>	<ul style="list-style-type: none"> • <i>Name</i> – the name of the column • <i>_value</i> – empty

Module.Object	the list of all Objects of the specified version of the current Module's as filtered/sorted by the selected View	<ul style="list-style-type: none"> • Object Identifier • Object Text • Object Heading • Absolute Number • Object Level • Any attribute elevated by the user in the schema discovery wizard
Module.Object.Attribute	the list of attributes for the current Object if this query is in the context of a <i>module.object</i> query, or the list of all attributes for all Objects in the Module	<ul style="list-style-type: none"> • <i>Name</i>: the attribute's name • <i>_value</i>: the attribute's value
Module.Object.Column	the list of columns in the selected View for the current Object	<ul style="list-style-type: none"> • <i>Name</i>: the column's name • <i>_value</i>: the column's value for the current Object
Module.Object.Table	<ul style="list-style-type: none"> • no results if the current Object is not a DOORS table • a single result, (the DOORS table) if the Object is a table header Object 	none
Module.Object.Table.Row	the current table's rows	none
Module.Object.Table.Row.object	A collection of Objects; the current rows' cells. Same attribute list available as for the <i>module.object</i> query	<ul style="list-style-type: none"> • Object Identifier • Object Text • Object Heading • Absolute Number • Object Level
Module.Object.External Link	A collection of External links for the current object.	<ul style="list-style-type: none"> • URL – the url of the external linked entity <p>Other attributes as defined in DOORS.</p>
Module.Object.Link	A list of DOORS Links for the current object	
Module.Object.Link.Attribute	The attribute list for the current link	

Module.Object.Link.Linke d Object	The link at the other end of the link.	
--------------------------------------	--	--

Images

Images are extracted with the attribute’s value. You do not have to nor can you query just for the images from a DOORS module.

TIP You can configure is the size of the extracted images. The max size is specified through the “image max width” and “image max height” properties. These properties can be specified in two places:

- *element format info*– defines the images min/max size for the images contained in that template element. Available for text template elements only

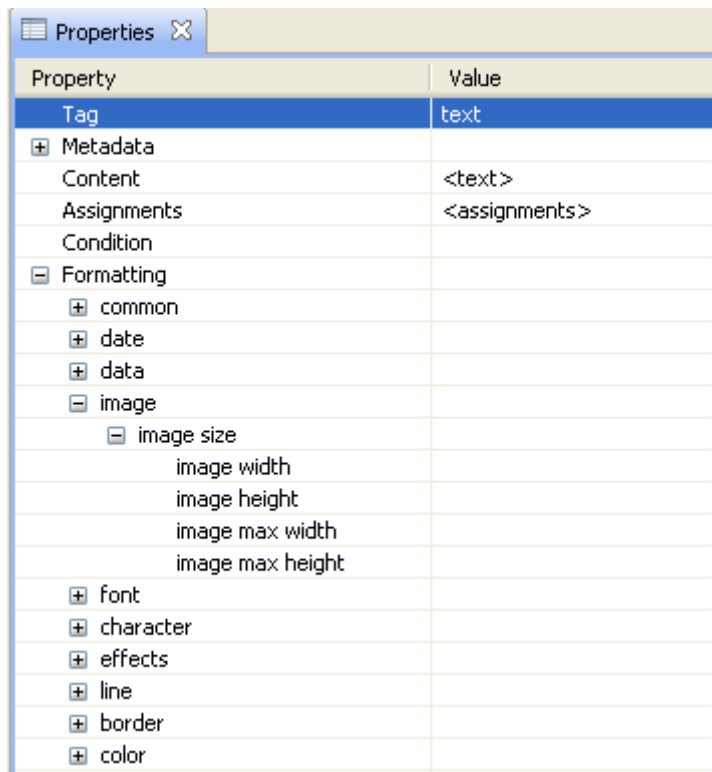
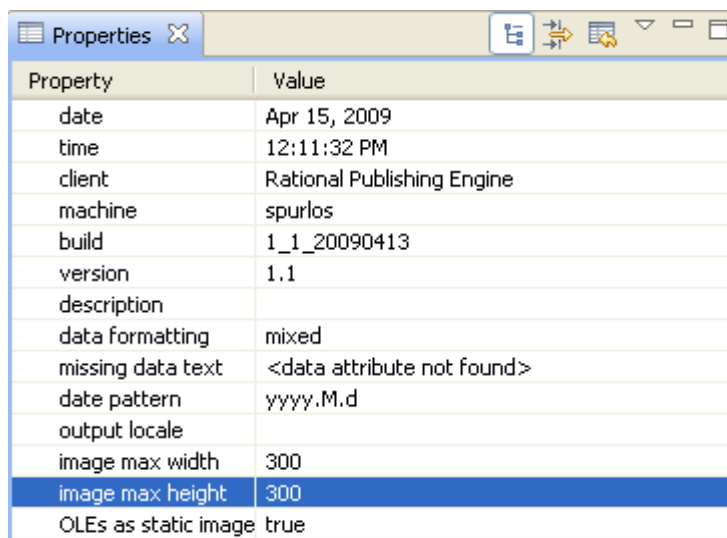


Figure 22 Image related formatting properties in text elements

- *document specification metadata* - defines the image’s min/max size for all the images in all the templates. The element level values override these global values.



Property	Value
date	Apr 15, 2009
time	12:11:32 PM
client	Rational Publishing Engine
machine	spurlos
build	1_1_20090413
version	1.1
description	
data formatting	mixed
missing data text	<data attribute not found>
date pattern	yyyy.M.d
output locale	
image max width	300
image max height	300
OLEs as static image	true

Figure 23 Image formatting options in metadata

OLEs

RPE can extract OLEs from a DOORS data source. How OLEs are displayed in the output documents depends on the output type and the options selected in the document specification.

OLEs will always be rendered as images in HTML, PDF and XSL-FO output as those formats do not support OLEs.

For Word output the OLEs will be rendered as static images or as OLEs depending on the “OLEs as static images” flag in the metadata section of the document specification.

If “*OLEs as static images*” is set to TRUE, OLEs will be included in the output document as static images.

If “*OLEs as static images*” is set to FALSE, RPE will generate a “ref” folder in the same location as the Word output document. The ref folder contains RTF files for the OLE objects in the DOORS data source. The Word output will have one include field pointing to a RTF file for each OLE exported from DOORS.

NOTE RPE cannot update Microsoft Word fields. As a consequence the include fields will not be visible when you open the generated Word document. To make the fields visible you need to take one of the following actions:

Action	Result
select the entire document content and use the “Update fields” function in Word	The OLEs are displayed in the document. The document is not self-contained.

use the “updateFields” macro provided by RPE	The OLEs are displayed in the document. The document is not self-contained.
use the “insertOLEs” macro provided by RPE	The OLEs are displayed in the document. The document is self-contained.

NOTE Updating the fields in the document will not make the Word document self contained. This means that moving such a Word document from the machine it was generated on will prevent editing the OLEs. To make the document self contained you need to run the “rpe” or “insertOLEs” macros.

Tables

You need to explicitly query for DOORS tables as they are not extracted automatically from DOORS. While manually adding the queries for extracting a table requires extra effort when building the template, this approach has the advantage of allowing fine grained control over the formatting of the table.

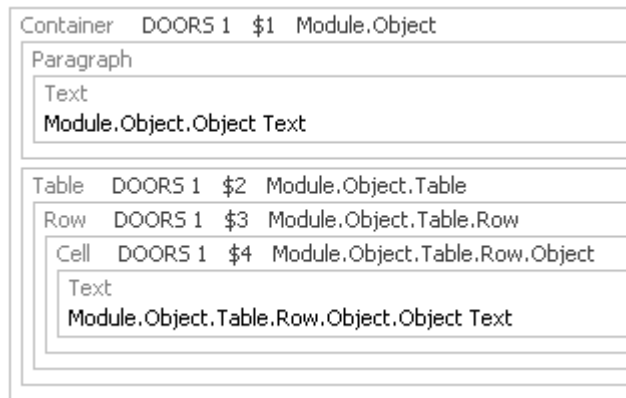


Figure 24 Extracting tables from DOORS

TIP The table will only get created for those DOORS objects that start a table.

NOTE The cells of a DOORS table do not have a dedicated type in the DOORS schema. The content of the cells can be retrieved through the *Module.Object.Table.Row.Object* query.

NOTE In the case of objects that are DOORS Table cells their *Object Text* attribute is a combination between *Object Heading* and *Object Text*.

DOORS External Links

With RPE 1.1 you can obtain the information related to DOORS External Links. The query to use is *Module.Object.External Link*.

The information that can be obtained for the external hyperlinks consists of the URL of the external entity and a set of other DOORS attributes defined for the external link.

DOORS Links

With RPE 1.1 you can obtain the information related to DOORS Inner Links. The query to use is `Module.Object.Link`. This query will return the object links and not the linked objects. To obtain the linked objects the `Module.Object.Link.Linked Object` must be used.

Recursive retrieval

RPE can follow DOORS links recursively. This is achieved by setting the “Recursive” value to a non 0 value.

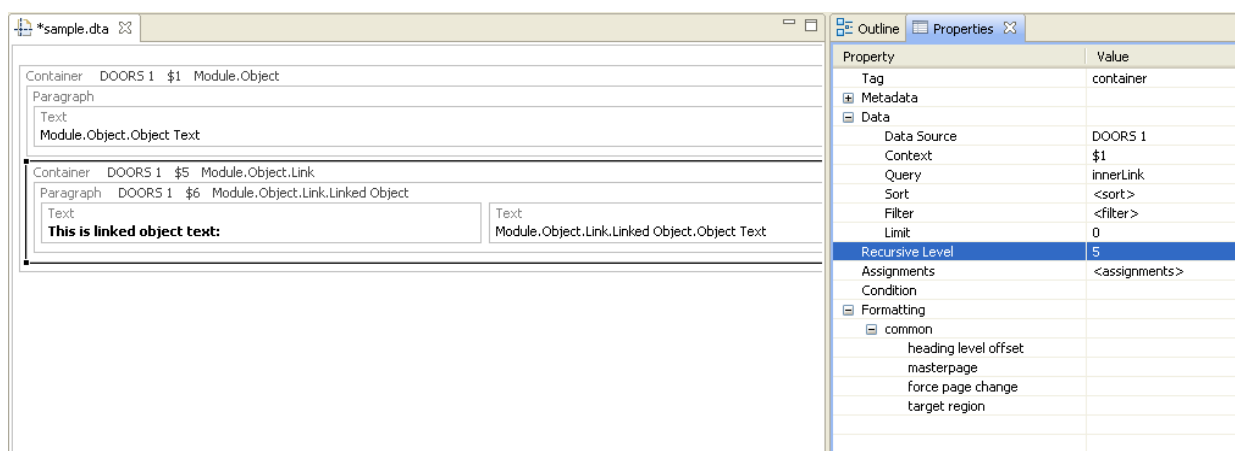


Figure 25 Recursive retrieval of DOORS Links

What happens is that RPE will try to follow links from the initial objects and from all its linked objects and so on up to the specified recursive level.

NOTE Following links in DOORS module is a time consuming process. As each object can have many links the number of objects to be processed can increase exponentially with each level. Filtering links using the native filter can dramatically improve the time required to follow links.

Filtering

RPE allows native filters to be defined on link queries. Using these filters can greatly reduce the number of links that need to be processed and thus reducing the time needed to generate the document.

Filtering by direction

The syntax for filtering all the in links or out links is:

Link direction in|out

NOTE The filter is case sensitive

Filtering by link module

The syntax for filtering all the links going through a particular link module is:

Link module <link module name>

In the above filter <link module name> is the full name of the link module.

Example:

Link module /Demo/Car/Link Module 1

NOTE The filter is case sensitive

Filtering by target module

The syntax for filtering all the links to or from a specified module is:

Link analyze module <target module name>

In the above filter <target module name> is the full name of the target module.

Example:

Link analyze module /Demo/Car/System Requirements

NOTE The filter is case sensitive

Combining filters

The above filters can be combined. Each filter must be on its own line.

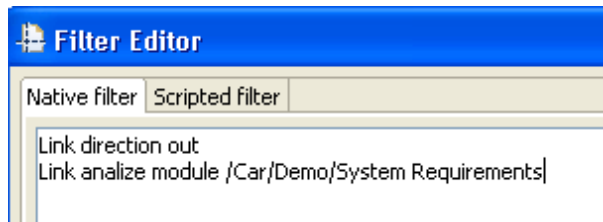


Figure 26 Native link filtering

What cannot be extracted

RPE does not provide the means to query the *module's baseline list* or the *module's list of views*.

NOTE A DOORS Data Source in RPE is defined by the <module, baseline, view>. Extracting data from more than one module or baseline or view can be done in three ways:

- Define more than one DOORS Data Source in the template and add the corresponding template elements
- Add the same template multiple times to the document specification and configure the data sources for each template instance to the desired <module, baseline, view>
- Add multiple templates to the document specification and configure the data sources for each template instance to the desired <module, baseline, view>

DOORS Database Structure

This is a new data source type introduced in RPE 1.1. A DOORS Database Structure is defined by a DOORS Database. You need a DOORS 9.1 or later client software in order to use it for document generation.

Configuring the data source

Configuring a DOORS Database Structure data source requires no user action if an existing DOORS Instance is running. If no DOORS instance is running or you want to use a different data source you need to provide the following parameters.

Property	Description	Interactive DOORS Client	Headless DOORS Client
doors_home	The absolute file path of doors.exe	not used	required
doors_param	The database to connect to and any other valid DOORS command line switch. Default: <i>-data 36677@localhost</i>	not used	required
username	The DOORS account name to use for data extraction	not used	optional
password	The DOORS account password (encrypted)	not used	optional
command	Pre-processing instructions.	optional	Optional
new_instance	If set to true a headless DOORS client is used otherwise RPE will attempt to use an existing DOORS instance. Values: <i>true/false</i> Default: <i>true</i> NOTE if multiple DOORS Instances are available, the DOORS client which had been open the longest is used.	-	-

DOORS Database Structure Schema

The schema for this data source models the structure of a DOORS Database. Using this schema you can query for:

- folders and projects

- modules

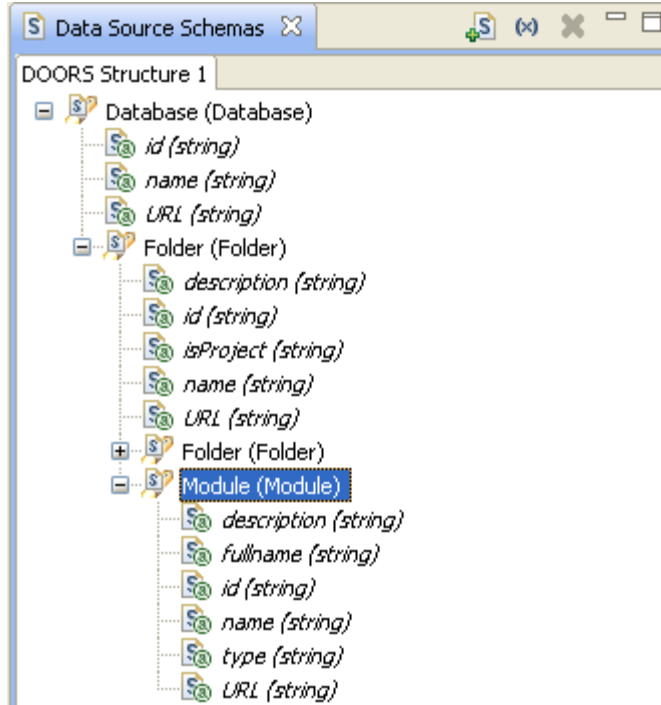


Figure 27 DOORS DB Structure Schema

Database

This is the top level element of the schema.

Attribute	Description
Id	Database unique identified
name	Database name
URL	Database URL

Folder and Project

The schema offers a single element named “folder”. Using its attribute “isProject” you can determine if the result is a folder or a project and depict it accordingly in the output. See the example package with RPE for details.

Attribute	Description
Description	Project/Folder description
Id	Project/ Folder identifier
isProject	“true” if the folder is a project, “false” otherwise.
Name	The unqualified name of the project/folder.
URL	The project/ Folder URL.

Module

The module element allows access to a few properties of the module

Attribute	Description
Description	Module description
fullname	The fully qualified name of the module (/demo/car/test/System requirements)
Id	Module identifier
Name	The unqualified name of the module (System requirements).
type	The type of the module: formal, descriptive, link
URL	The project/module URL.

Baselines and views

The baseline and view list for the modules cannot be obtained as there is no efficient manner of doing it.

Filtering

No native filtering is available for this data source type. RPE filters can be used.

Sorting

No native sorting is available for this data source type. RPE sorts can be used.

Retrieving information recursively

The schema is designed in a fashion that allows retrieving all the structure information with a single query.

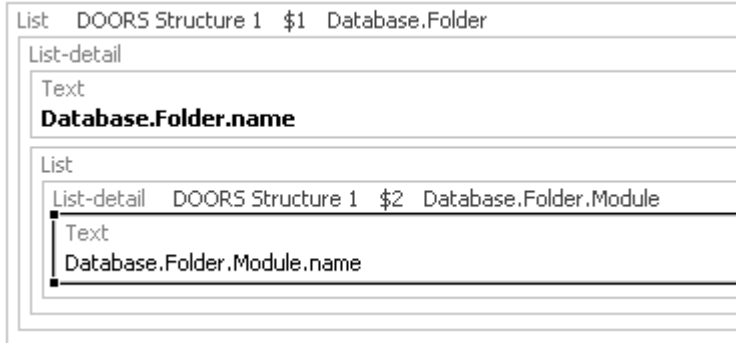


Figure 28 DOORS DB Structure

Defining a recursive level greater than 0 for the “database.folder” query will retrieve all the folders and projects up to that level of nesting.

Tau Data

A Tau source is defined by a single Tau Project (*ttp* file). RPE can extract Tau data as long as IBM Rational Tau 4.2.0.1 or later is installed on the same machine.

NOTE Please check installation guide for details on configuring RPE and Tau for document generation.

Configuring the data source

To configure a Tau data source you need to provide the full path to the Tau project (*ttp*) you want to use. The configuration can be done using the “Configure data source” dialog or by manually introducing the path.

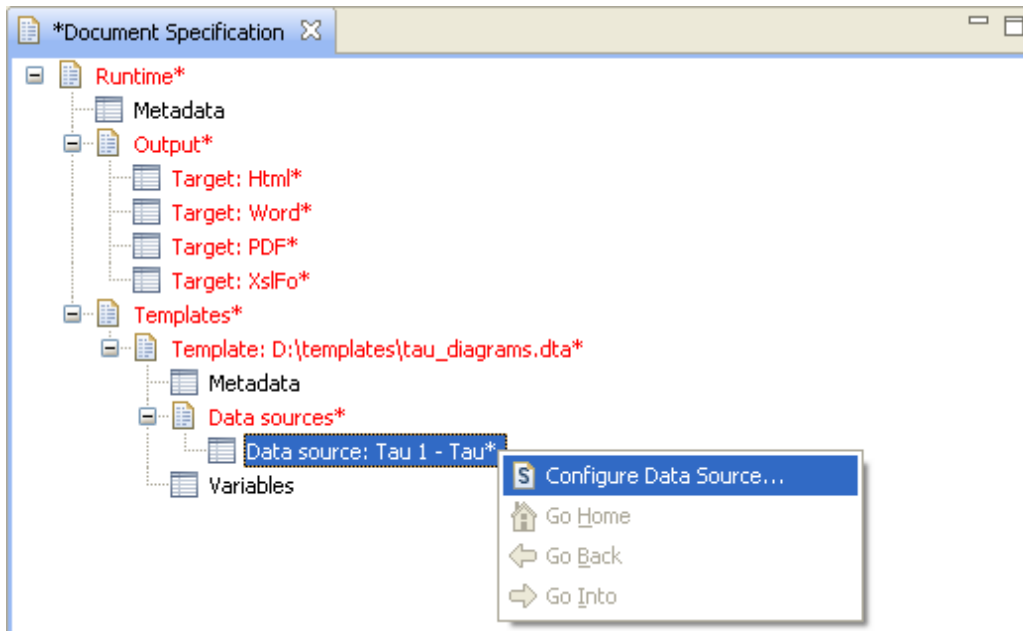


Figure 29 Configuring a Tau data source

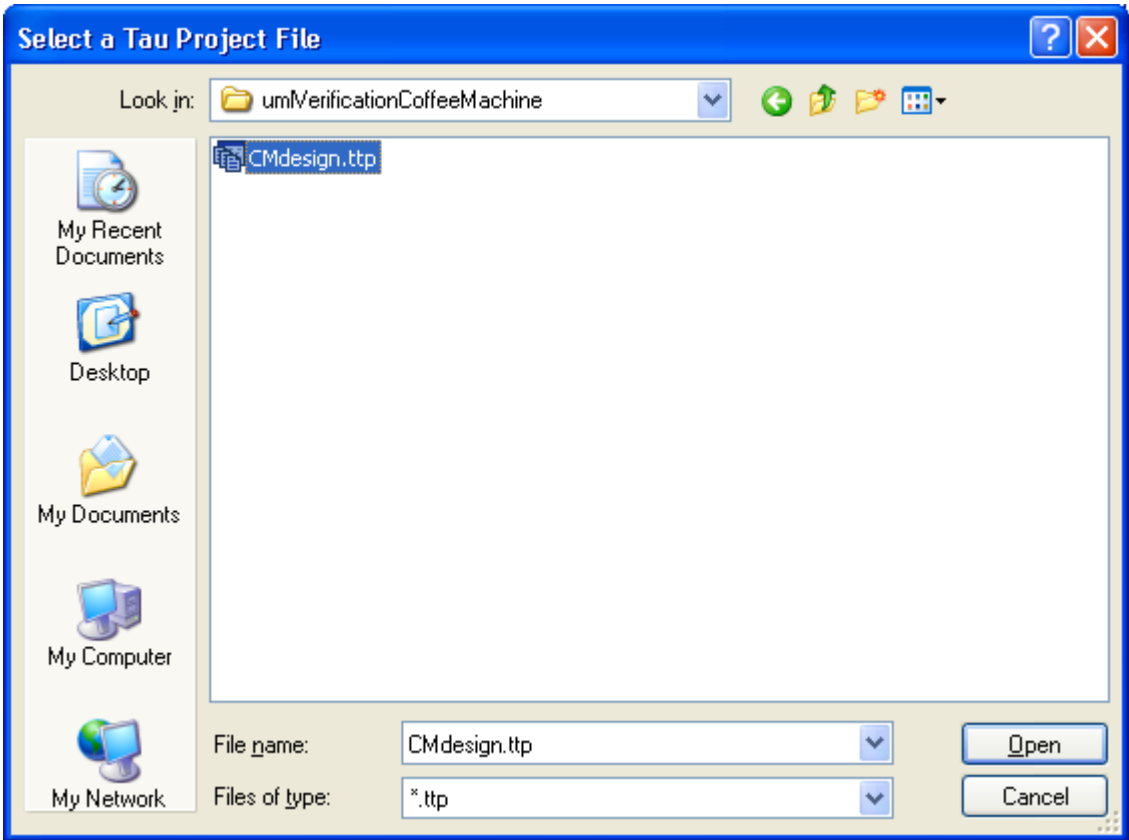


Figure 30 Selecting the Tau Data Source via configure data source dialog

The screenshot shows a "Properties" dialog box with a table of configuration properties. The "URI" property is highlighted, showing the path "Rational\TAU\4.3\examples\umVerificationCoffeeMachine\CMdesign.ttp".

Property	Value
name	Tau 1
description	
type	Tau
driver	eval
URI	Rational\TAU\4.3\examples\umVerificationCoffeeMachine\CMdesign.ttp
command	
ignored	false

Figure 31 Manual configuration

Tau Schema

The Tau schema used by RPE is automatically generated from a Tau metamodel using the provided `GenerateSchema.tcl`. The script is located in `\source\Tau\schema` in the RPE installation folder.

NOTE The predefined Tau schema is built from the Tau metamodel. In order to build Tau documents you need to be familiar with the metamodel structure and its relationship with the model displayed in Tau model browser.

TIP You can use the TCL script provided with RPE to customize the Tau schema, provided you have in depth knowledge of the metamodel you will be using.

The schema view will display all the elements reachable from the root element (usually model).

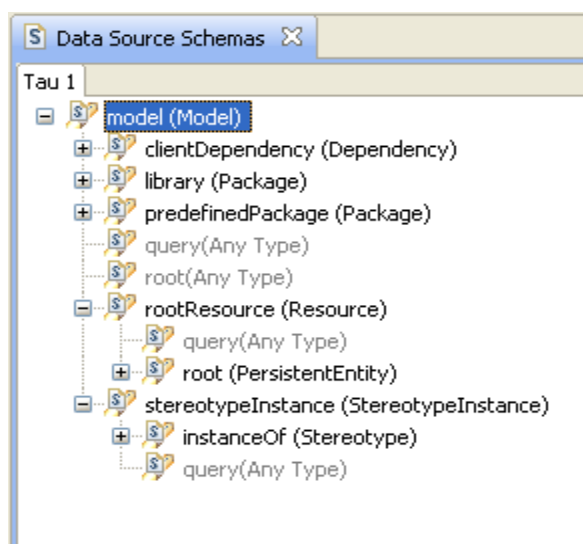


Figure 32 Tau Default Schema

Queries

A RPE query on a schema is a path in the schema, ex: `model.rootResource`. A query is applied to a RPE template element and defines the data context for that element and all its children.

NOTE The RPE syntax for the queries is similar to the XPath syntax. A major difference between the two is that a RPE query does not specify the filter in the query. RPE defines the filter and sort clauses separate from the query itself.

NOTE Each schema element (excepting the query) is defined through a Tau native query (expressed in OCL) that will be used to fetch Tau data. For example, the `root` element under the `model` element has the following Tau query attached: `GetModelRoots()`

Nested queries

Adding queries to template elements and their children creates nested query.

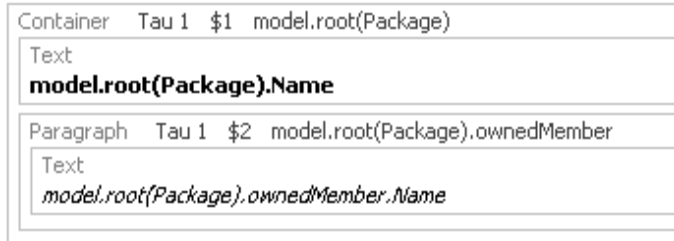


Figure 33 Nested queries

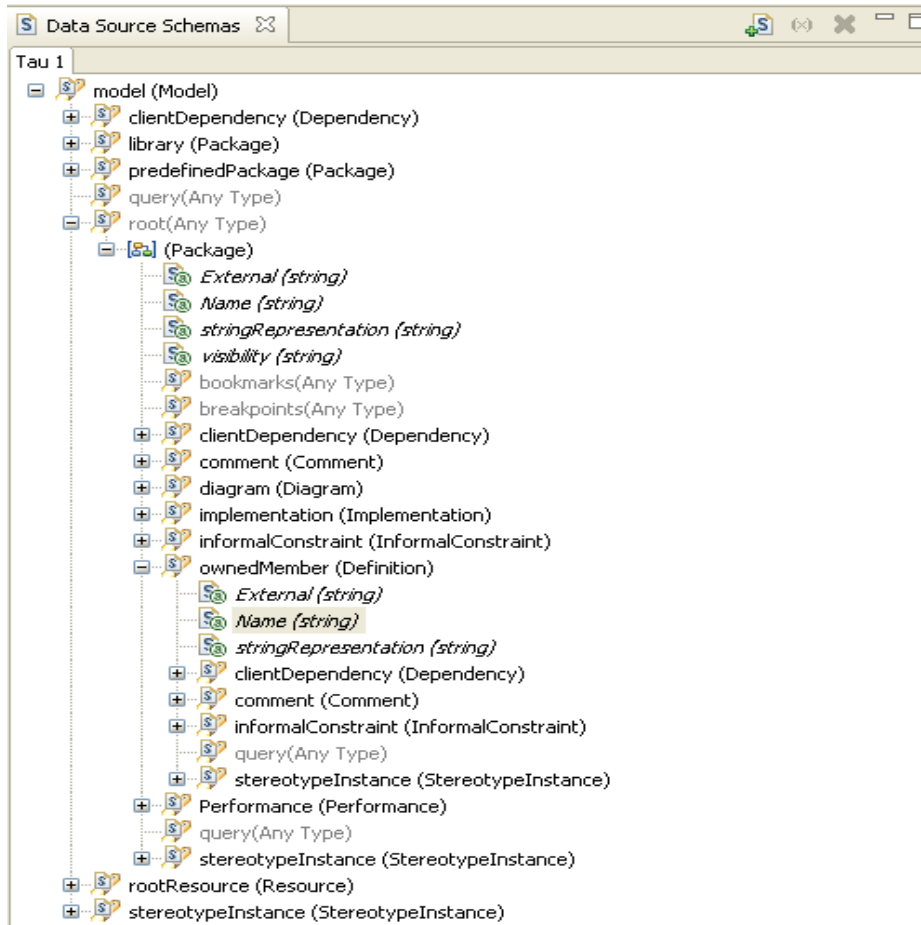


Figure 34 Schema

A nested query will be executed in the context of the results of the parent element. The first query, *model.root(Package)* will be performed in the context of the Tau model. The second query, *model.root(Package).ownedMember* will be performed in the context of each package returned by first query.

How RPE builds queries

Using the above example if the list of all classes from the top level packages in model is needed, the query would be: *model.root(Package).ownedMember(Class)*

Paragraph Tau 1 \$3 model.root(Package).ownedMember(Class)	
Text model.root(Package).ownedMember(Class).Name	Text model.root(Package).ownedMember(Class).isActive

Figure 35 query

While in this form the output document will no longer contain the name of each package, RPE will build the class list in the same way as it did in the first case. The query is split into its component queries, and each query is executed in the context defined by the previous queries:

Sub query	Context	Result
model	-	the model
model.root(Package)	model	list of packages
ownedMember(Class)	list of packages	list of classes

Each sub query is performed once for every element in the context, and the results of each execution are concatenated. These results become the context for the next sub query, or the results list if the sub query is the last one.

Filtering

For Tau Data Sources both native and RPE filters can be used.

NOTE You cannot have a RPE filter and a native filter applied at the same time on a query

Native Filter

The native filter for a query is specified in the “native filter” tab of the filter editor in Document Studio. As for any native filtering, the job is delegated to Tau, with no processing done in RPE.

A Tau native filter must be a valid Tau OCL script that returns a collection of elements. The native filter is concatenated to the underlying Tau query for the current schema element.

Example

Schema element:

model.predefinedPackage

Underlying Tau query:

```
GetEntities("predefinedPackage").select(IsKindOf("Package"))
```

Native filter:

```
select(HasPropertyWithValue("Name", "Predefined"))
```

The query that will be executed by Tau query evaluator is:

```
GetEntities("predefinedPackage").select(IsKindOf("Package")).  
select(HasPropertyWithValue("Name", "Predefined"))
```

RPE filter

The scripted filter for a query is specified in the “script filter” tab of the filter editor in Document Studio. Scripted filtering is performed by RPE itself and follows the same rules and limitations as for any other data source.

Sorting Tau data

Tau data sources do not have support for native sorting. For these data sources only RPE sorting can be used.

Type casting

Some schema elements do not have a type assigned to them as there can be more than 1 valid type for that. In such cases you can define which type to use through the “Cast to type” function in the schema view bar.

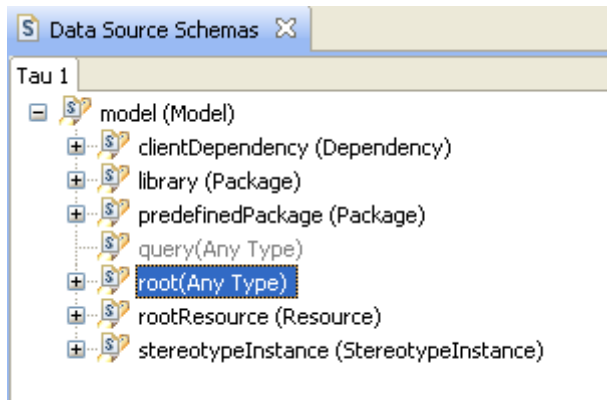


Figure 36 An element with no type

NOTE You cannot build queries of type “Any Type”. A cast needs to be defined before RPE will allow using the query.

NOTE A cast query will filter the results of the regular query to return only the elements that can be casted to the selected type. Internally this is implemented by adding the `select(IsKindOf("Type"))` filter to the query.

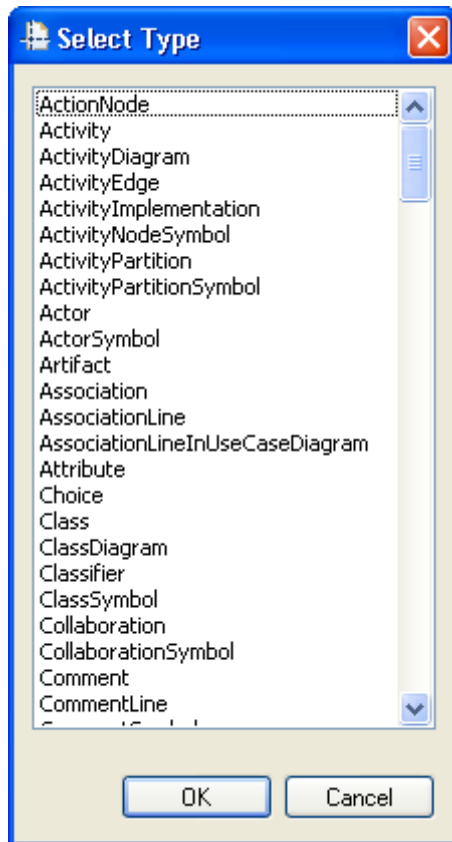


Figure 37 Select type

After the type is selected, it becomes available in the Schema View under the “anyType” element.

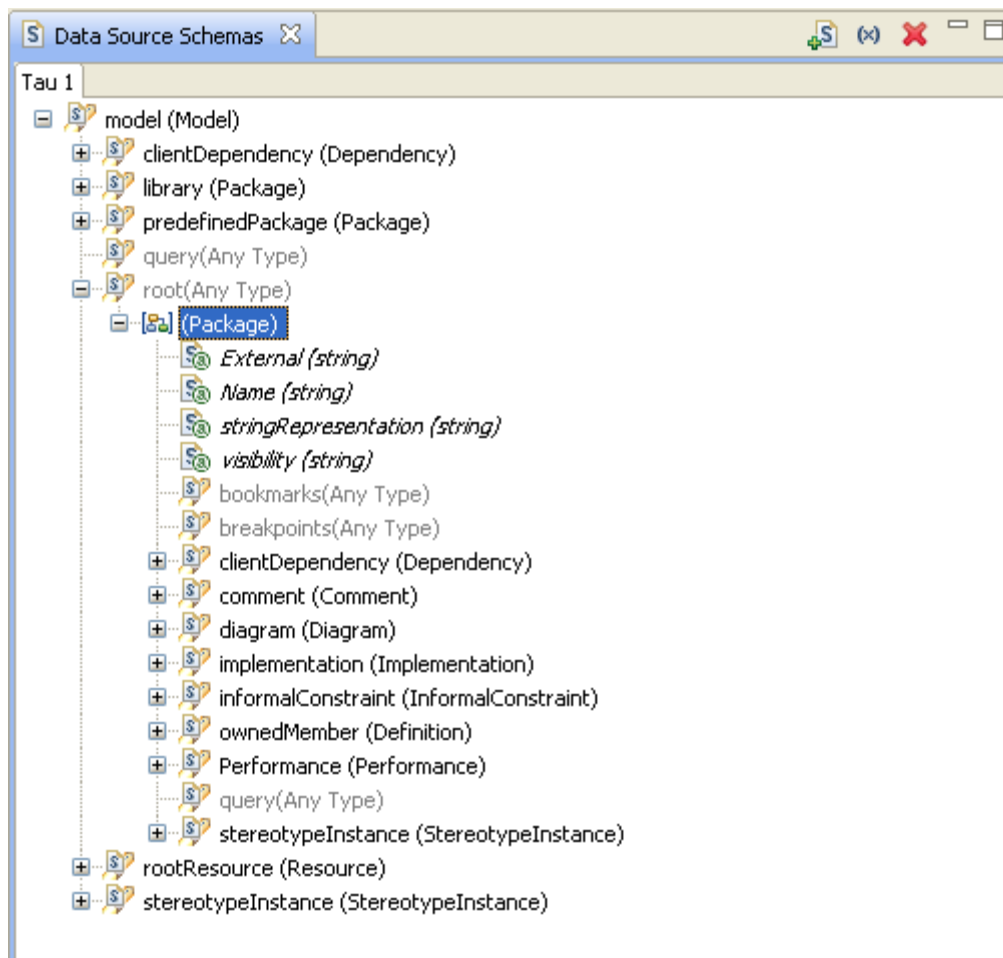


Figure 38 A "cast" to Package

Queries built using "cast" allow access to all the child elements and attributes of the type cast. Type casting can also be used to refine the results of a query. Type casting works essentially as a secondary filter for Tau elements.

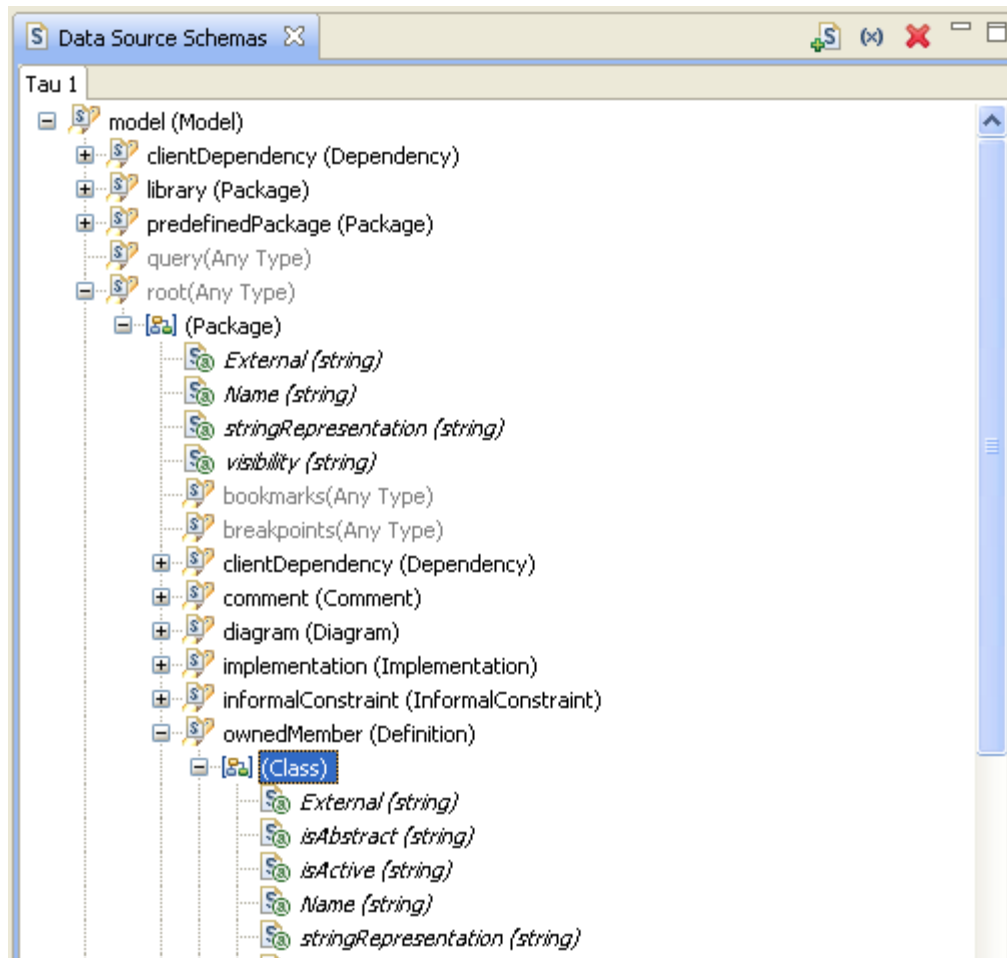


Figure 39 Cast

Adding a “Class” cast to the “ownedMember” element of a Package allows defining the following query:

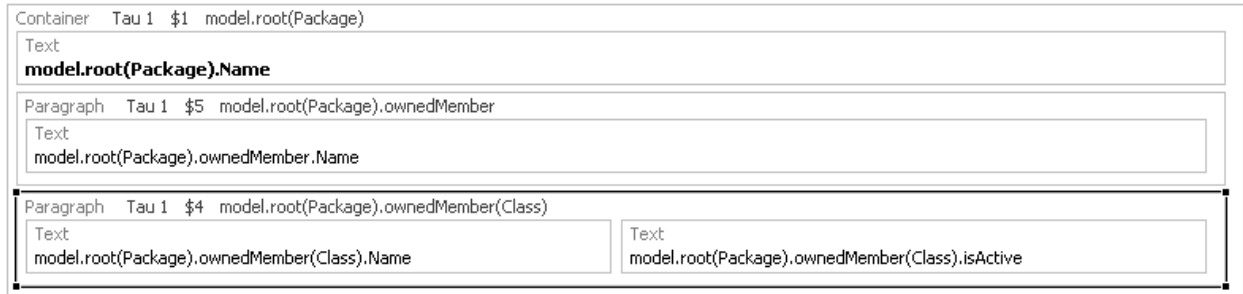


Figure 40

In the above example the query \$1 retrieves all the definitions contained in each top level package while query \$4 will return only the definitions that are classes from the same context.

NOTE With the current RPE version you can specify casts that are not correct (ex: from Class to Package). This query will yield no results.

Cast vs. Filter

The result set returned by a *cast query* is identical with the result set returned by query having a filter using an equivalent *IsKindOf* predicate. The main difference is that using a cast query gives access to the cast type attributes and child elements while using a query with a filter does not.

Attributes

Once queries are assigned to template elements, all the attributes of the elements returned by the queries can be used.

An attribute can be used in multiple ways:

- To define the content of a template element
- To be used in expressions calculating the content of a template element or its formatting capabilities

Special attributes

GUID

Every schema element has a special attribute named “guid”. This attribute is filled by RPE with the unique GUID of the current model element.

image

Every schema element representing a Tau diagram has a special attribute named *_image*. Using this attribute will generate an image file for the current diagram, image that can be included in the output.

stringRepresentation

Available for all expressions, actions and definitions, this attribute holds the unparsed representation of the element.

The query element

Every schema element has a special child element named “*query*”. Unlike the other elements in the schema, a “*query*” has no underline Tau query assigned and no type. Using the query element as is will return no results.

The purpose of the query element is to offer another level of customization for the RPE document generation. Should the existing elements should not suffice (or be optimal) for a given task, the user can define what he wants to extract (the type) and how he wants that extracted (the query) using the query element. The *type* is defined by adding a *type cast* to the query element while the *query* is defined in the template element’s filter, as *native filter*.

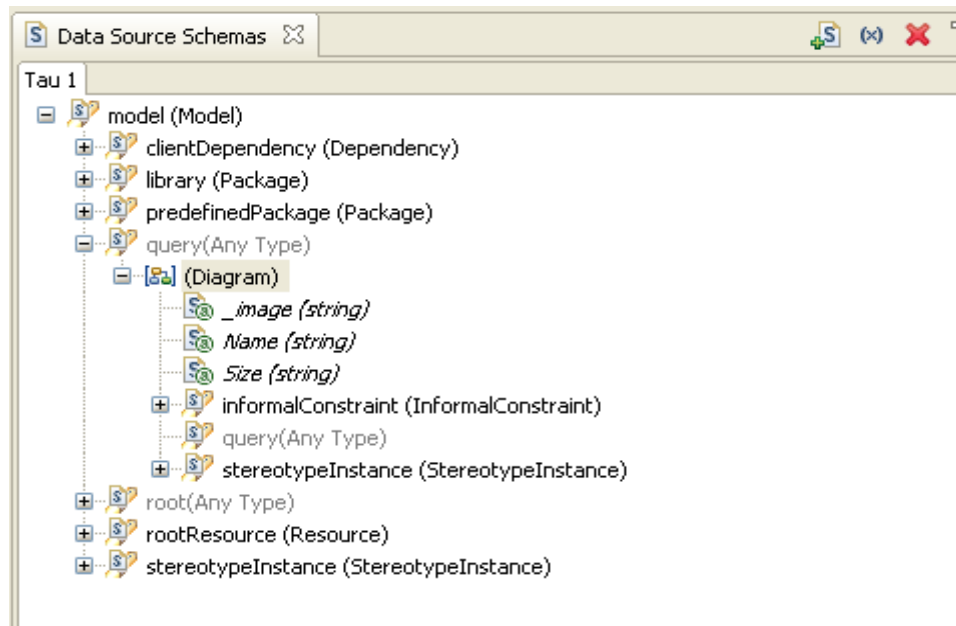


Figure 41 A cast added to a "query" element

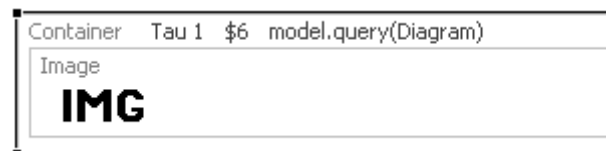


Figure 42 The query element with cast is used in the template

For the above scenario, a valid query is:

```
GetAllEntities().select( IsKindOf("Diagram"))
```

NOTE RPE will filter the results of the query based on the type cast used. If the query is syntactically correct, the result will contain only elements matching the specified type cast.

XML

RPE can extract data from any valid XML file if provided with the schema of the XML.

NOTE The document generation process will fail if the XML document is not valid (according to W3C XML definition – www.w3c.org/XML). A simple way to verify the file's validity is to open it in any web browser supporting XML. If errors are reported, RPE will not be able to handle the XML file either.

NOTE By default RPE does not try to validate the XML source against the schema used in the template. This will lead to documents being generated even if the XML file does not respect the schema. How much of the data can be generated depends on how different the XML schema used in the template and the actual XML file are.

Configuring the data source

The only mandatory property that needs to be configured for an XML data source is the URI of the XML file. The file can reside on the local file system, an accessible network file system, or on a remote system accessible through a URL.

An XML data source can be configured via the “Configure data source” function in the launcher.

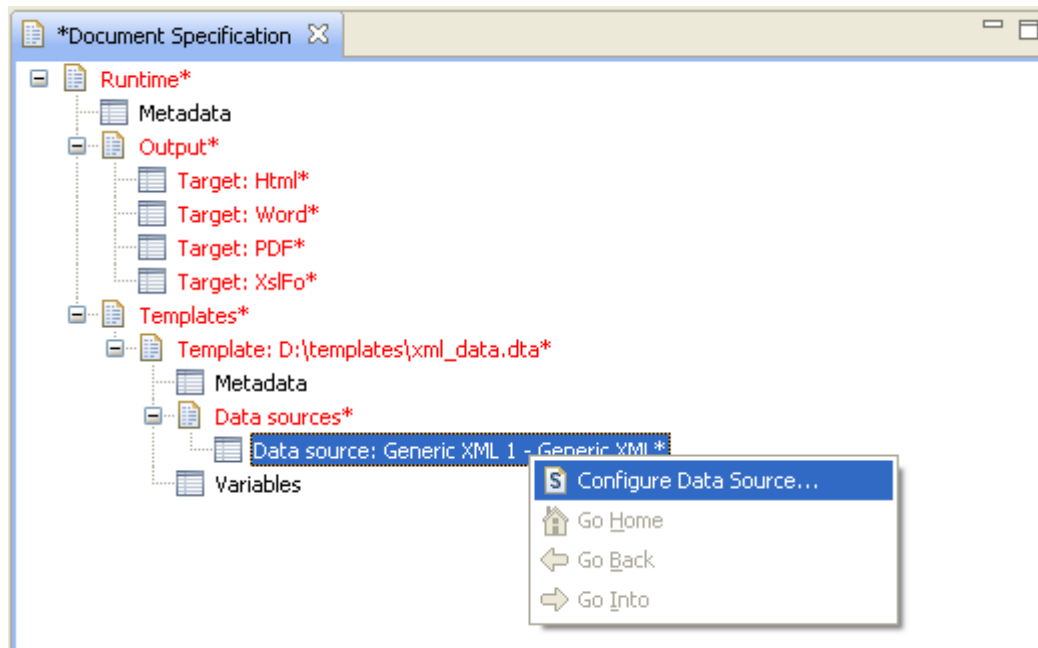


Figure 43 Configuring an XML data source

A configuration dialog is shown:

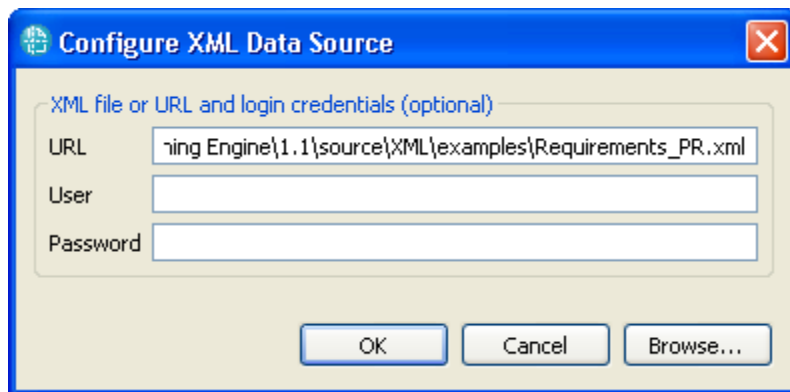


Figure 44 XML data source configuration

Alternatively you can input the URL to the XML source in the properties window:

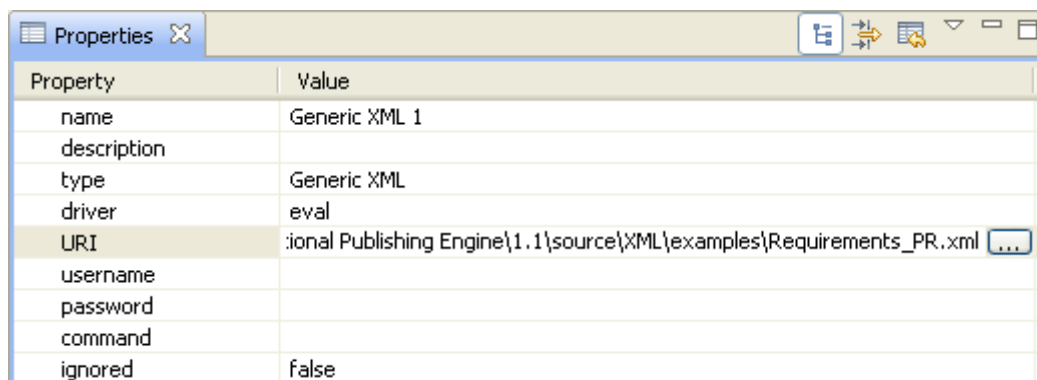


Figure 45 Manual configuration

NOTE If a web browser can open the XML file than RPE should also be able to access the file using the same URL.

In addition to the URI you can provide a user name and a password for accessing the data. RPE will attempt to authenticate using basic authentication and form base authentication.

What can be extracted

Any attribute specified in the schema. For XML simple types RPE provides an attribute named “_value” that allows access to the content of the element.

NOTE If an attribute is not specified in the schema it is not accessible from RPE.

REST Data

RPE can extract data from REST data sources exposing the Rational REST Get Specification.

Obtaining a schema

RPE requires a schema for template design and document generation. Schemas for REST data sources can be obtained through the Schema Discovery feature of Document Studio.

Alternatively you can provide the URL to the schema in the Add Data Source Wizard from document studio.

Configuring a REST Data Source

Configuring a REST data source requires providing the URL of the resource and optionally a user name and password. The user name and password are needed if the resource is on secure location.

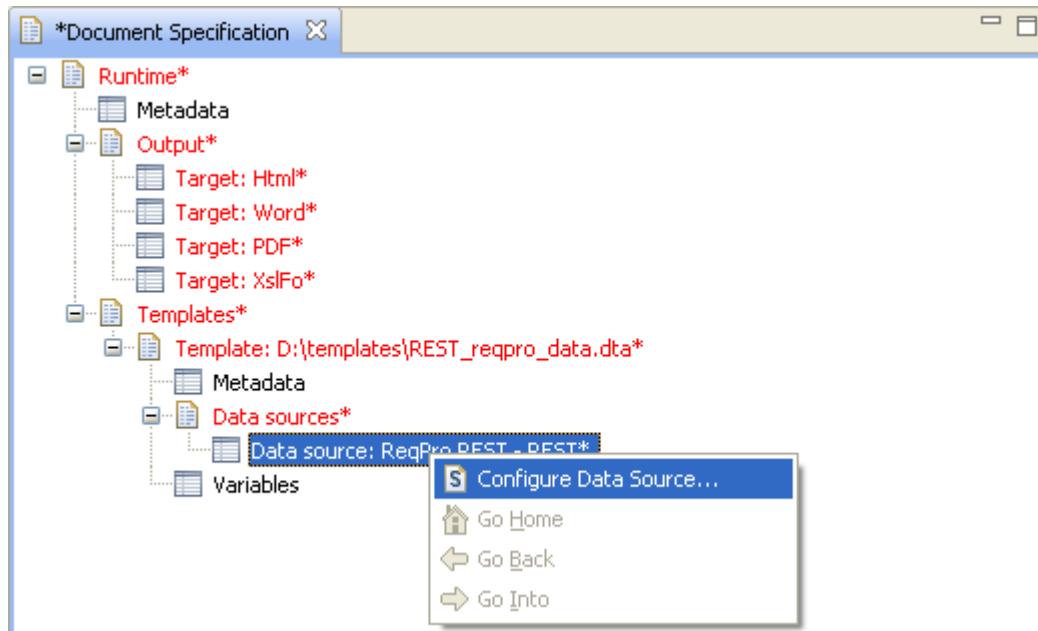


Figure 46 Configuring a REST Data Source

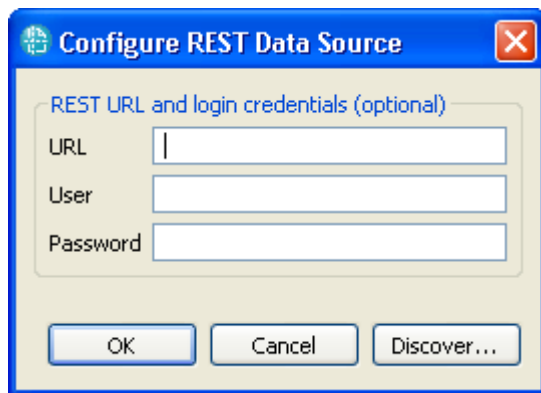


Figure 47 Configuring a REST Data Source

Alternatively you can provide the URL to the REST resource in the document specification.

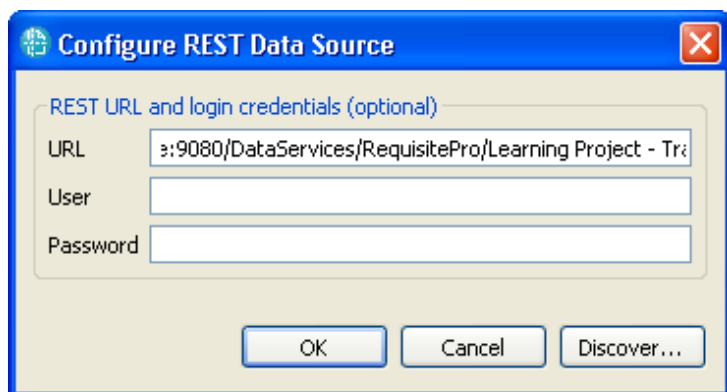


Figure 48 Configuring REST Data Sources

Resource Discovery Wizard

Clicking the “Discover” button will trigger the resource discovery wizard. The wizard assists you in discovering the resource.

NOTE The wizard only functions for REST resources complying with the Rational REST Get Specification.

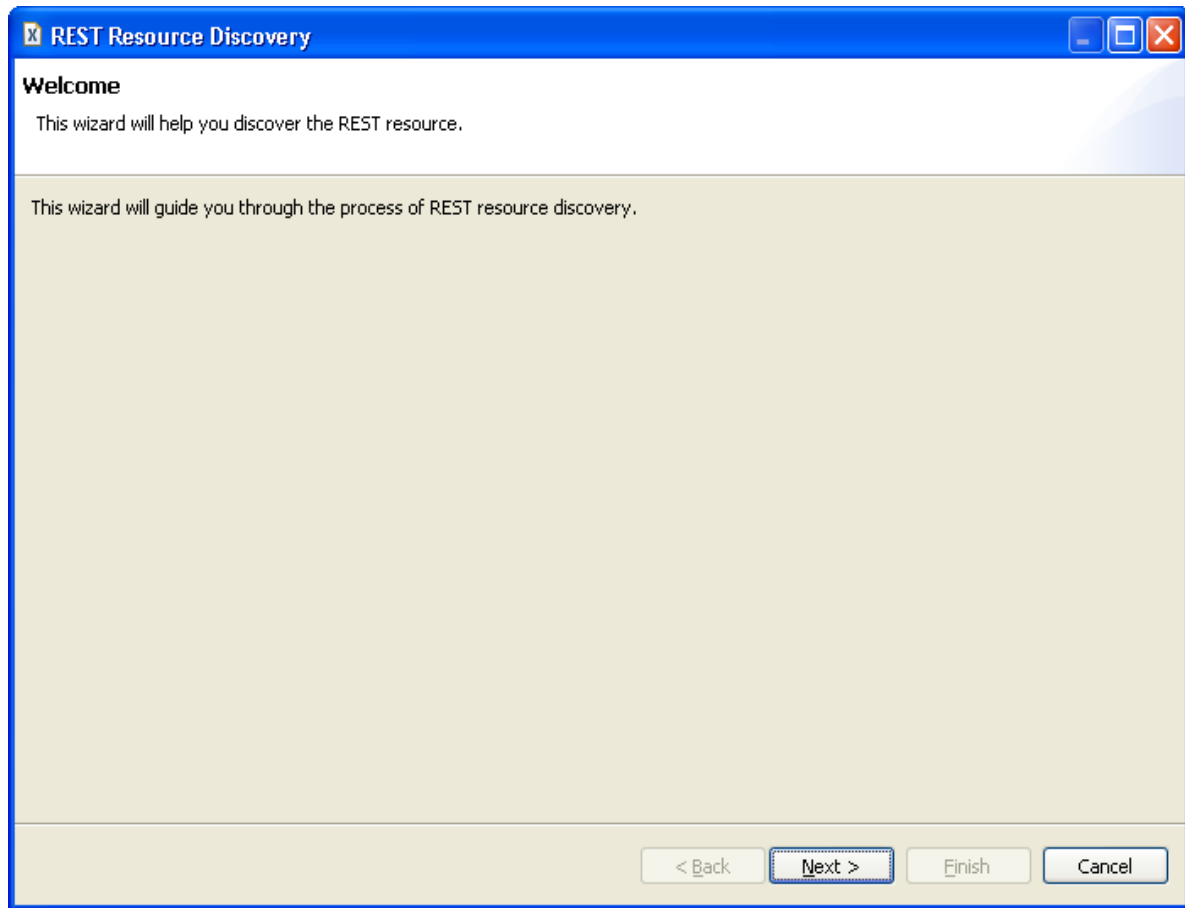


Figure 49 Welcome Screen

After the welcome screen you have the possibility to configure the base URL to be used to start the resource discovery along with credentials information if needed.

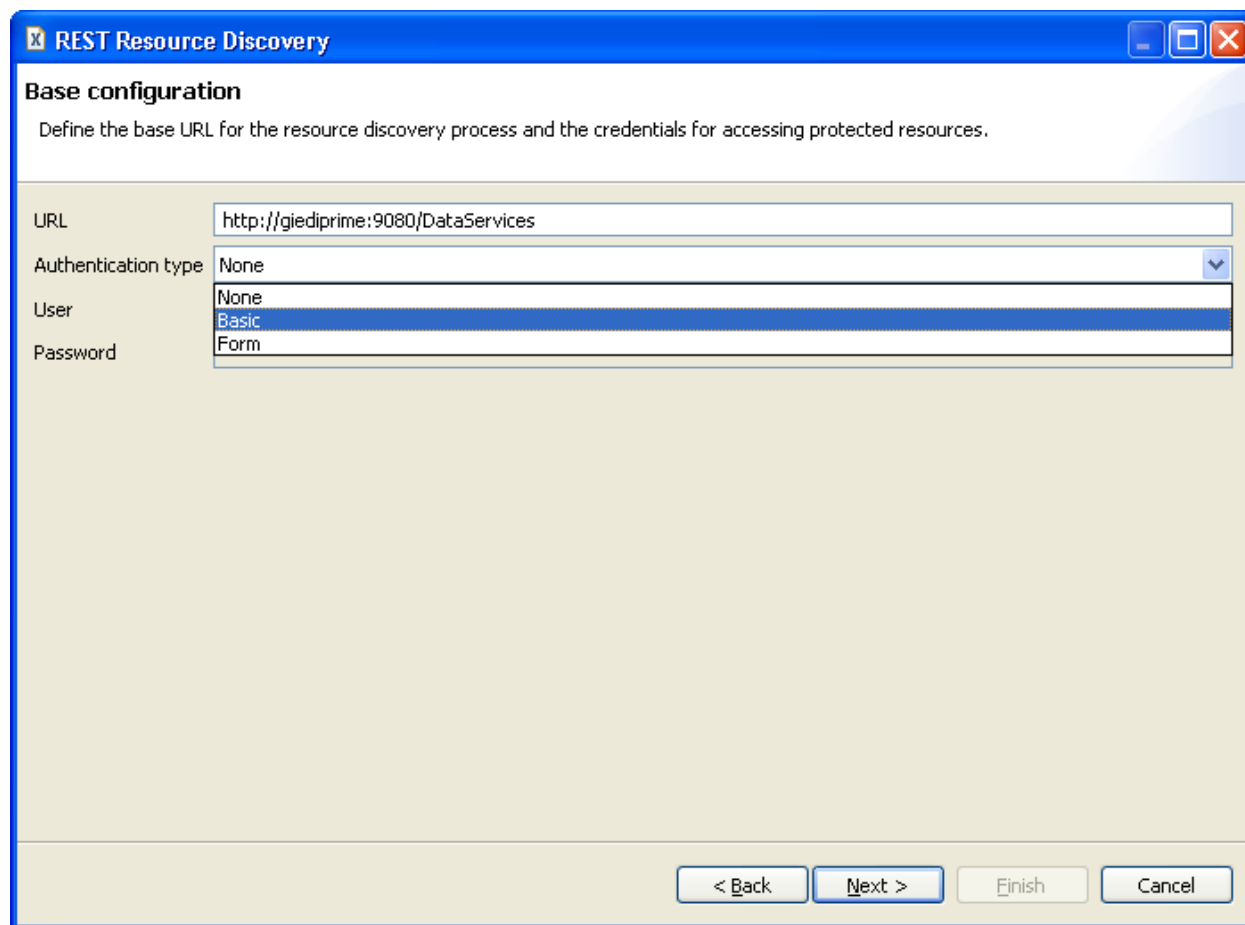


Figure 50 Base configuration

NOTE Please consult the documentation of the REST data source to learn what is the base URL and what credentials are needed.

The next screen displays the name and the description of the data source being configured. If the REST data source is accessible via the Rational Insight Data Service you can use the "Locate using data service" option, otherwise you need to provide the relative URL of the resource.

The screenshot shows a Windows-style dialog box titled "REST Resource Discovery". The main heading is "Create a Resource". Below the heading is a descriptive paragraph: "Specify a name, description and relative path for the new resource. If the resource is accessible via the Rational Insight Data Service, you can use the data service to locate the resource and determine its relative path." To the right of this text is a small icon of a document with a blue 'X' on it. The form contains three input fields: "Name:" with the text "ReqPro REST", "Description:" which is empty, and "Relative path:" which is also empty. Below the "Relative path:" label are two radio buttons. The first is "Manual entry" and is unselected. The second is "Locate using data service" and is selected. At the bottom of the dialog are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Figure 51 Data source details

At this point the wizard will query the REST provider for information about the resources available. The resource navigation is possible for resources having a href attribute. You need to drill down the resources until you identify the one you need.

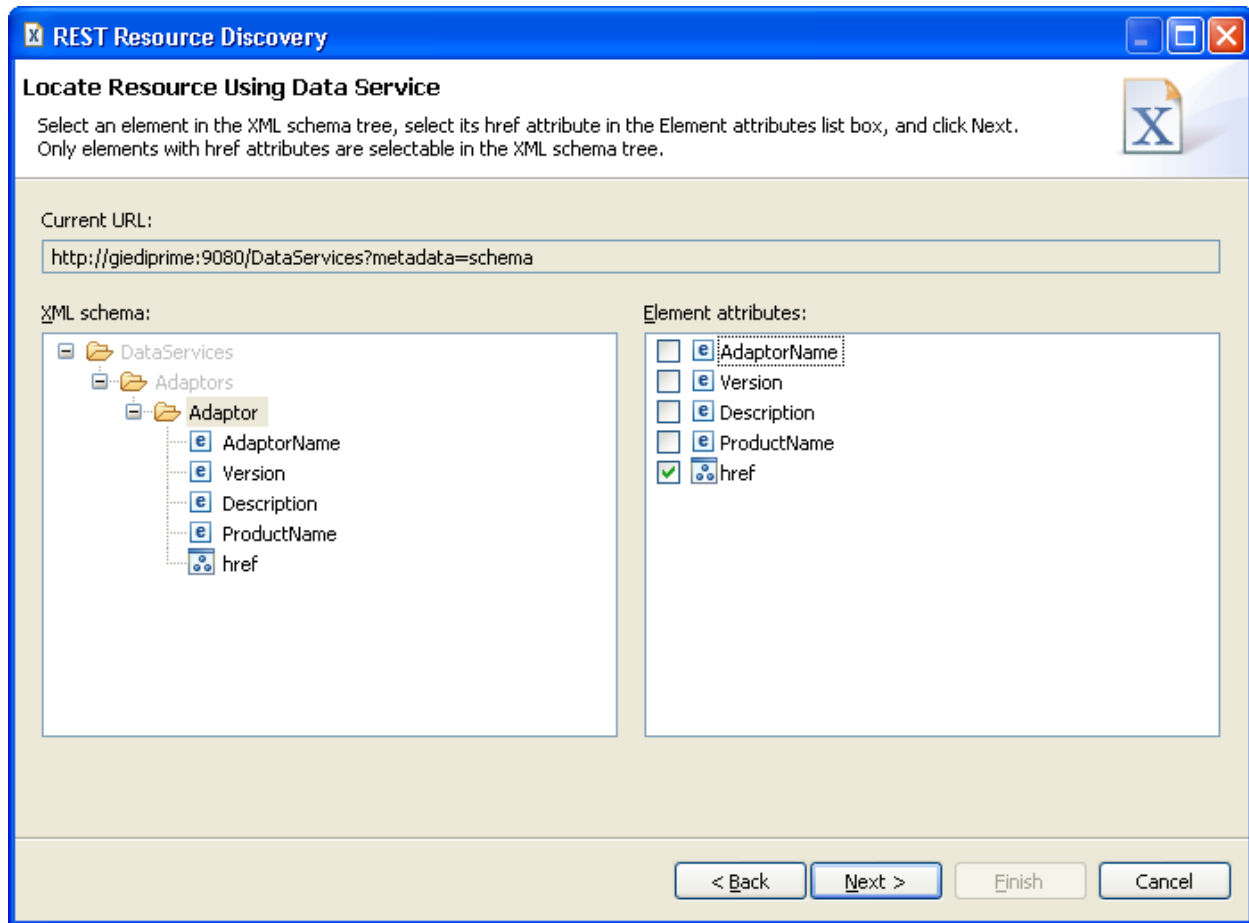


Figure 52 Resource discovery

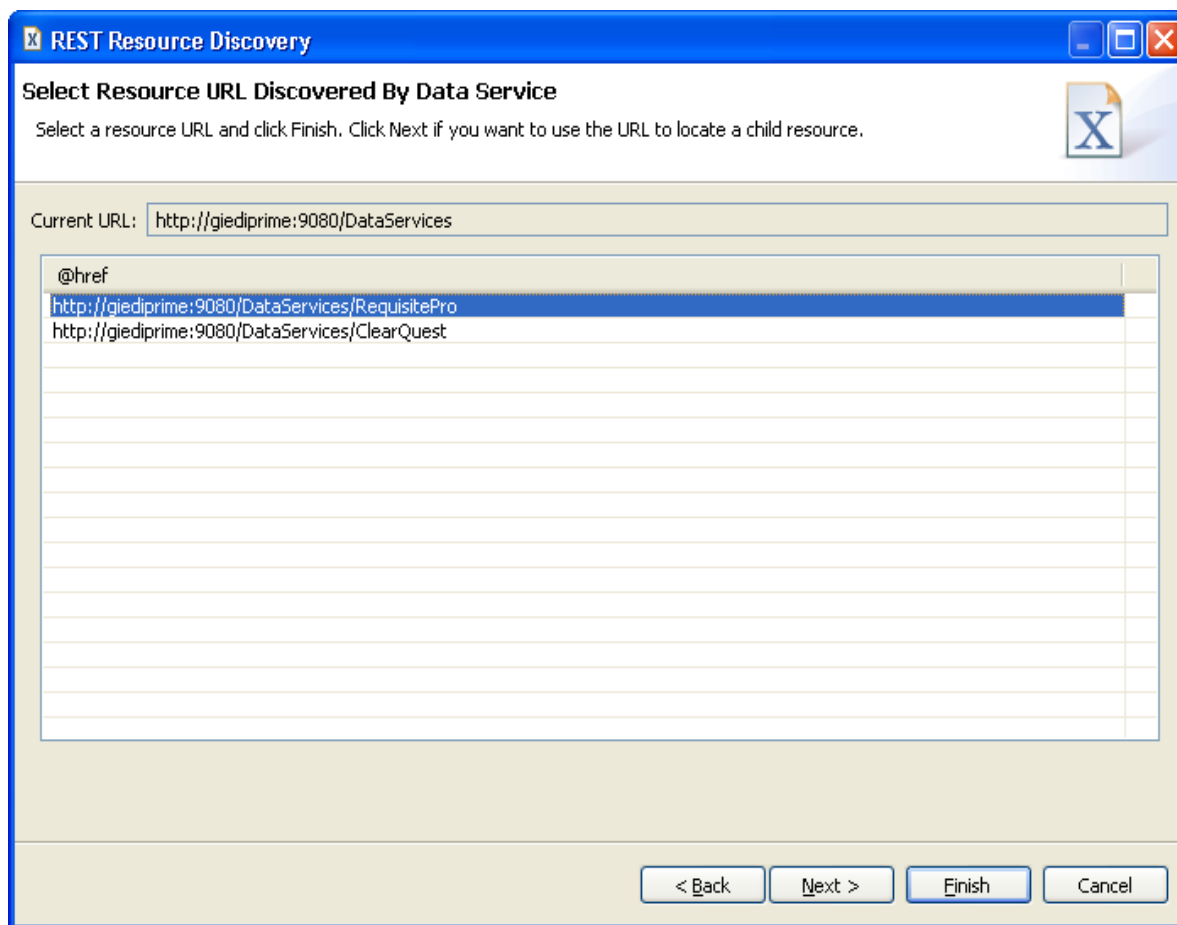


Figure 53 Resource discovery

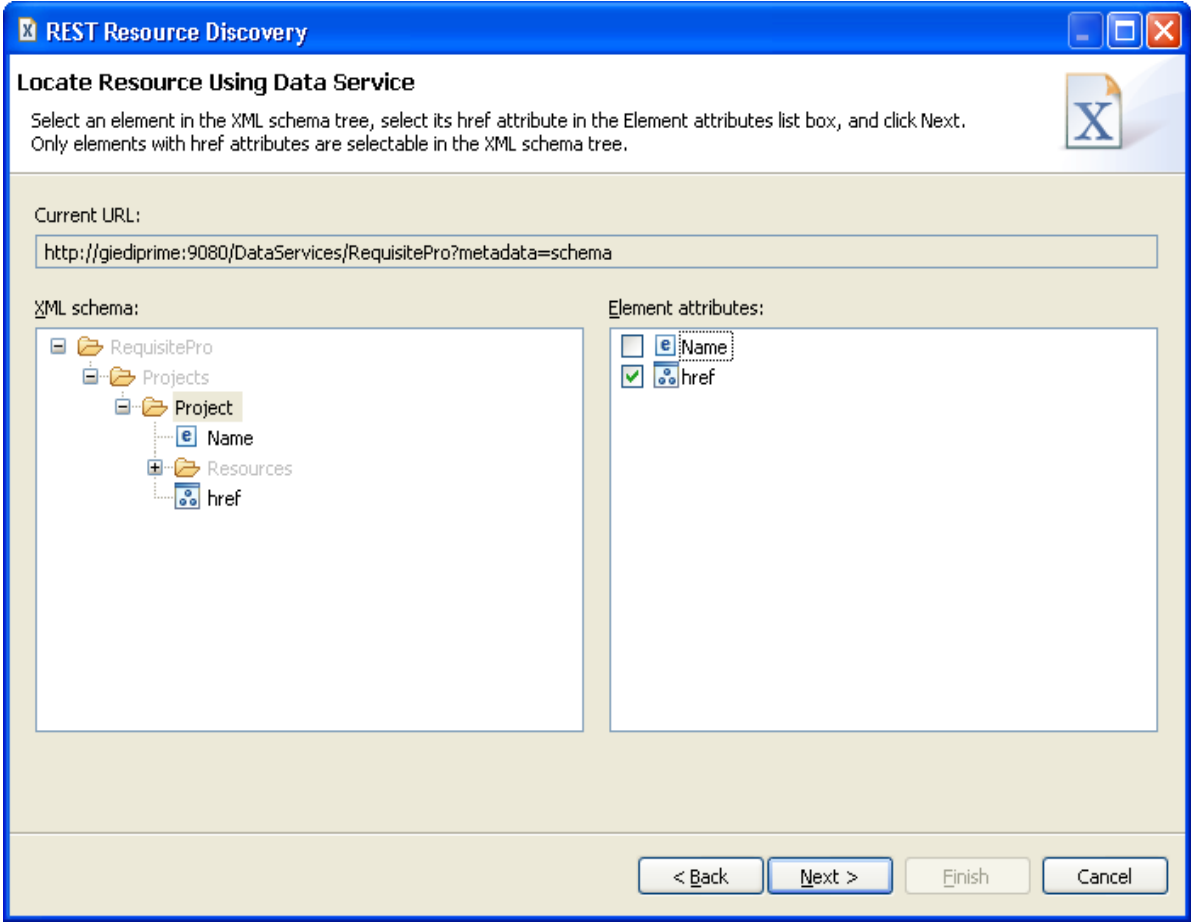


Figure 54 Resource discovery

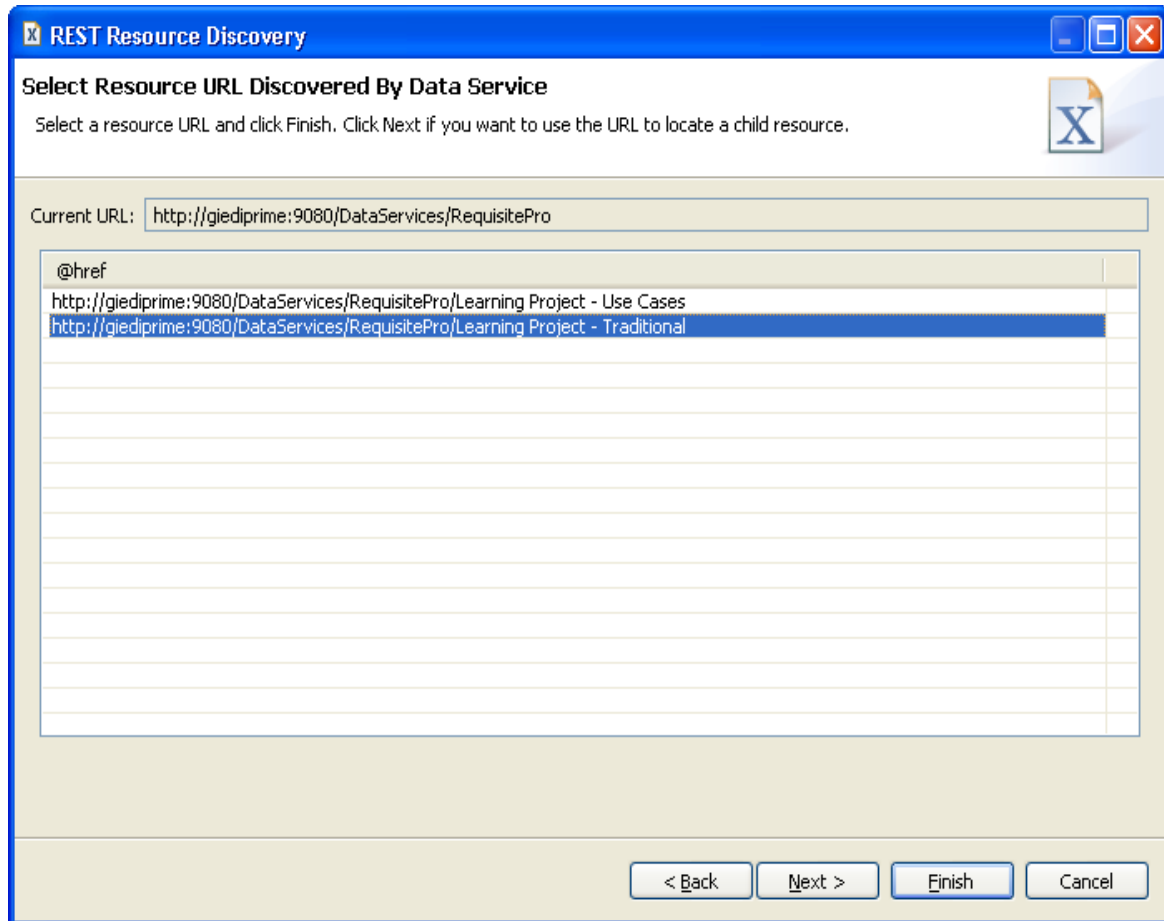


Figure 55 Resource discovery

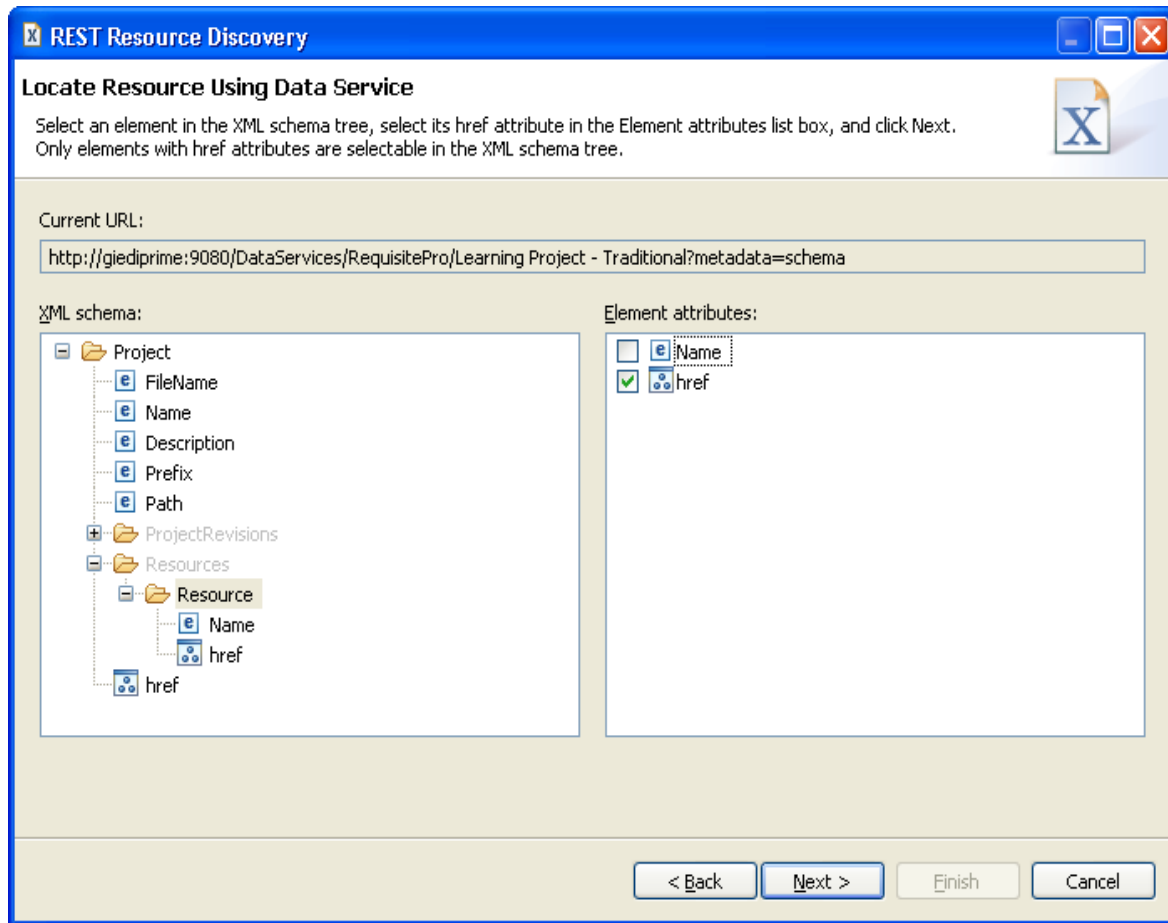


Figure 56 Resource discovery

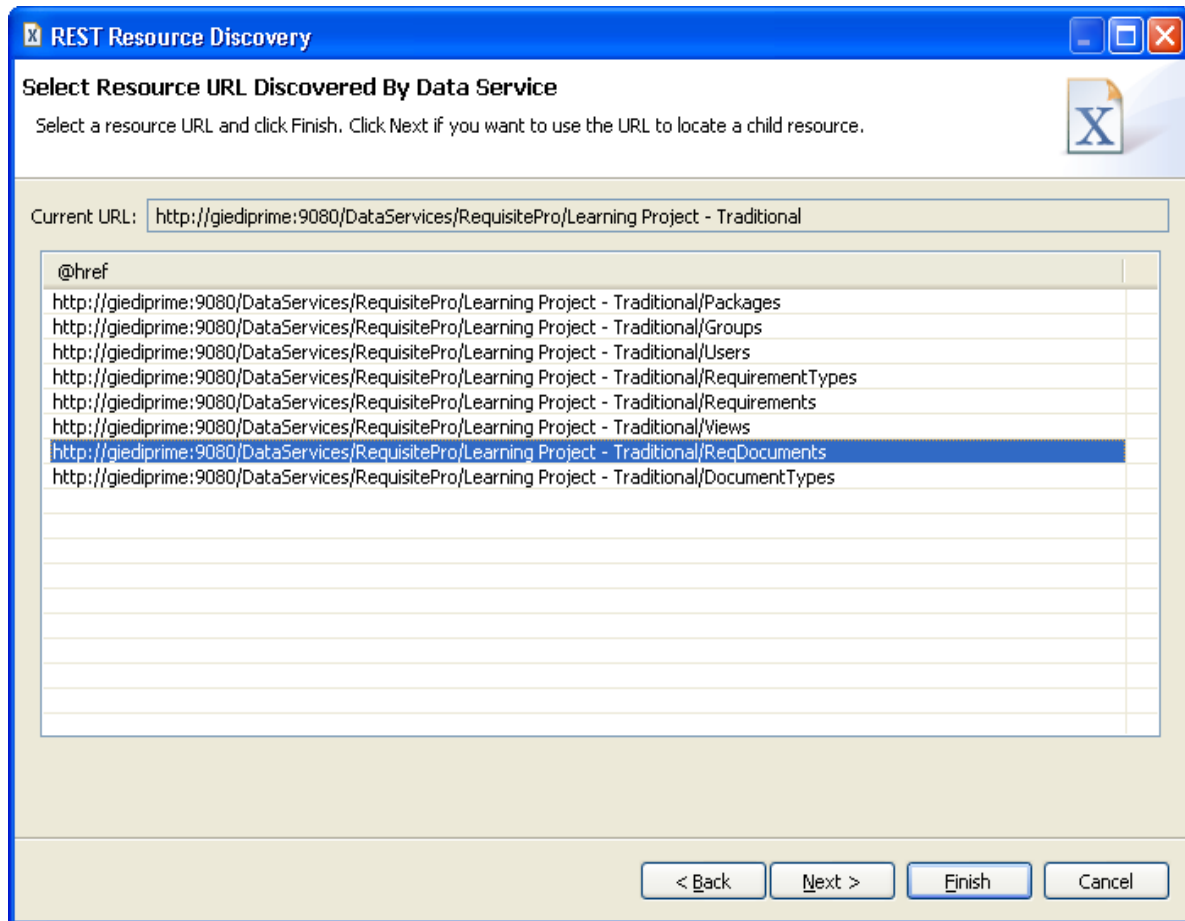


Figure 57 Resource discovery

Once you have identified the resource you can click finish. At this moment the wizard completes and the URL you've discovered is displayed in the configuration dialog.

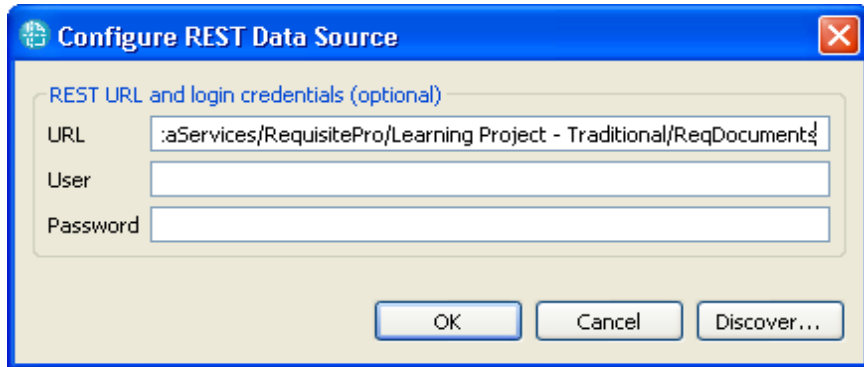


Figure 58 Discovered URL

You can provide a user name and password here if your resource requires. Clicking ok will complete the REST Data Configuration process and will fill the gathered information in the Document Specification.

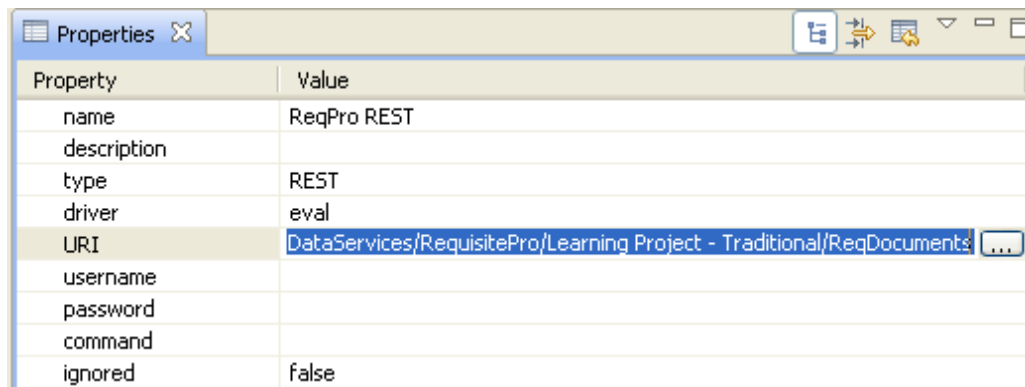


Figure 59 Configured REST Data Source

Alternatively you can manually enter the URL and credentials in the properties window.

Filtering

When getting data from REST sources, RPE creates the REST Get URL based on the template's contents. If the REST provider fully implements the REST Get Specification, then the data returned is a filtered subset of the resource.

An example REST URL using the "fields" argument is:

```
http://giediprime:9080/DataServices/RequisitePro/Learning Project - Traditional/  
Packages?fields=Project/Packages/Package/(Name|GUID)
```

NOTE You must provide the base URL of the resource; RPE will generate all the required arguments.

Native filtering

RPE supports REST native filtering. You define the filter, according to the REST Get Specification as native filter.

Paragraph	REST 1	\$4	Project.Packages.Package
Text			
Project.Packages.Package.GUID			
Paragraph			
Text			
Project.Packages.Package.Name			

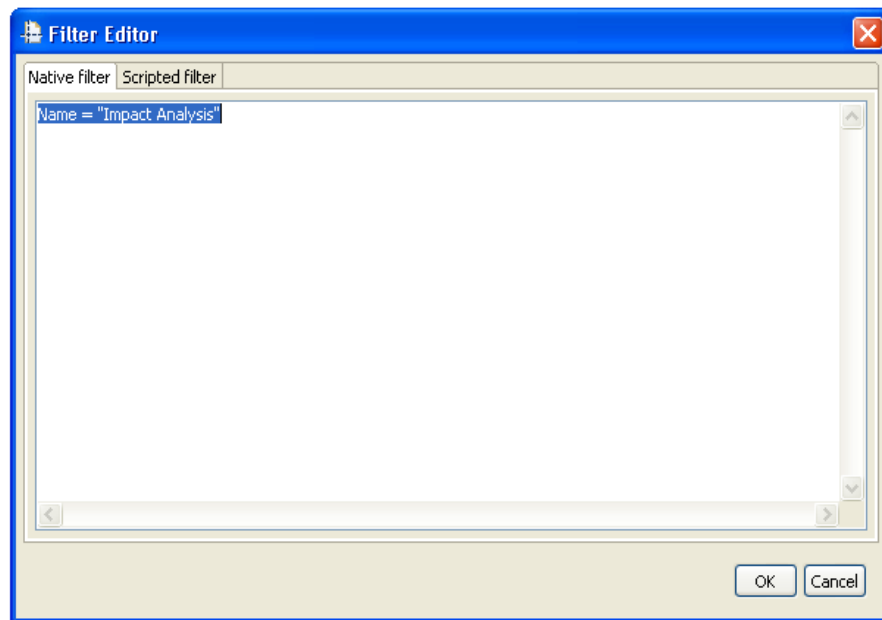


Figure 60 REST Native Filter

The REST URL generated for the above template would look like this:

`http://giediprime:9080/DataServices/RequisitePro/Learning Project - Traditional/Packages?fields=Project/Packages/Package[Name = "Impact Analysis"]/(Name|GUID)`

NOTE you must provide the base URL of the resource; RPE will generate all the required arguments.

RPE filters can also be used.

RPE Output

RPE can produce Microsoft Word, HTML, PDF and XSL-FO documents. No additional software is required for the document generation process. You might need additional software to view the results of document generation or to perform post processing operations where applicable.

RPE offers a broad range of template elements and formatting operations but it is not, and doesn't attempt to be, as rich as a fully fledged Word Processing environment such as Microsoft Office or Open Office.

RPE's Template Editor does not attempt to be a replacement for your Word Processor. Although RPE's Template Editor allows building templates visually, its primary purpose is to define the document flow and structure hence not all of the formatting that can be specified is rendered visually.

For example, while you can specify that a paragraph should be centered, the Document Studio will provide no visual feedback. The generated document on the other hand will contain the paragraph centered as expected.

Stylesheets

The HTML and Word output formats allow specifying stylesheets to be used in the document generation. A stylesheet can be used to ensure certain styles are available in the generated document. A stylesheet can also be used to add content to the generated document: standard front pages with company logos, legal information etc.

Word

The Microsoft Word documents generated by RPE are in the Word 97-2003 format. You do not need Microsoft Word or any additional software installed on your machine unless you want to perform post-processing operations on the output.

Defining a Word output

By default any newly created document specification contains a Word Output. If you have deleted it you can add it at a later time. A Word output is defined by the following properties:

Property	Description	Required
path	The full path where RPE should produce the output document. If not specified, RPE will produce the file in the temp folder (with a random generated name).	no
stylesheet	The full path to the stylesheet to be used.	no
macro	The macro to run after the document is generated. NOTE Requires Microsoft Word installed on the machine hosting the RPE engine.	no

Word 2003 and 2007 formats

Selecting the output format for Word is made through the extension used. If doc is used the generated document will be saved in Word 2003 format. If docx or docm is used then Word 2007 format is generated.

TIP In order to run macros for Word 2007 formats you need to specify the docm extension (Document with Macros).

Output

RPE's Word output consists of a single Microsoft Word document. You need to manually (or via a macro) update any fields you might have added, such as Tables of Contents, Tables of Figures, Tables of Tables, figure captions and label captions.

NOTE Due to implementation details RPE cannot update Microsoft Word fields in the generated document.

Stylesheets

You can specify any Microsoft Word 97-2003 document or template (.doc or .dot) or Microsoft 2007 (.docx or .dotx). RPE will create the output document based on the stylesheet. This translates into:

- Any content in the stylesheet will be present in the generated document
- All styles from the stylesheet will be available in the generated document

Unicode data

Unicode data will appear as Unicode as long as the font used in the output can display such characters. The specified font must exist on the machine used to view the document.

The font family used for a template element can be specified through:

Location	Applies to
"font family" property of the element	The element and all the child elements that do not explicitly set the property.
"font family" property of the style applied to the element	The element and all the child elements that do not explicitly set the property.
font properties of the styles defined in the stylesheet (other than normal)	All the template elements using the specified style
Normal style in the template	All template elements that do not have a different style applied.

OLEs

RPE can extract OLEs from data sources supporting them, such as DOORS. Depending on the “OLEs as static images” flag in the metadata section of the document specification the OLEs are inserted in the output document as static images or as include fields to rtf files containing the OLEs. If “OLEs as static images” is set to FALSE, RPE will generate a “ref” folder in the same location as the Word output document. The ref folder contains RTF files for the OLE objects in the DOORS data source. The Word output will have one include field pointing to a RTF file for each OLE exported from DOORS.

NOTE RPE cannot update Microsoft Word fields. As a consequence the include fields will not be visible when you open the generated Word document. To make the fields visible you need to do one of the following:

- Select all the document content and use the “Update fields” function in Word
- Use the “updateFields” macro provided by RPE
- Use the “insertOLEs” macro provided by RPE

NOTE Updating the fields in the document will not insert the OLEs in the document. Moving such a Word document from the machine it was generated on will prevent editing the OLEs. To make the document self contained you need to run the “rpe” or the “insertOLEs” macro.

Post-processing

When generating documents in Word format it is possible to specify a macro to be executed at the end of the process. The macro will be executed only if Microsoft Word is installed and available on the machine hosting the RPE engine.

NOTE The macro needs to be defined in the stylesheet used for Word output.

NOTE The macro is run with Microsoft Word in visible mode. This behavior is intended so that you can confirm/cancel security messages or any other messages added by the macros.

NOTE The macro execution might take a while depending on how large your document is and how complex the macro is. You should wait for the macro execution to complete before opening the Word document.

NOTE In order to run macros for Word 2007 formats you need to specify the docm extension (Document with Macros) for the output document.

Word macros

RPE comes with a predefined set of Word macros meant to assist you in post processing RPE generated Word documents.

updateTOCs

Location: RPE_INSTALLATION/utills/word/updateTOCs.txt

Updates all the Table of Contents in the document.

updateFields

Location: RPE_INSTALLATION/utills/word/updateFields.txt

Updates all the fields in the document.

updateTablesOfFigures

Location: RPE_INSTALLATION/utills/word/updateTablesOfFigures.txt

Updates all the Tables of Figures in the document.

insertOLEs

Location: RPE_INSTALLATION/utills/word/insertOLEs.txt

Embeds all the OLEs from the linked RTF files in the output document.

RPE

Location: RPE_INSTALLATION/utills/word/RPE.txt

The *RPE* macro performs the actions from all the above macros.

For your convenience *RPE* provides a Word dot file that contains all the above macros. The file is located in RPE_INSTALLATION/utills/word/RPE.dot You can use this file as a stylesheet for Word output, in which case you can use any of the above macros.

Self contained Word document

If your Word document contains links to other files, moving it to a different machine will lose the linked information. To make the document self contained you need to break all links from the document.

Word 2007:

- Office Button->Prepare->Edit link to files (the menu entry is not available if your document does not contain linked data)
- Use the "Break link" button

Word 2003:

- Edit->Links ...
- Use the "Break link" button

HTML

By default any newly created document specification contains an HTML Output. If you have deleted it you can add it at a later time. An HTML output is defined by the following properties:

Defining an HTML output

An HTML output is defined by the following properties:

Property	Description	Required
path	The full path where RPE should produce the output document. If not specified, RPE will produce the file in the temp folder (with a random generated name).	no
stylesheet	The full path to the stylesheet to be used.	no

Output

A HTML file in the specified location and a .css file with the same name. A folder named *img* will be created in the same location and will contain all the images referred in the template. If a stylesheet is specified, the stylesheet will also be copied into the same folder as the result file.

Stylesheets

You can specify any .css file to be used by the generated HTML. The generated HTML file will use that .css file along with the one generated by RPE.

Unicode data

Unicode data will appear as Unicode as long as the font used in the output can display such characters.

PDF

By default any newly created document specification contains a PDF Output. A PDF output is defined by the following properties:

Defining a PDF output

Property	Description	Required
path	The full path where RPE should produce the output document. If not specified, RPE will produce the file in the temp folder (with a random generated name).	no
default font	The font to be used when this information is not specified in the template	no
unicode output	Set this value to true if the input data is Unicode.	no

Stylesheets

The PDF output does not support stylesheets, all formatting needs to be specified in the template.

Unicode data

This value can be safely set to true as long as the following conditions are met:

- the *default font* is a True Type Font
- no template element uses a non-True Type Font family

If “*unicode output*” is set to false, any non-English characters will not be visible in the output.

If “*unicode output*” is set to true, all non-English characters will be visible in the output as long as the default font or the element’s font can render them.

NOTE The above behavior is a limitation of the technical solution for RPE 1.0 and we are looking to improve it in future versions of the tool.

XSL-FO

This output produces an XSL-FO document according to <http://www.w3.org/TR/xsl11/>

By default any newly created document specification contains an XSL-FO Output. If you have deleted it you can add it at a later time.

NOTE The XSL-FO is valuable in itself as there are several commercial and open source tools that can render it. In addition an XSL-FO document is to be used as the source for other formats. There are various tools available to convert XSL-FO to PDF and other formats.

Defining an XSL-FO output

An XSL-FO output is defined by the following properties:

Property	Description	Required
path	The full path where RPE should produce the output document. If not specified, RPE will produce the file in the temp folder (with a random generated name).	no

Output

The result consists of a single .fo file in the specified location. A folder named *img* will be created in the same location and will contain all the images referred to by the template.

Stylesheets

XSL-FO does not use stylesheets.

Unicode data

Unicode data will appear as Unicode as long as the font used in the output can display such characters.

Post-processing

As XSL-FO is an XML format you can post process it using XSLT or any other XML processing techniques.

RPE Launcher

This RPE component allows building specification for RPE document templates and generating output documents from document specifications.

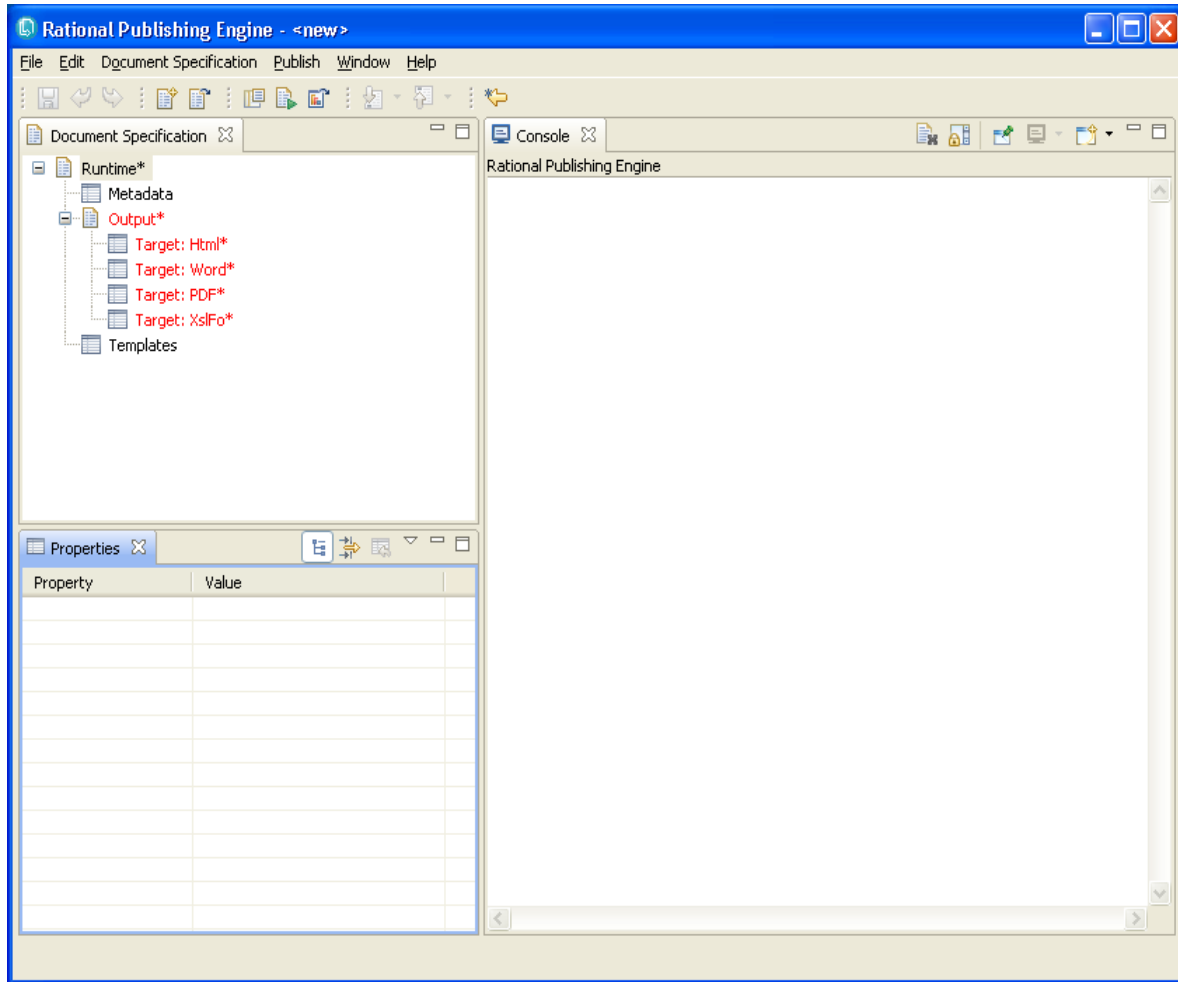


Figure 61 RPE Launcher

NOTE From RPE 1.1, the launcher does not consume licenses when started.

Defining a document specification

A document specification consists of:

- metadata – global settings for document generation
- output – the selected output formats
 - target – configures the document output (see Output sections for details)
- templates – the list of templates used by the current document specification
 - template
 - data sources – the list of data sources that need to be configured for the template. Each data source in the document specification matches exactly one data source schema in the template
 - variables – the list of variables defined in the template. Allows the user to define the values to be used when generating the output

Metadata

The metadata section of a document specification contains non-editable data generated by RPE meant to serve for informational purposes and editable data meant to configure the document generation.

Property	Description	Editable
Date	The date when the document specification was created	No
Time	The time when the document specification was created	No
Machine	The machine name on which the document specification was created	No
client	The application used to create the document specification	No
Build	The build number of the application used to create the document specification	No
Data_formatting	Flag controlling how RPE mixes template and data source formatting: Mixed – mixes source and template formatting Source – discards template formatting preserving data source formatting only Template – discards source formatting preserving data source formatting only	Yes

Missing data text	New in 1.1 This property allows customizing the text used by RPE when a data attribute is not found in the data source. Default is <i><data attribute not found></i>	Yes
Data_pattern	The format in which dates are to be rendered in the output.	Yes
Output_locale	The output locale to be used (for default date formats) if it needs to be different from the system's locale	Yes
Image max width	The maximum width of all images in the output documents	Yes
Image max height	The maximum height of all images in the output documents	Yes
OLEs as static images	Flag controlling how RPE handles OLES: <ul style="list-style-type: none"> • True – OLEs reach the output as static images • False – for Word output only OLEs are preserved in the output 	yes

NOTE None of the metadata properties are mandatory.

Multi - template document specifications

A document specification can contain as many templates as desired. A document template can be referred to more than once in a single document specification but you will have to configure its data sources for each instance. See the Input Section for details on how to configure each data source type.

Functions

Function	Location	Description
New	Menu, toolbar	Creates a new document specification
Open	Menu, toolbar	Opens an existing document specification
Close	Menu	Closes any existing documentation specification
Save	Menu, toolbar	Saves the current document specification. If this is a new document specification, the "Save As" dialog is presented to the user.
Save as	Menu	Saves the current document specification in a new location.
Synchronize Document Specification	Menu	see details bellow
Generate document	Menu, toolbar	"Runs" the current document specification, generating the output documents.
Preview document	Menu, toolbar	"Previews" the current document specification, generating the output documents. Identical to the Generate Document function but internally limits the number of results for all queries in all templates to the number given in the preference pages.
Open results	Menu, toolbar	Opens a dialog containing the list of the most recent generation results. You can click on the links to open/save the results or right click to save them.
Configure Data Source	Tree context menu	Starts the data source configuration process, specific to each data source type, as describe in the input section.

Preferences

RPE Launcher has a set of properties that can be configured via its preference pages.

NOTE The standalone Launcher and the Launcher Plugin embedded in Document Studio do not share their preferences as they are handled as different applications.

Engine preferences

Defines the RPE engine to use. By default the local RPE engine installation is used.

A remote RPE engine installation is specified with the URL to the RPE webservice.

Ex: `http://machine:8080/rpe/services/RPEService?wsdl`

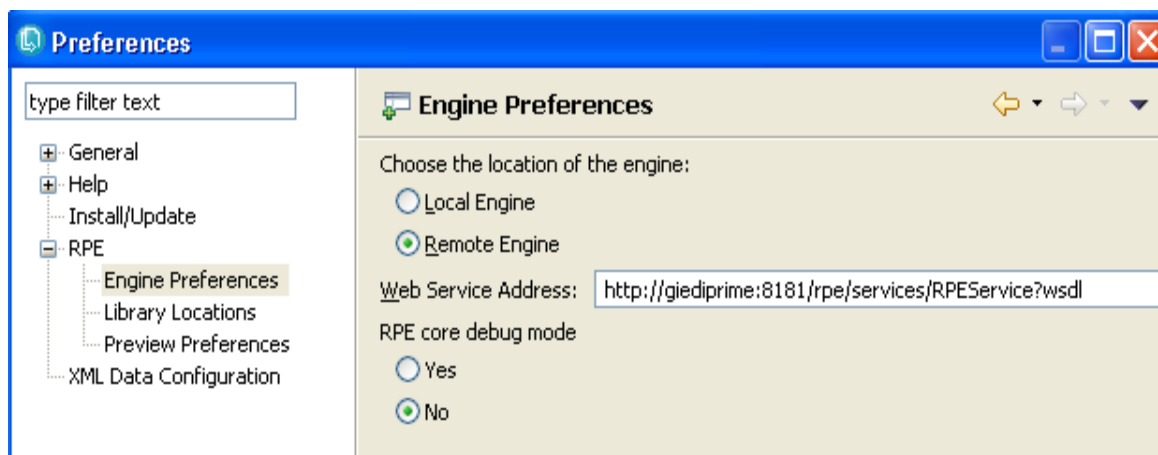
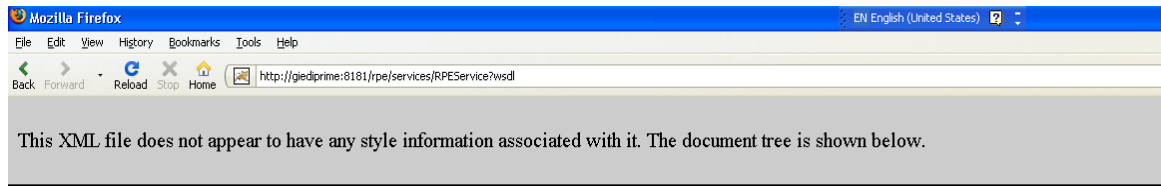


Figure 62

NOTE You need to contact your system administrator to get the RPE engine URL valid in your environment.

TIP Try the URL in a browser first. If the WSDL (XML format) is loaded successfully than you can use the same value in RPE.



```

- <wsdl:definitions name="RPEService" targetNamespace="http://ibm.com/rational/rpe">
- <wsdl:types>
- <xsd:schema targetNamespace="http://ibm.com/rational/rpe">
  <xsd:import namespace="http://www.w3.org/2005/05/xmlmime" schemaLocation="RPEService?xsd=xmime.xsd"/>
  <xsd:element name="getStatus" type="xsd:int"/>
  <xsd:element name="getStatusResponse" type="tns:statusResponse"/>
- <xsd:complexType name="statusResponse">
- <xsd:sequence>
  - <xsd:element name="message" type="xsd:string">
    - <xsd:annotation>
      <xsd:documentation> Details on the progress. </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="engineStatus" type="xsd:int"/>
</xsd:sequence>

```

Figure 63 testing the service URL in a browser

Shared template library

Starting with RPE 1.1 there, the template library is no longer kept in a system variable but stored in the application's preferences. Initially the library locations are the example folders from RPE's installation folder.

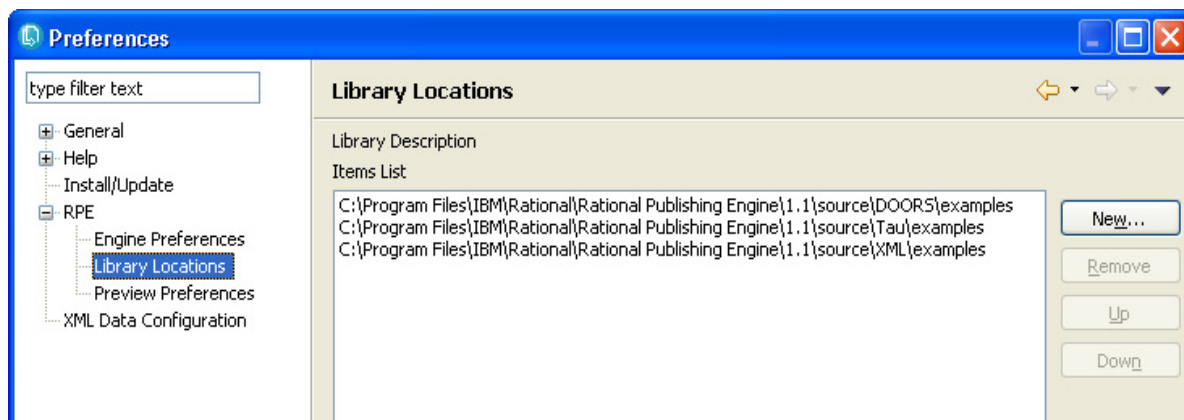


Figure 64 Library locations

You can add and remove any of the library locations. The library location can be folders on the file system (local machine or shared network drives) or URLs to an XML file containing the library description.

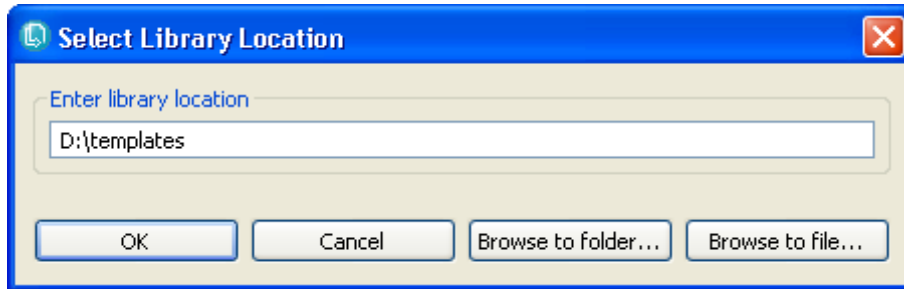


Figure 65 adding a folder from the file system as a shared library location

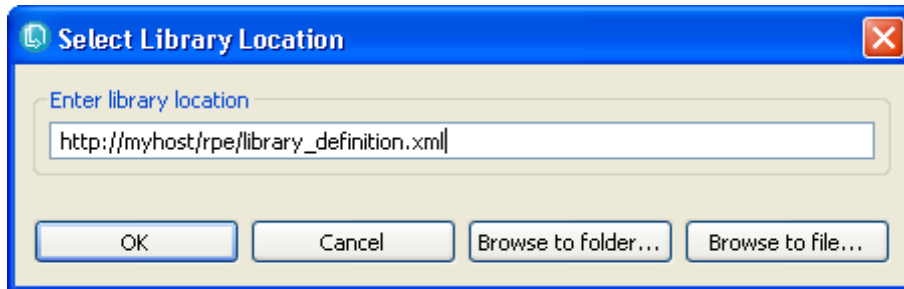


Figure 66 adding a remote library definition file

NOTE The structure of the library definition file is described in the Annex.

Preview preferences

Controls the maximum number of elements per query when doing a preview

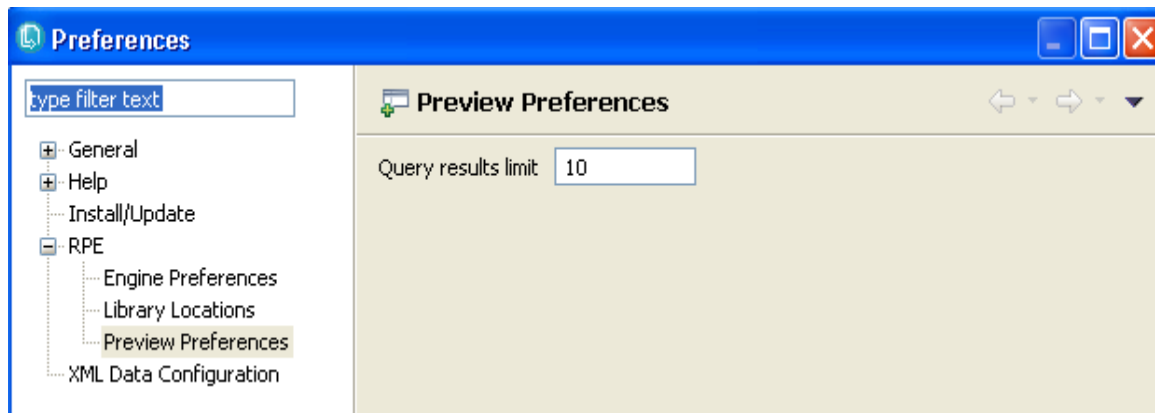


Figure 67 Preview preferences

Synchronizing document specifications with templates

If data source schemas are added to a template after the template was assigned to a document specification, the document specification is considered out of synch with the template.

This function scans all the document templates assigned to the current document specification, and adds Data Sources and variables for all the Data Source Schemas and variables from templates.

- Existing data source – schema associations are preserved
- Data sources for which a schema association no longer exists are removed
- New data sources are added for new schemas in the templates
- New variables defined in the templates become available in the document specification
- Variables deleted from the template are also deleted from the document specification

Generating a document

If a document specification is loaded in the Launcher you can start the document generation process. While the output is being generated, a progress dialog is displayed.

NOTE RPE will start the document generation process even if the document specification is incompleteley configured.

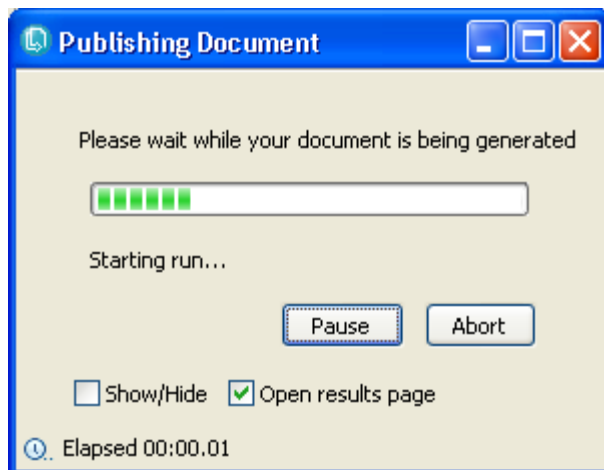


Figure 68 Generate document

RPE does not know beforehand the size of the data that will be processed and it cannot offer an accurate estimation of the time needed. In consequence the progress bar is cyclic. The messages displayed in the progress bar window and in the console view are good indicators on the status of the document generation.

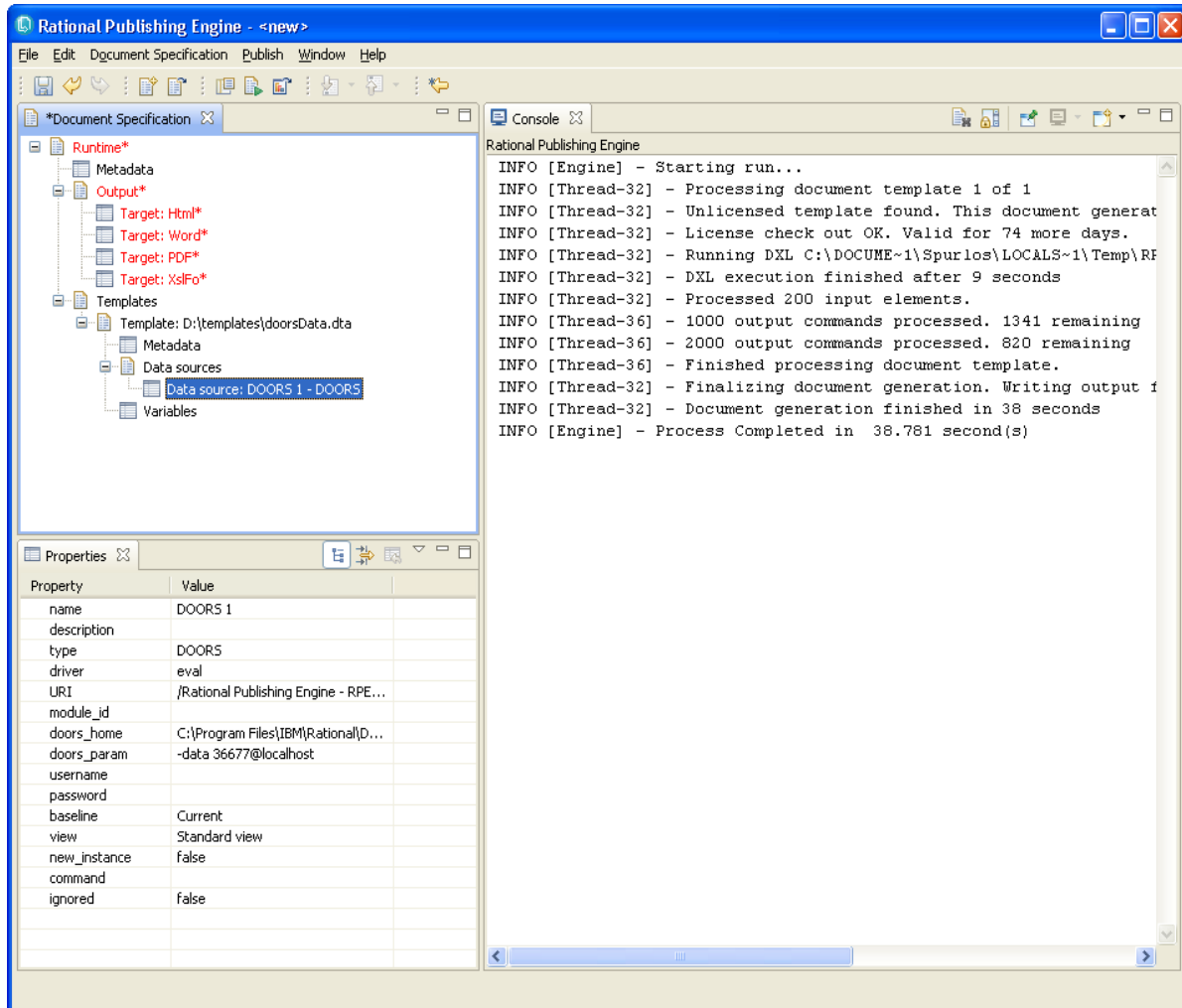


Figure 69 Console view

Pause and abort

At any time during the document generation you can request RPE to temporarily suspend the document generation process or to cancel it altogether.

NOTE In the current RPE build for DOORS extractions the document generation will wait for DXL code to finish before aborting the process. This means that the process will not instantly pause/abort if DXL execution is in progress.

Results dialog

This window becomes available after at least one document generation process has been completed. The window displays an HTML page listing the documents generated by RPE. You can click on any of the links to open the output document (if an association is made for the file extension and the required software is installed on your machine). You can also right click to save the output in a different location.

Local engine installation

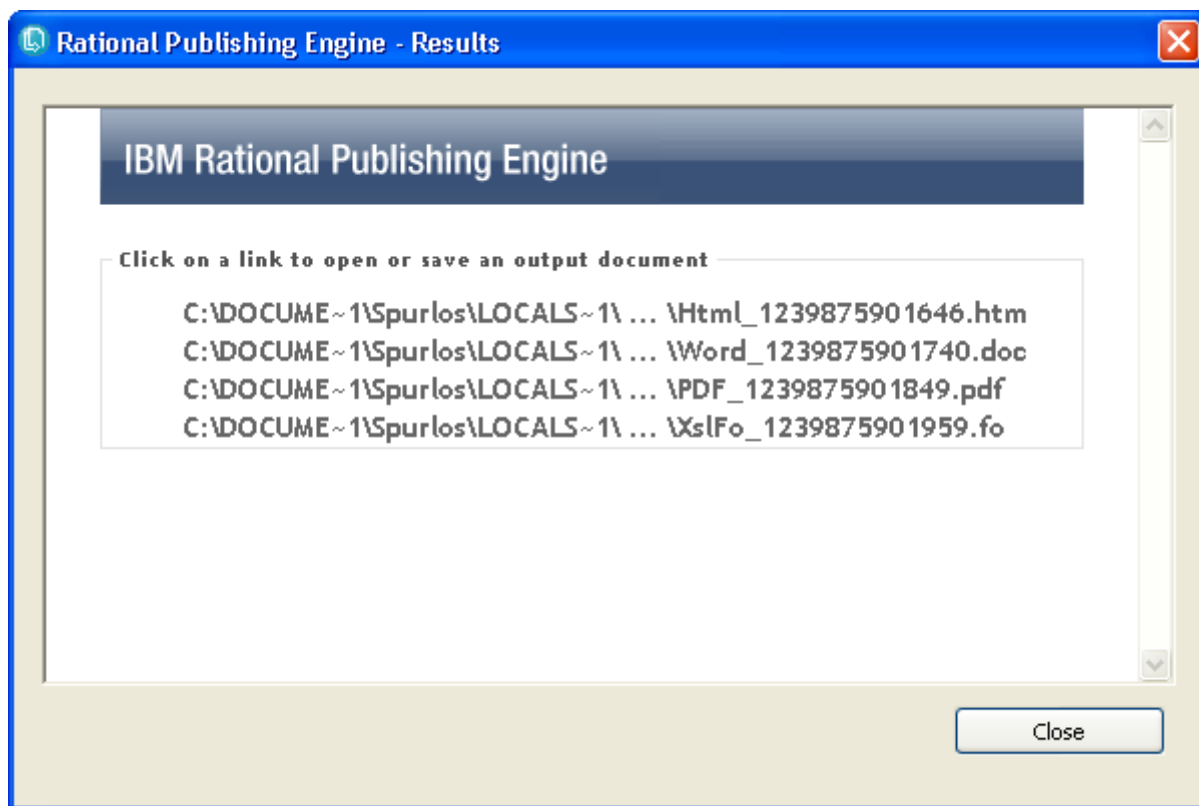


Figure 70 Results window for local engine installation

NOTE If the output path is not specified, the output is considered to be un-configured and marked in red. RPE will generate unique names for each output type in the current user temp folder.

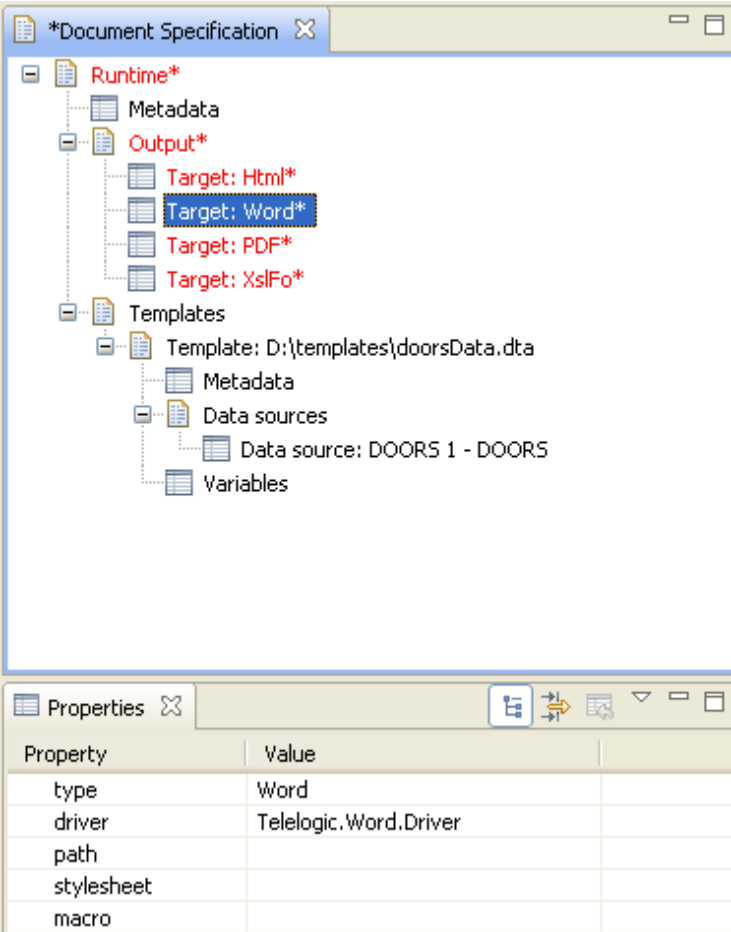


Figure 71 Output not configured

NOTE If you choose to save the HTML output then you need to manually copy the additional files generated (the img folder and the stylesheets).

Remote engine results

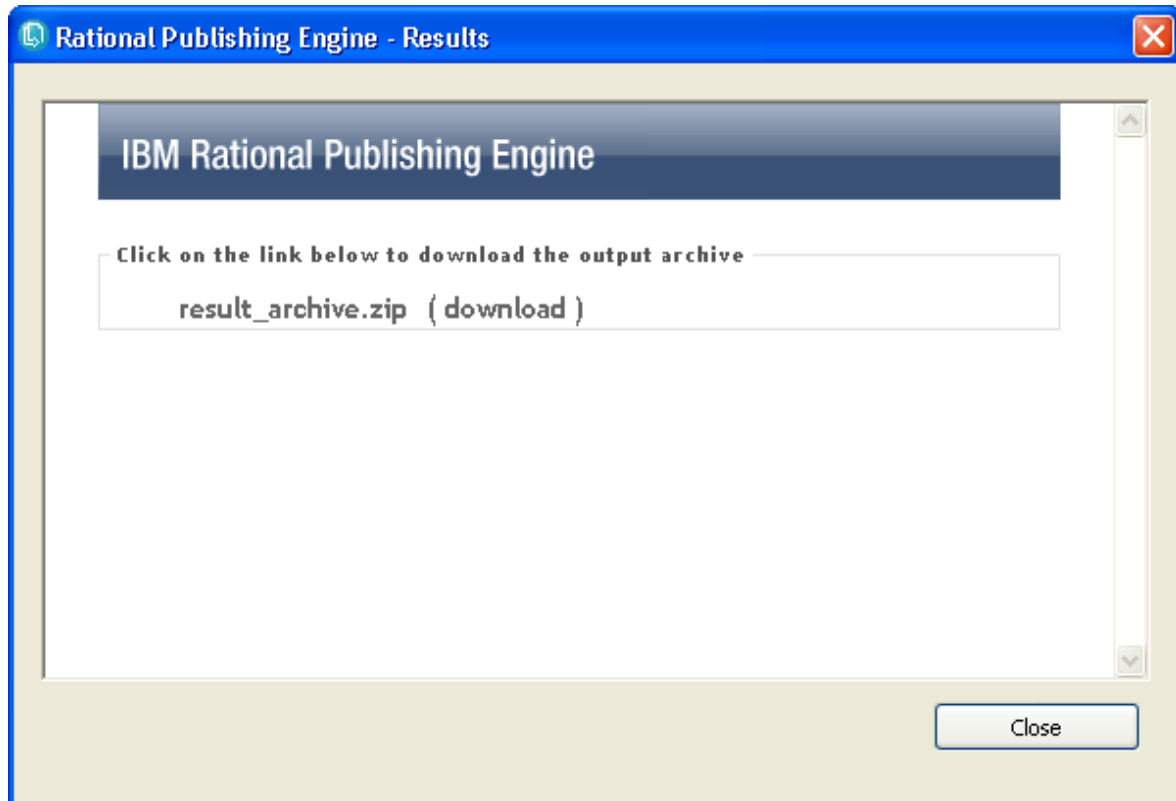


Figure 72 Remote engine results

The archive contains all the output files.

Command line switches

RPE Launcher exposes a command line interface that can be used to accomplish for various tasks. Please consult the “RPE API” manual.

RPE Publish Wizard

The RPE publishing process can be started from the 'Publish' > 'Generate Document...' menu entry, available both in RPE Launcher and the Document Studio. This will start a wizard, which can also be started from the DOORS and Tau add-ins.

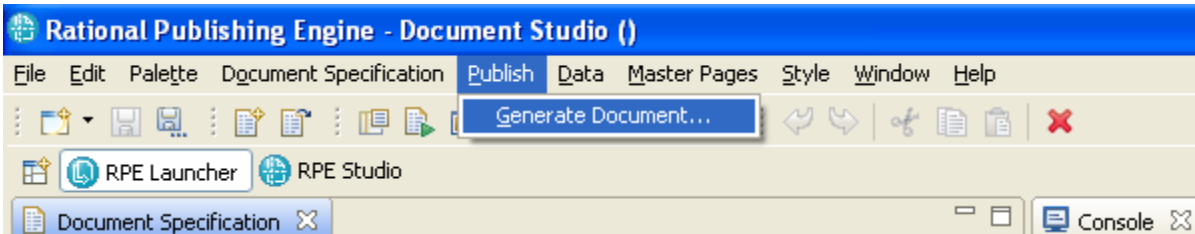


Figure 73 Starting the Publish Document Wizard from the Document Studio

The first page is a dialog welcoming you to the Publish Document Wizard. Click 'Next' to continue.

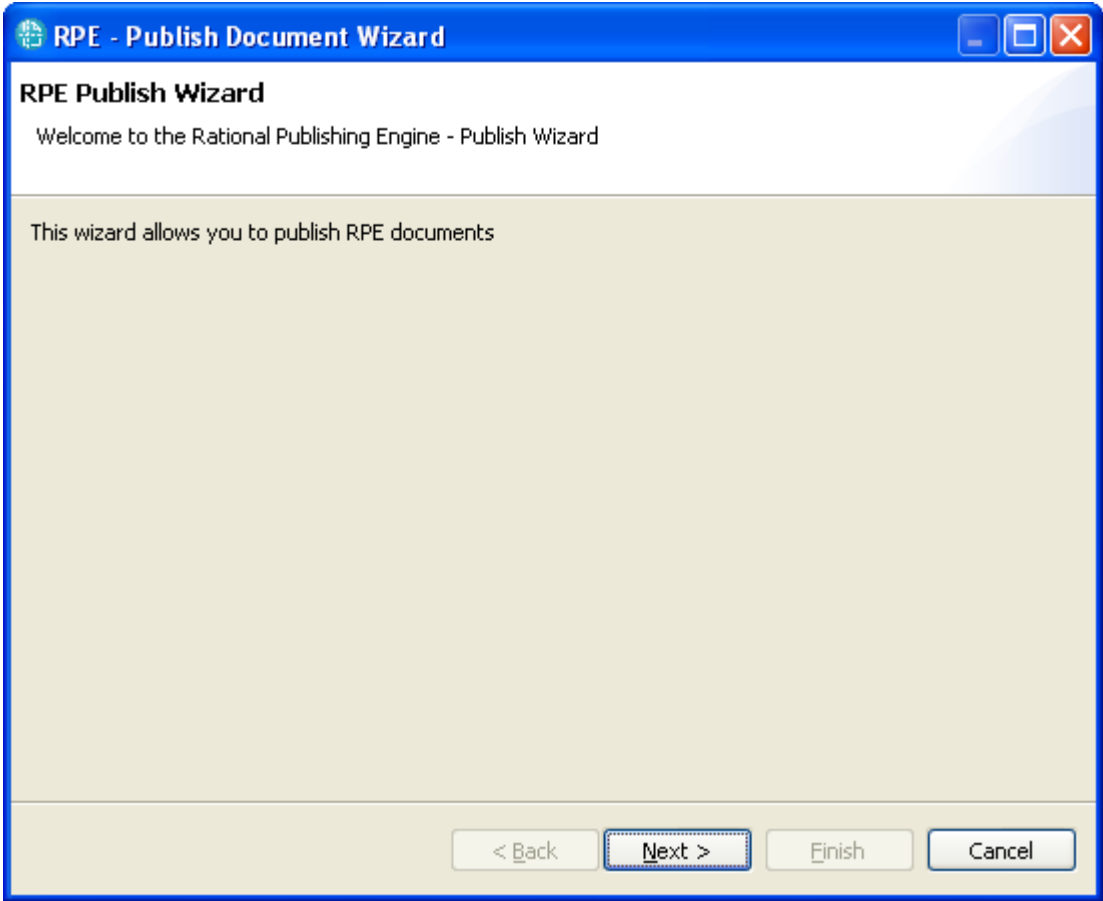


Figure 74 The Welcome page of the wizard

The second page displays the Document Specifications and Templates that are available directly from the RPE Library. You can either select a document in the file tree or browse to a different

location. Also, a URL to a file can be typed. Once the Document Template or Document Specification selection is complete, click 'Next'.

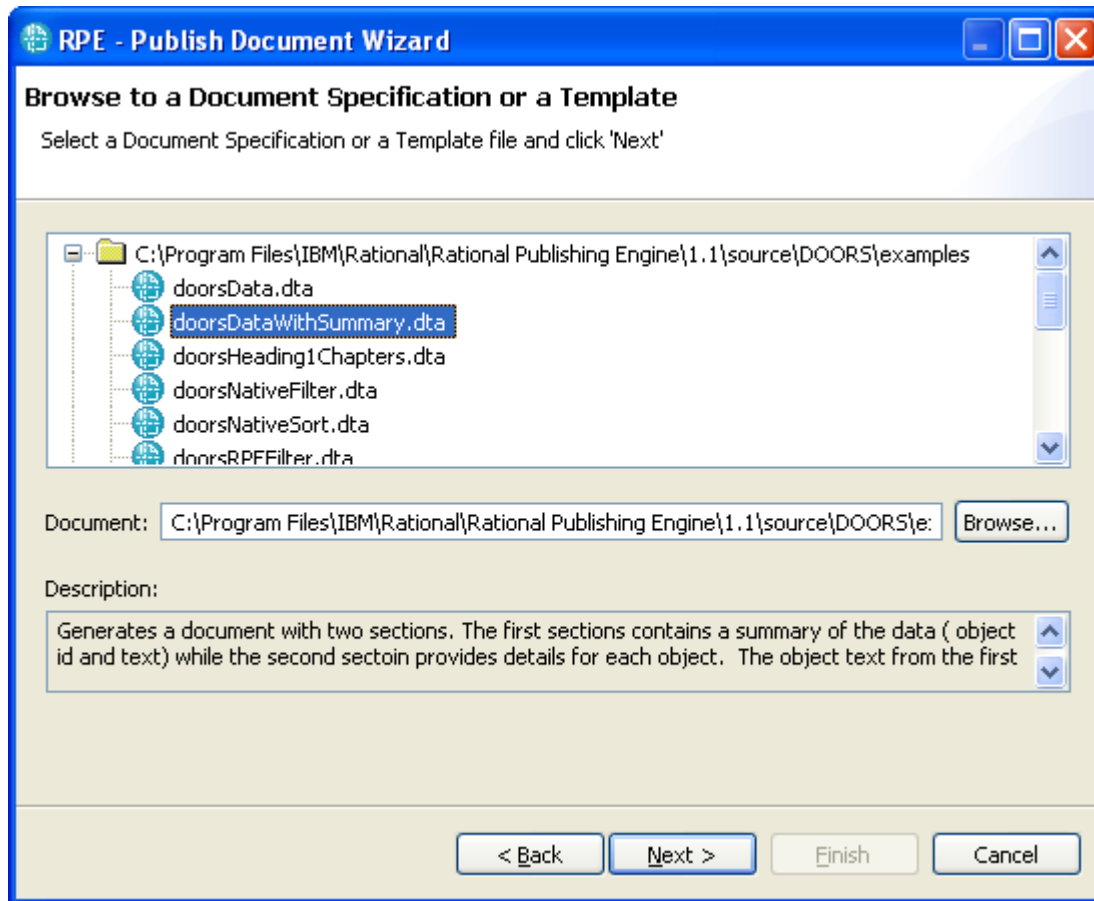


Figure 75 Template or Document Specification selection page

In the next dialog a table is shown with all available data sources, as shown in Figure 76 Data source configuration page76. The template file, the name and the type of the data source are shown, along with a field that allows the data source to be configured. Clicking on the '...' button in the 'Configuration' column starts the configuration tool specific for each data source. Shown in Figure 77 The configuration tool for a DOORS data source77 is the configuration tool for a DOORS data source.

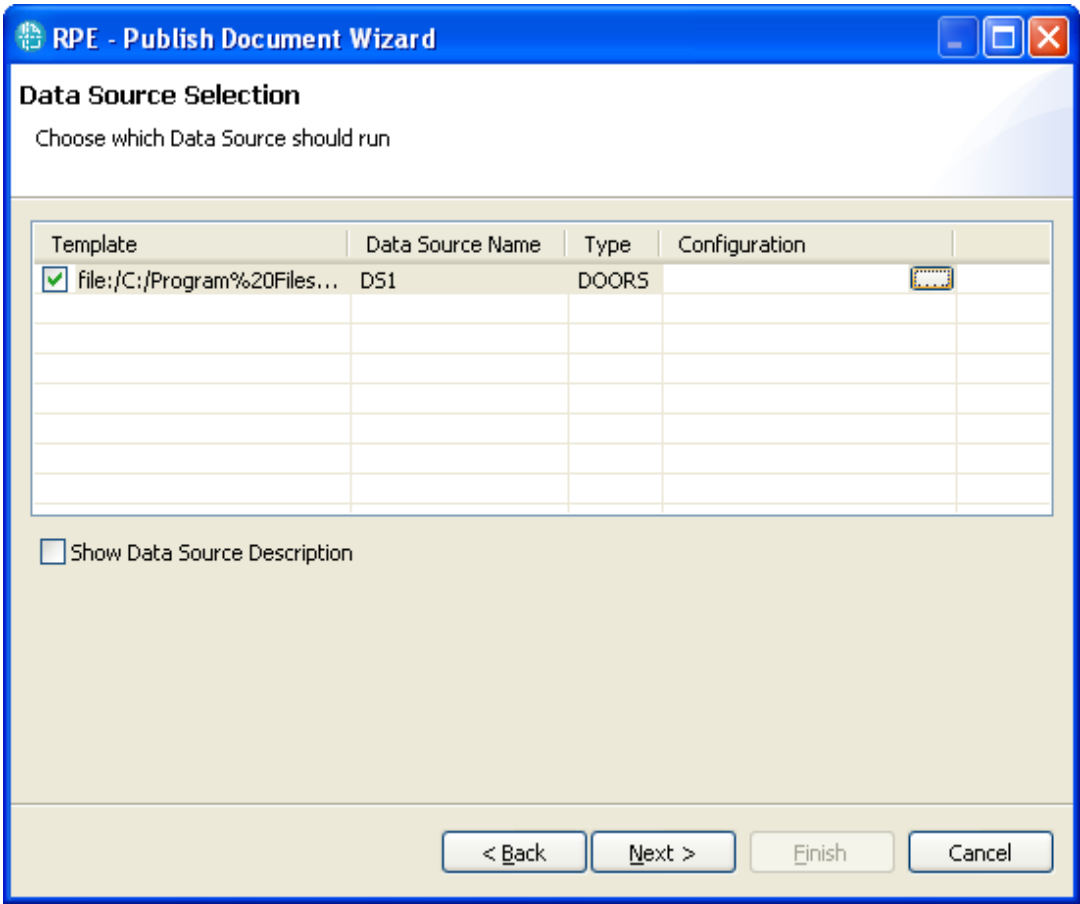


Figure 76 Data source configuration page

An optional data source description appears if the 'Show Data Source Description' checkbox is checked.

Clicking the configuration button will open the appropriate "Data Source Configuration" wizard for the selected data source. See the Data section for details.

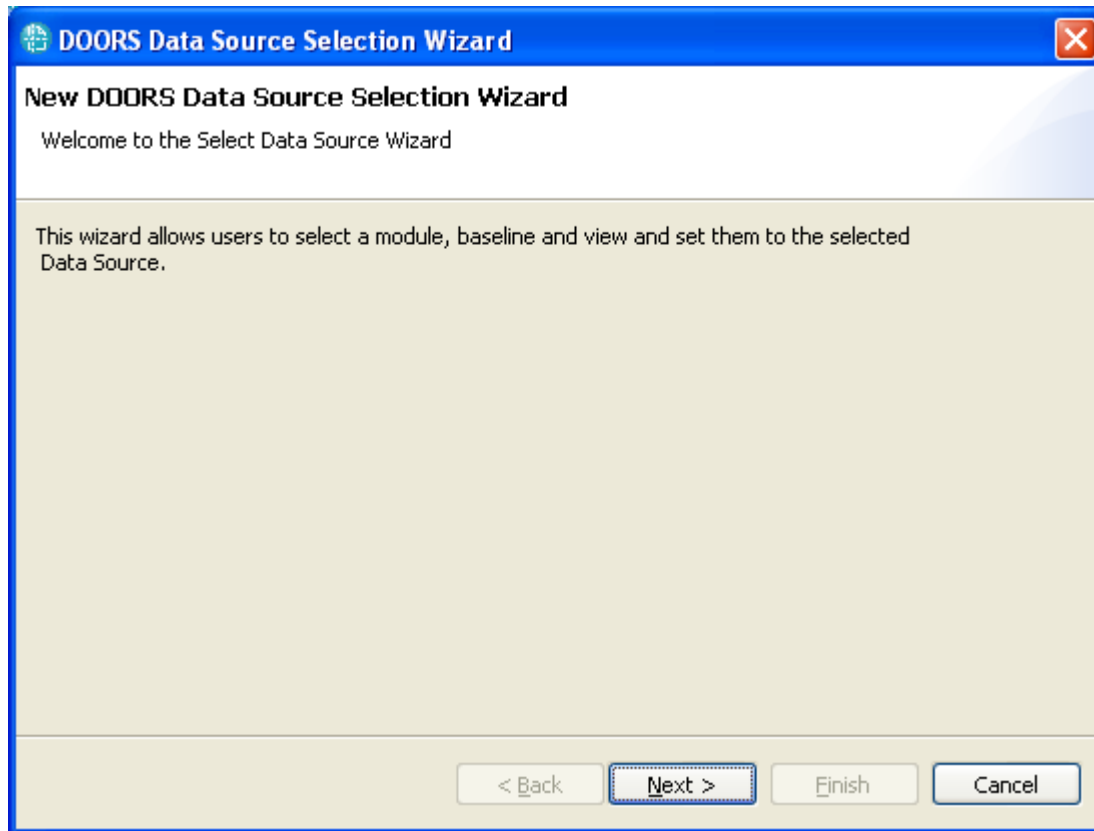


Figure 77 The configuration tool for a DOORS data source

If the selected Document Template has external variables, the next dialog contains a table with these variables, as shown in Figure 78 Variables value page78. The template file, the name and the default value of the variable are displayed, along with their user value, which is editable. Click on the '...' button in the 'User Value' column to edit the variable's value.

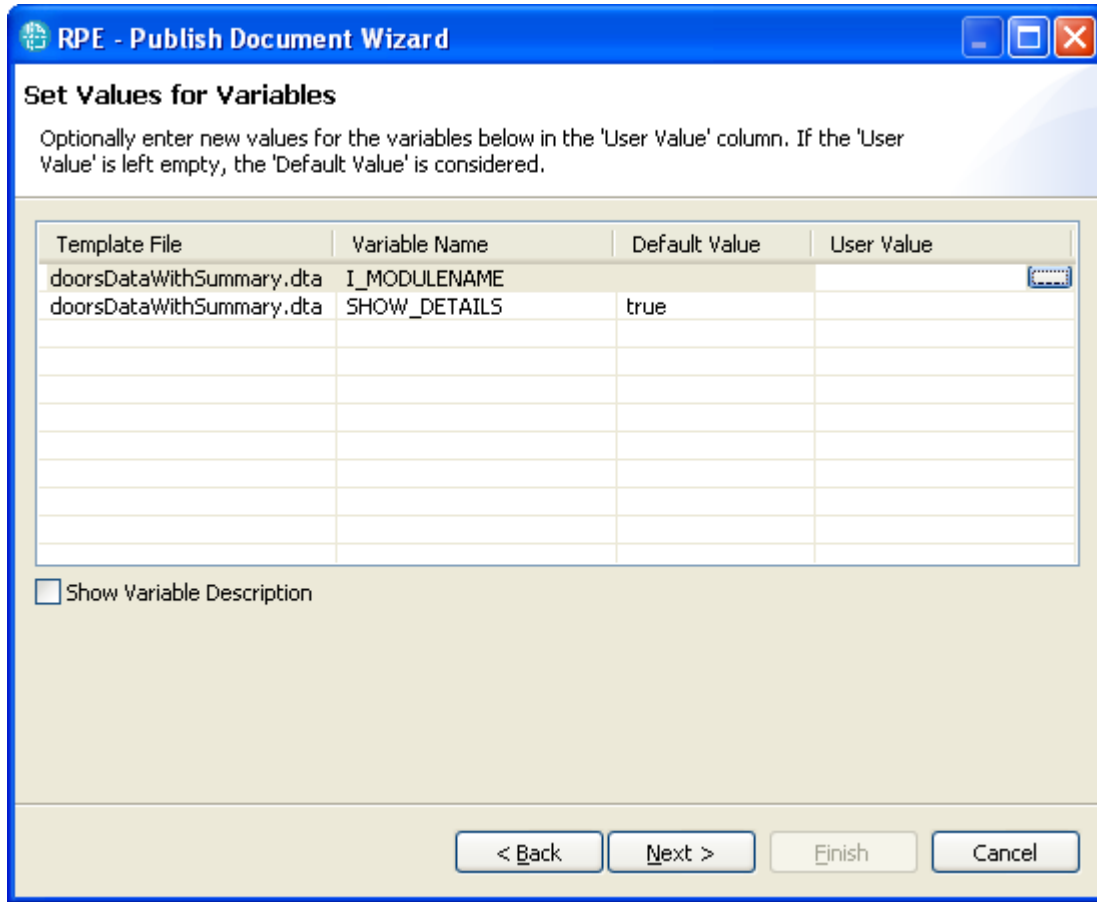


Figure 78 Variables value page

An optional variable description appears if the 'Show Variable Description' checkbox is checked.

The next page allows different output types to be added, removed and configured.

For the output types that support stylesheets, they can be selected by clicking on the '...' button from the second column of the table. If left empty, no stylesheet file will be used.

The output file path is shown in the third column. It can also be specified by browsing to a different location. If left empty, the file will be generated in a temporary location.

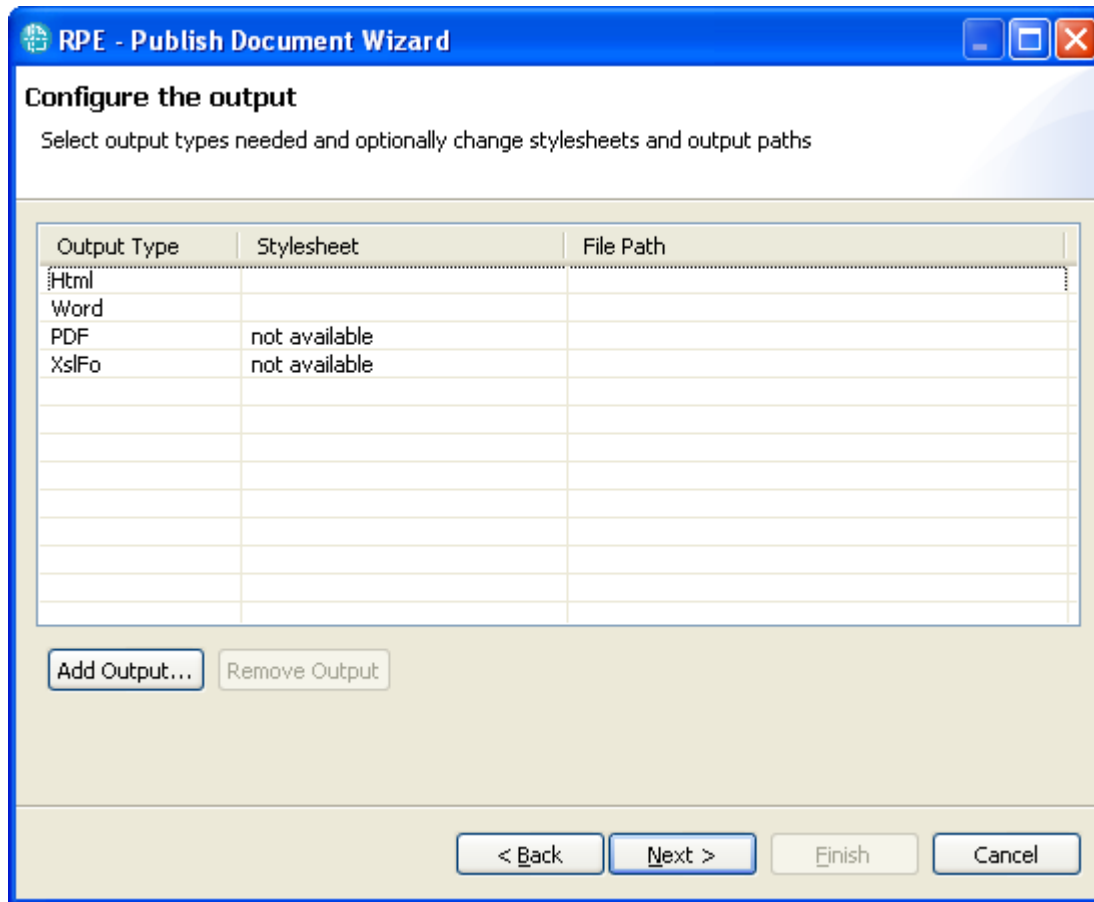


Figure 79 Output configuration page

The next dialog allows a Document Specification containing all the selections to be saved to a custom location. If the 'Publish now' checkbox is checked, the publishing process starts after the wizard closes.

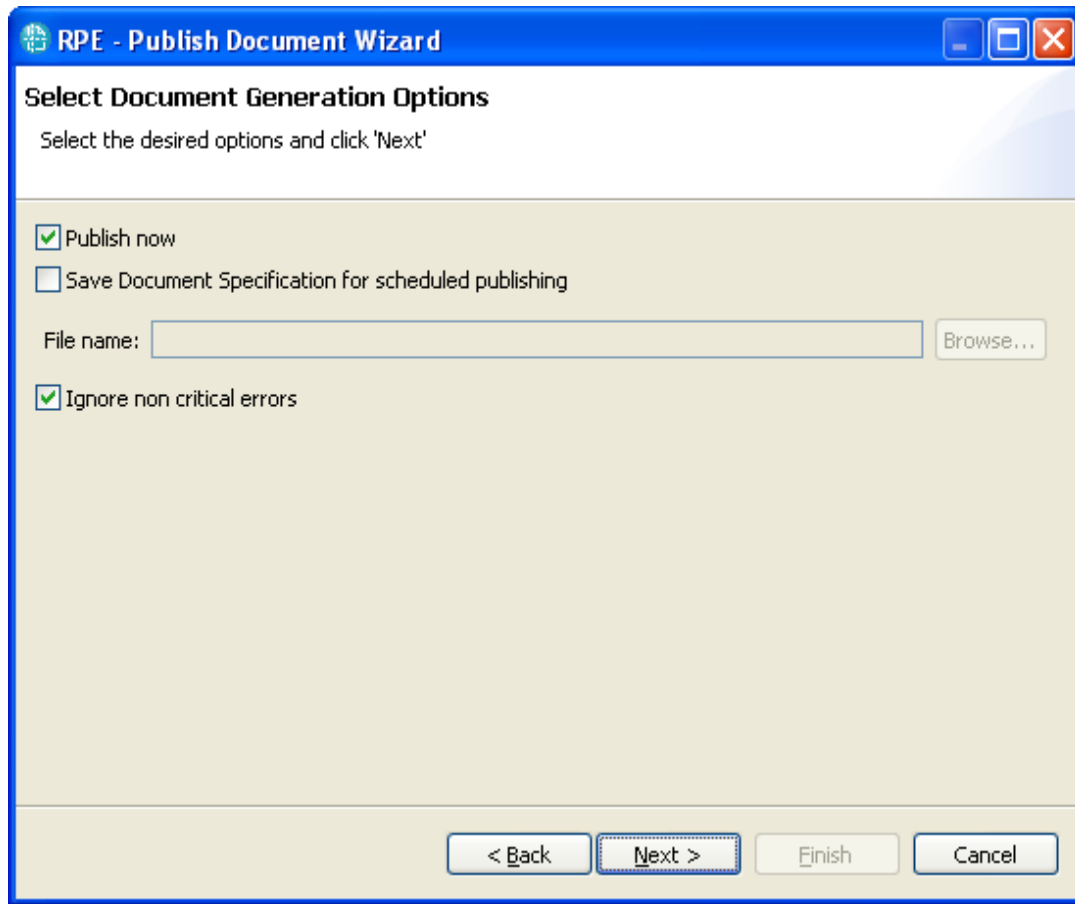


Figure 80 Publishing options page

The last page shows a summary of all the selections made in the wizard. Review this page and click 'Finish' to load the Document Specification in the RPE Launcher and have it published or saved, depending on previous selections.

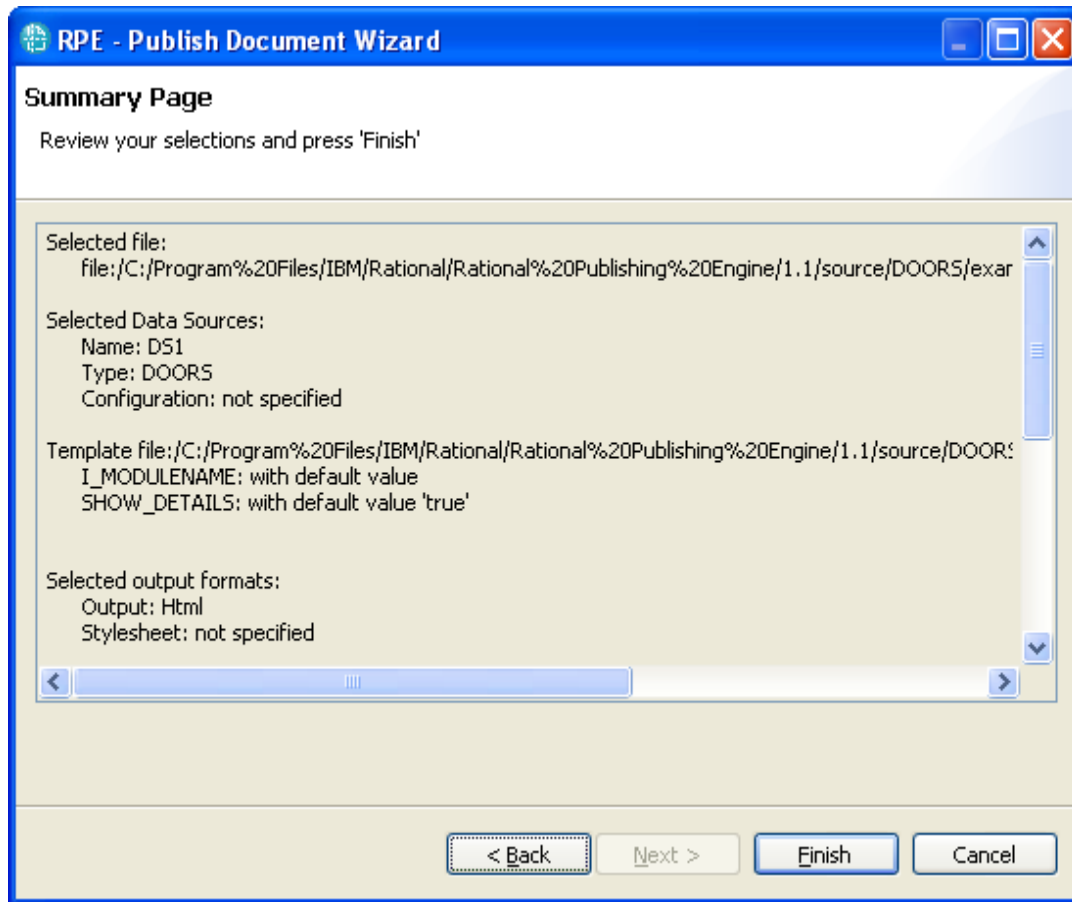


Figure 81 Summary page

RPE Document Studio

The Document Studio component of RPE allows designing document templates.

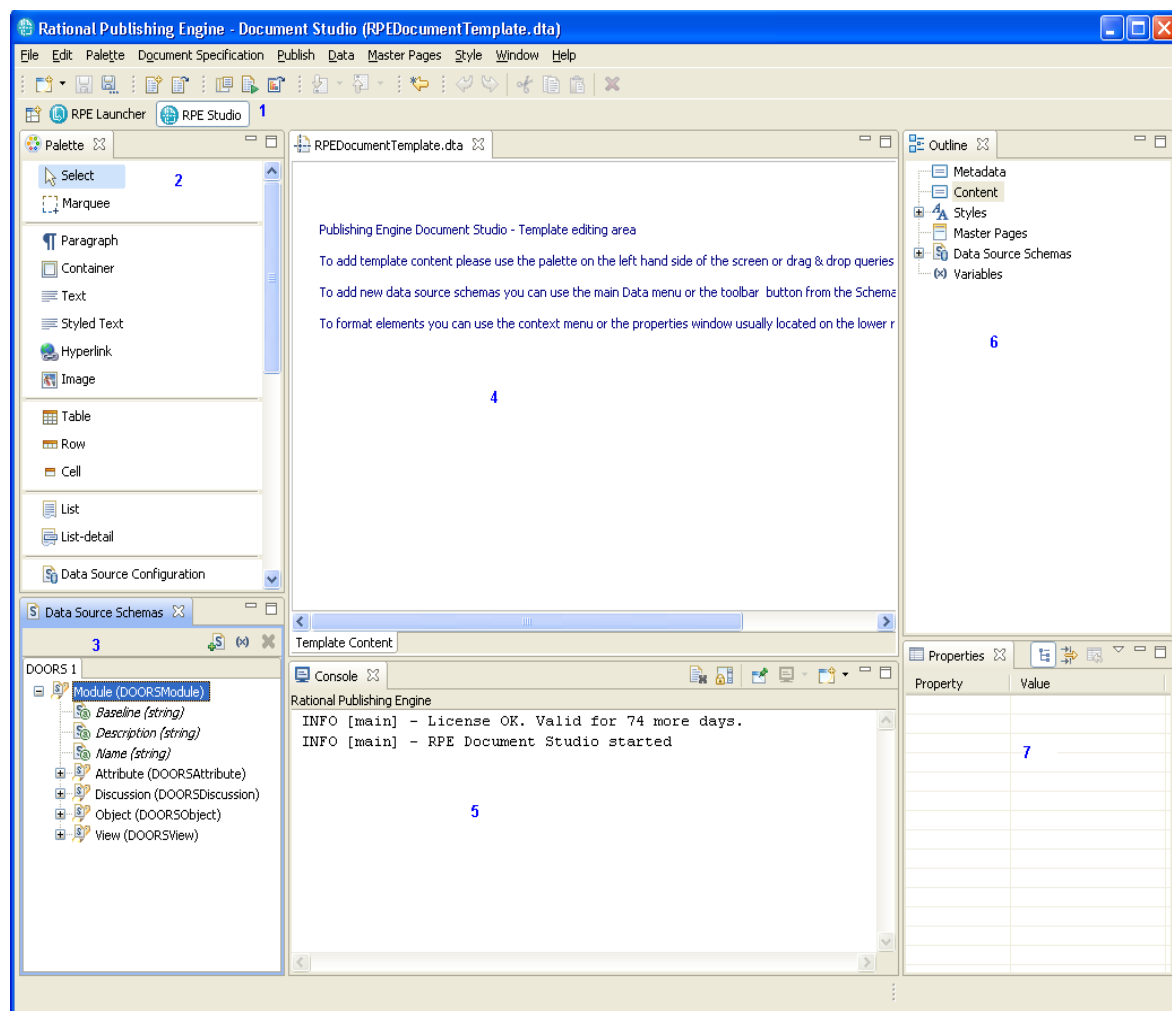










Figure 82 RPE Studio









Legend








- 1 – Perspective Selector
- 2 – Palette
- 3 – Data source schema view
- 4 – Editor area
- 5 – Console
- 6 – Outline
- 7 – Property view





Template elements

This section briefly describes the elements you can use when building a template. For a detailed description of elements and their properties, please consult the annex.

Icon	Name	Description	Container	Hosts data
	Paragraph	A container for other elements. Cannot contain data directly. Adds a carriage return into the output then renders its child elements.	yes	no
	Container	Similar to paragraph with 2 exceptions: cannot be styled does not add a carriage return into the output * When used in a table, the container can host only row elements. * When used in a row, the container can host only cell elements. * When used in a list, the container can host only list-detail elements.	yes	no
	Text	The basic block for building templates. Renders its content with the specified formatting. The formatting is applied to the whole content. Can contain data but cannot contain other elements.	no	yes
	Styled text	A block of text that can have basic formatting at word level. The text is static.	no	yes
	Hyperlink	Adds a hyperlink in the output. The hyperlink can point to a location inside the document or to an external location.	no	yes
	Image	Inserts an image into the output. The image can be taken from the template or have its path calculated at runtime.	no	yes
	Table	A table can contain rows or Container elements. Creates a table in the output then renders its child elements, rows or containers.	yes	no
	Row	Creates a row in the current table (direct child or inside a container in a table), then renders its child elements (cells).	yes	no

	Cell	Creates a cell in the current row of the current table then render its child elements (any).	yes	no
	List	Creates a list in the output then renders its child elements (list items).	yes	no
	List detail	Creates a list item in the current list in the output then renders its child elements (any).	yes	no
	Data source configuration	See “Data Source Configuration” sub-section for details.	no	no
	Include file	Includes the specified file in the output as an “INCLUDETEXT” field. Supported on Word output only. You need to update all fields in the Word document in order to see the included file.	no	yes
	Footnote	Creates a footnote in the current output page. The text of the footnote is the “content” of the footnote element. Supported on Word output only.	no	yes
	Region	Defines a region that will host elements. If an element has the “region” property specified, it will be rendered in the specified region instead of the current position in the document. You can define elements inside the region.	no	no
	Bookmark	Defines a bookmark in the document. The name of the generated bookmark is the “content” of the bookmark element. When generating the documents, RPE will generate a unique name for each bookmark. The name is based on the name provided at design time and a unique identifier generated at runtime.	no	yes

	Comment	<p>Adds a comment element into the output. The form and shape of the output comment is specific to the output format. The comment's text is taken from the template element content.</p> <p>Supported on Word and PDF.</p>	no	yes
	Page Break	Adds a page break to the output.	no	no
	Section Break	Adds a section break to the output.	no	no
	Table of Contents	Adds a table of contents to the output.	no	no
	Table of Tables	<p>Adds a table of tables to the output. You need to add Table Captions in your document template for this table to contain anything.</p> <p>Supported on Word output only.</p> <p>You need to update all fields in the Word document in order to see/update the value of the field.</p>	no	no
	Table of Figures	<p>Adds a table of figures to the output. You need to add Figure Captions in your document template for this table to have contain.</p> <p>Supported on Word output only.</p> <p>You need to update all fields in the Word document in order to see/update the value of the field.</p>	no	no
	Field	<p>Adds a generic Word field element. You can type any valid Microsoft Word code in the "field code" property.</p> <p>Supported on Word output only.</p> <p>You need to update all fields in the Word document in order to see/update the value of the field.</p>	no	no

	Page Number	Adds a page number in the output. Supported on Word and PDF output only.	no	no
	Total Pages Number	Adds the total number of pages in the output. Supported on Word and PDF output only.	no	no
	Table Caption	Adds a table caption. While you cannot specify dynamic content into a table caption, any subsequent text element will be concatenated to it. Supported on Word output only. You need to update all fields in the Word document in order to see/update the value of the field.	no	no
	Figure Caption	Adds a figure caption. While you cannot specify dynamic content into a figure caption, any subsequent text element will be concatenated to it. Supported on Word output only. You need to update all fields in the Word document in order to see/update the value of the field.	no	no

Components

Palette

The palette displays the elements (tools) that can be used in the template. Two special tools are provided:

- Select – allows selecting elements in the editor area. A single click selects the clicked element while Ctrl + Click performs adds the clicked element to the existing selection
- Marquee – allows the selection of multiple elements

To add a template element, click on the appropriate entry in the palette and then click in the template in the desired location.

By default the palette is sticky, meaning that a palette element remains selected until you explicitly select a new tool. You can change the way the palette works from the preferences menu.

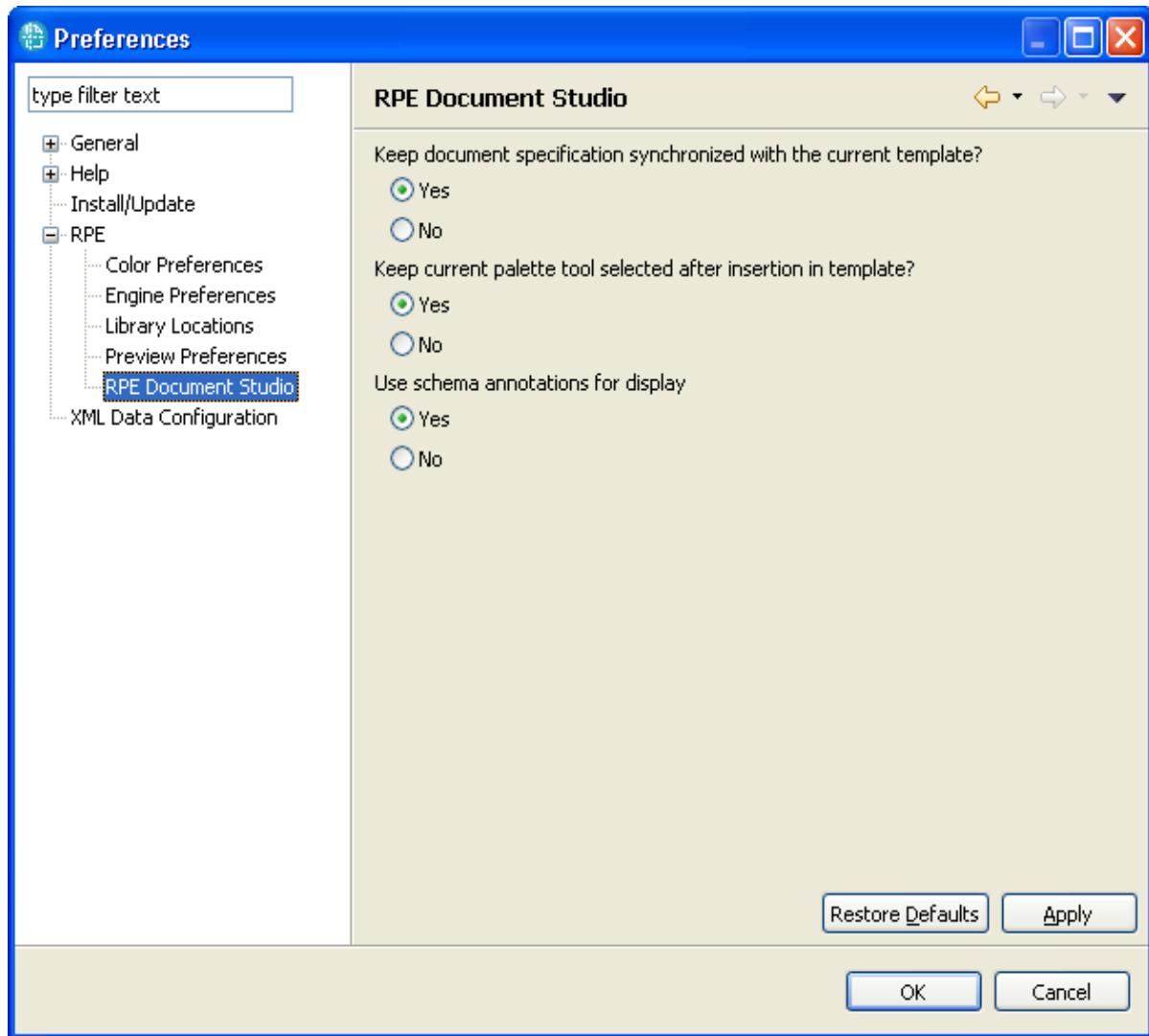


Figure 83 Configuring palette behavior from Studio preferences

Schema view

This view displays all the data schemas in the template. Each schema is displayed in its own tab in this view.

Editor area

This is the core of Document Studio. The template's visual content and the master pages' content are created here. There is one tab for the template's content and one for each master page in the document template.

Outline

Displays, in a tree structure, all the visual and non visual content.

Properties

Displays and allows changing the properties of the selected element in the editor area or the outline.

Content

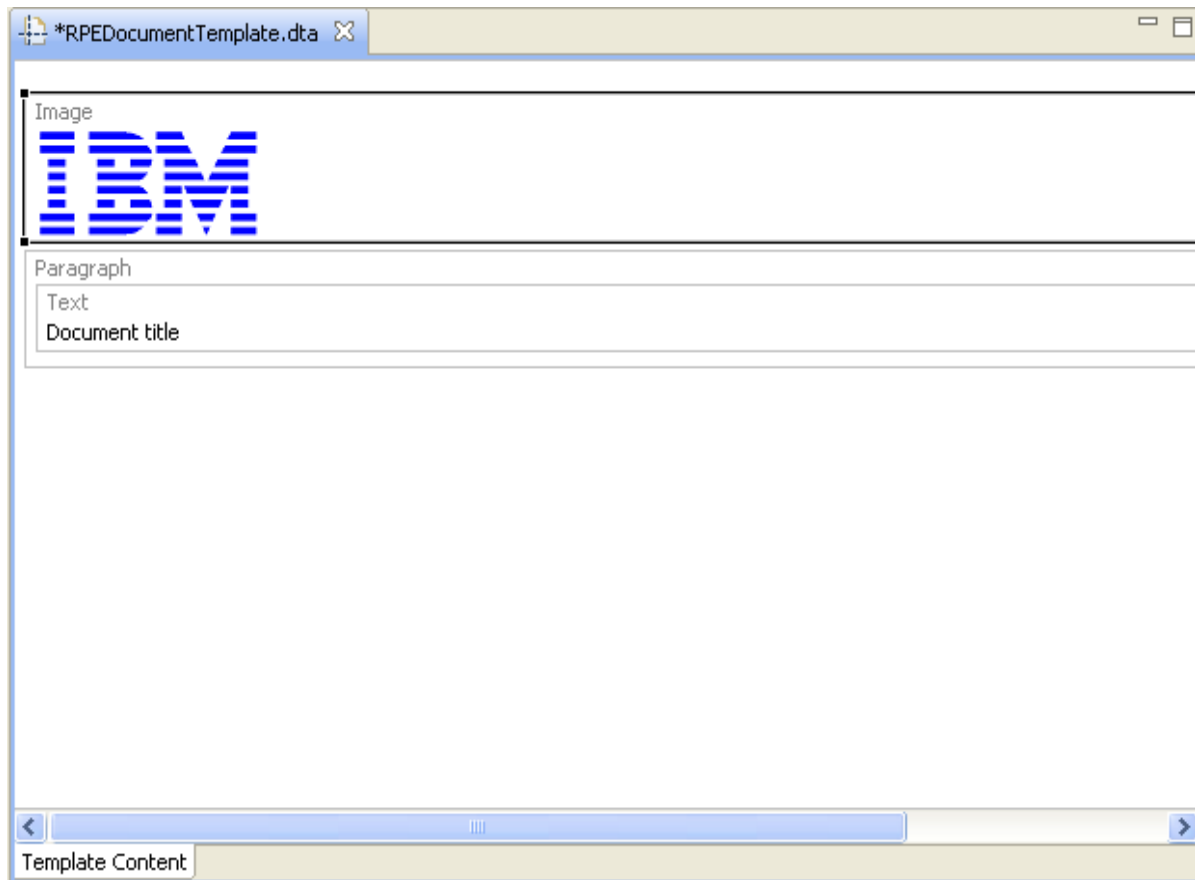


Figure 84 Editor Area

Adding elements

You add elements in the editor area by selecting the appropriate tool in the palette and clicking in the desired location in the editor area. Each element has its own set of rules when it comes to what elements it can contain and be contained by. See the annex for details.

Naming elements

With RPE 1.1 you can provide symbolic names and descriptions to the template elements. The name and description are used exclusively for display purposes. The name and description are not used in the document generation process.

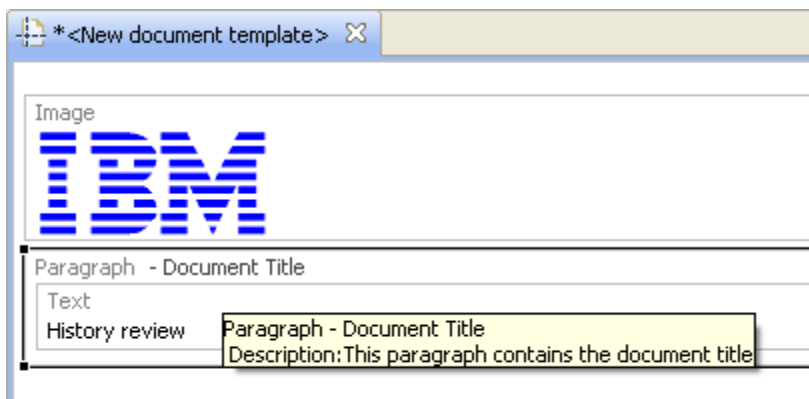


Figure 85

The name and description are set from the property page.

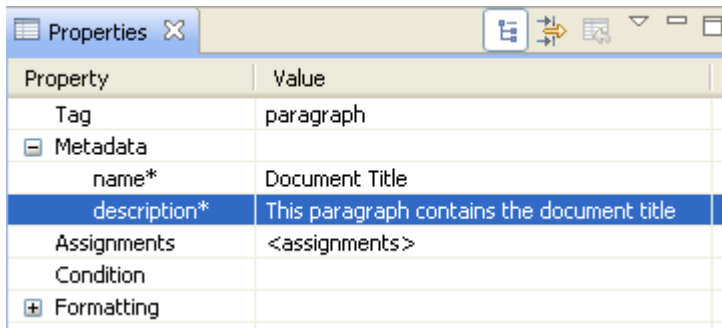


Figure 86

The name and description are visible in the element figure, its tooltip and also in the outline page.

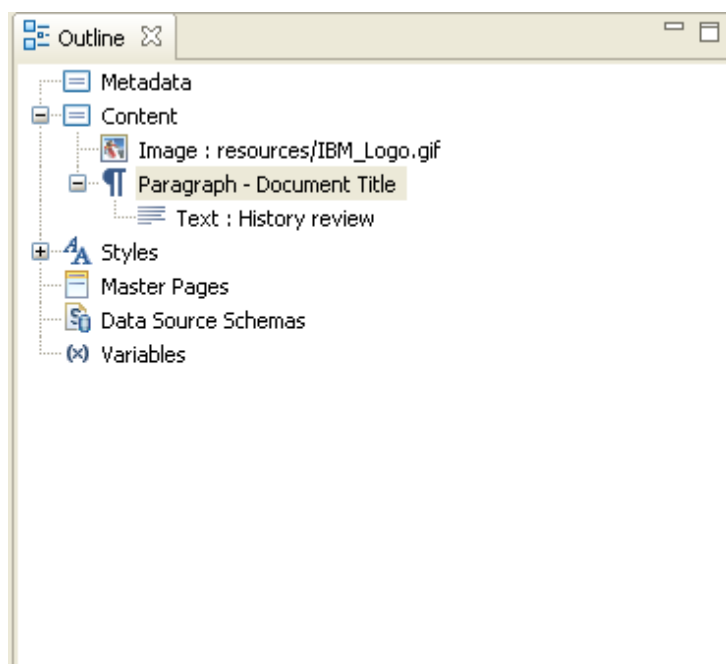


Figure 87

Editing element properties

All the element's properties can be edited through the property page. There are two different types of properties:

- data related properties – related to queries set on the element, conditions etc
- formatting properties – define the way the element will be rendered in the output format

The formatting properties are grouped further as font properties (font family, color, size), alignment properties (centered), border properties etc.

NOTE The available formatting properties depend on the element type.

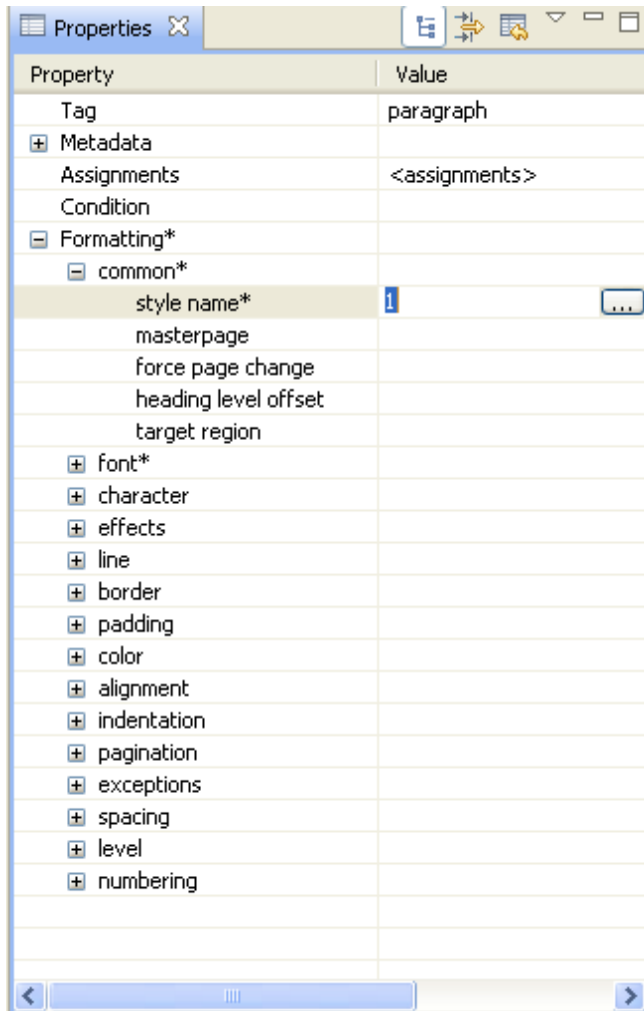


Figure 88 Properties page for a paragraph element

NOTE The modified properties and their direct and indirect parent groups are marked with a * to help identifying changed properties.

Formatting properties

Some of the properties (such as font family) accept any value provided by the user. Other properties (such as paragraph alignment) accept values from a list of possible options. RPE Studio assists the user in providing the right value through a tooltip describing the property value domain and through the property value editor.

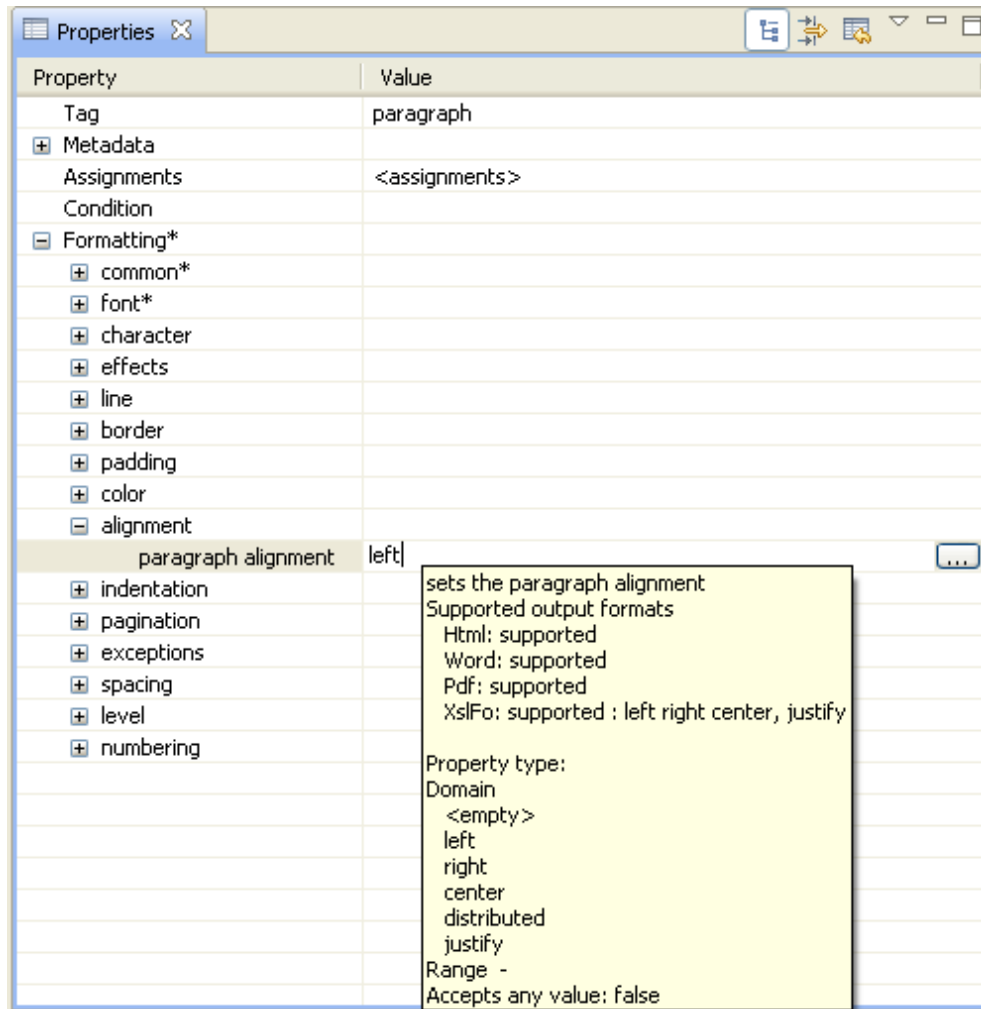


Figure 89 Tooltip in properties page

The tooltip presents a short description of the property purpose, the support on the various output formats and the list of valid properties.

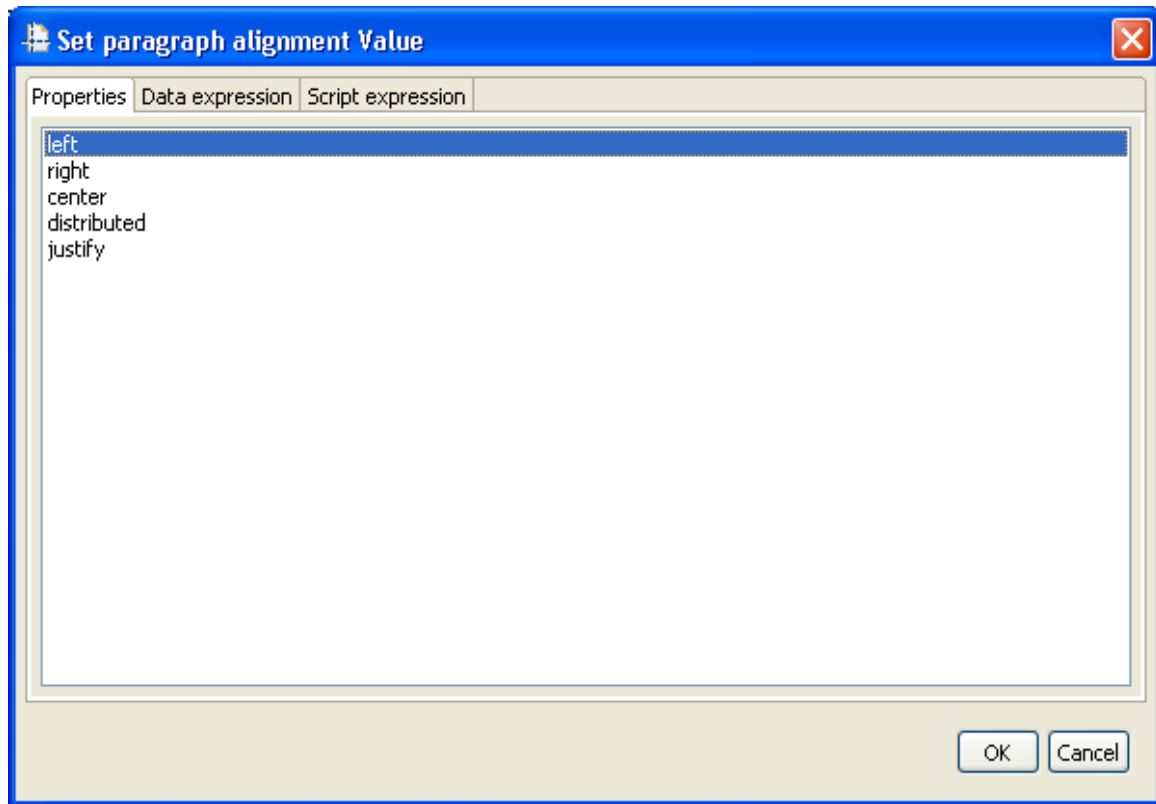


Figure 90 Value editor

NOTE You can type the value directly in the property editor or select it from the expression editor list.

NOTE Some values accept both user-provided values and values selected from the list.

NOTE Invalid properties values (typed by the user) are reported as warnings by Studio and ignored during the generation process. RPE will not discard invalid values.

Calculated properties

In a document template you can define static values for properties (left, right etc) as well as dynamic values. The dynamic values are either data attributes extracted from the data sources, template variables or the result of evaluating JavaScript expressions. See the data section for more details on calculated properties.

In the example bellow the color property is calculated based on data source values.

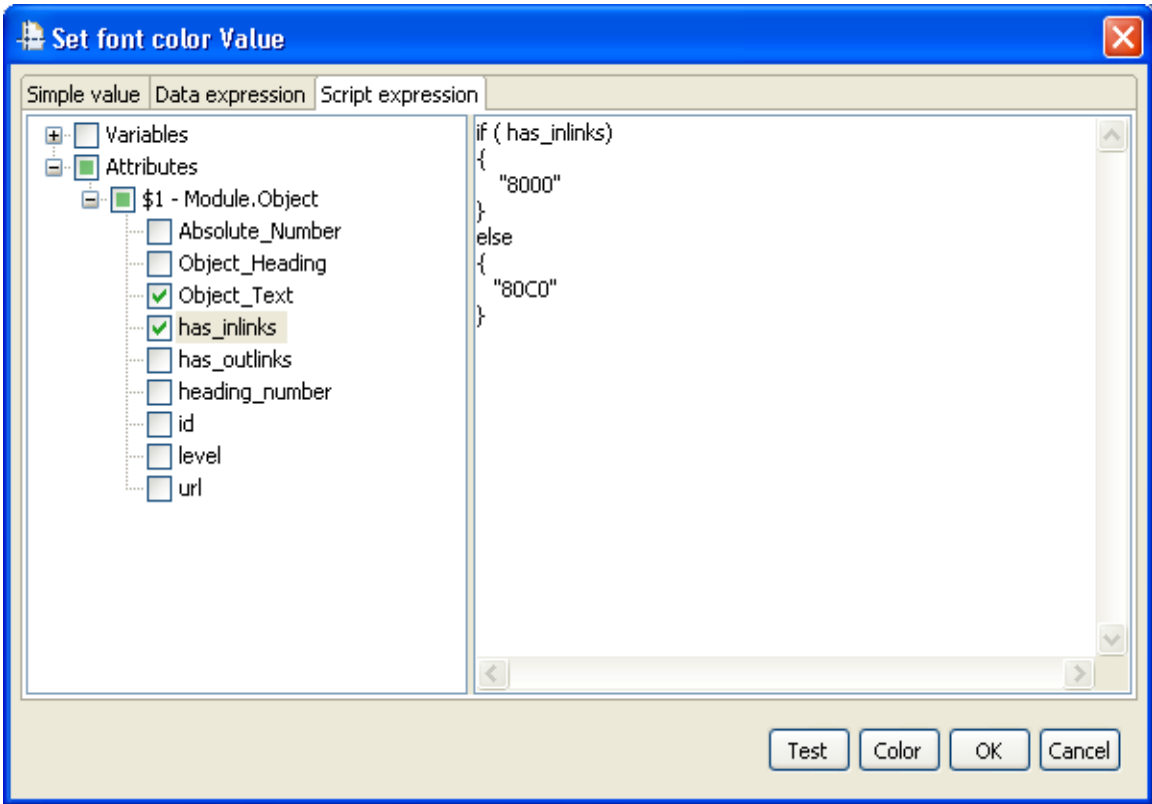
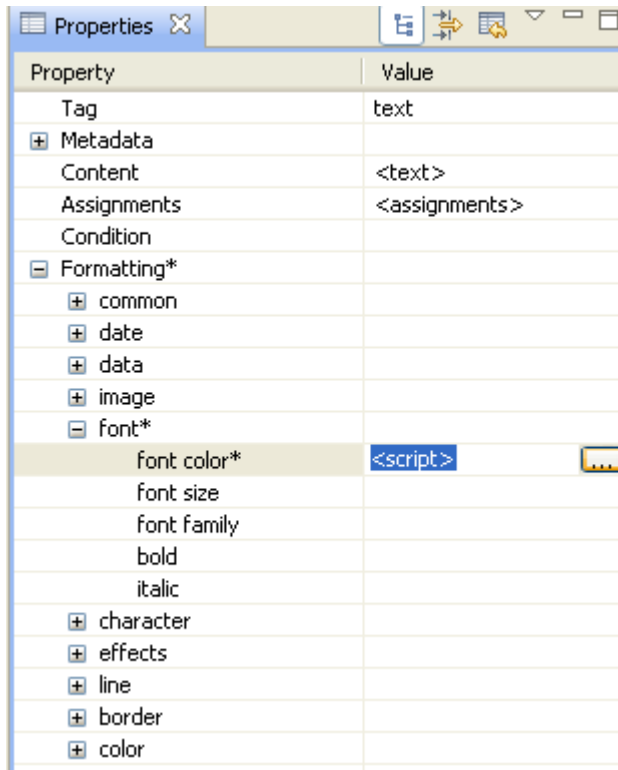


Figure 91 Calculated value properties




Property	Value
Tag	text
Metadata	
Content	<text>
Assignments	<assignments>
Condition	
Formatting*	
common	
date	
data	
image	
font*	
font color*	<script> 
font size	
font family	
bold	
italic	
character	
effects	
line	
border	
color	

Figure 92 Scripted value displayed in the property page

Create New Template Wizard

As an alternative to creating a document template from scratch RPE provides the “Create New Template Wizard”. The 'Create Template' wizard is available in the RPE Document Studio, under the 'File' > 'New' > 'New Template Wizard' menu entry.

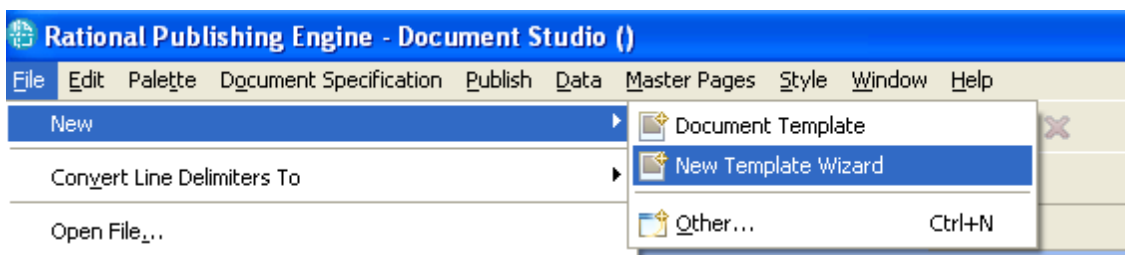


Figure 93 Create Template menu entry

The first page is a dialog welcoming you to the Create Template Wizard. Click 'Next' to continue.

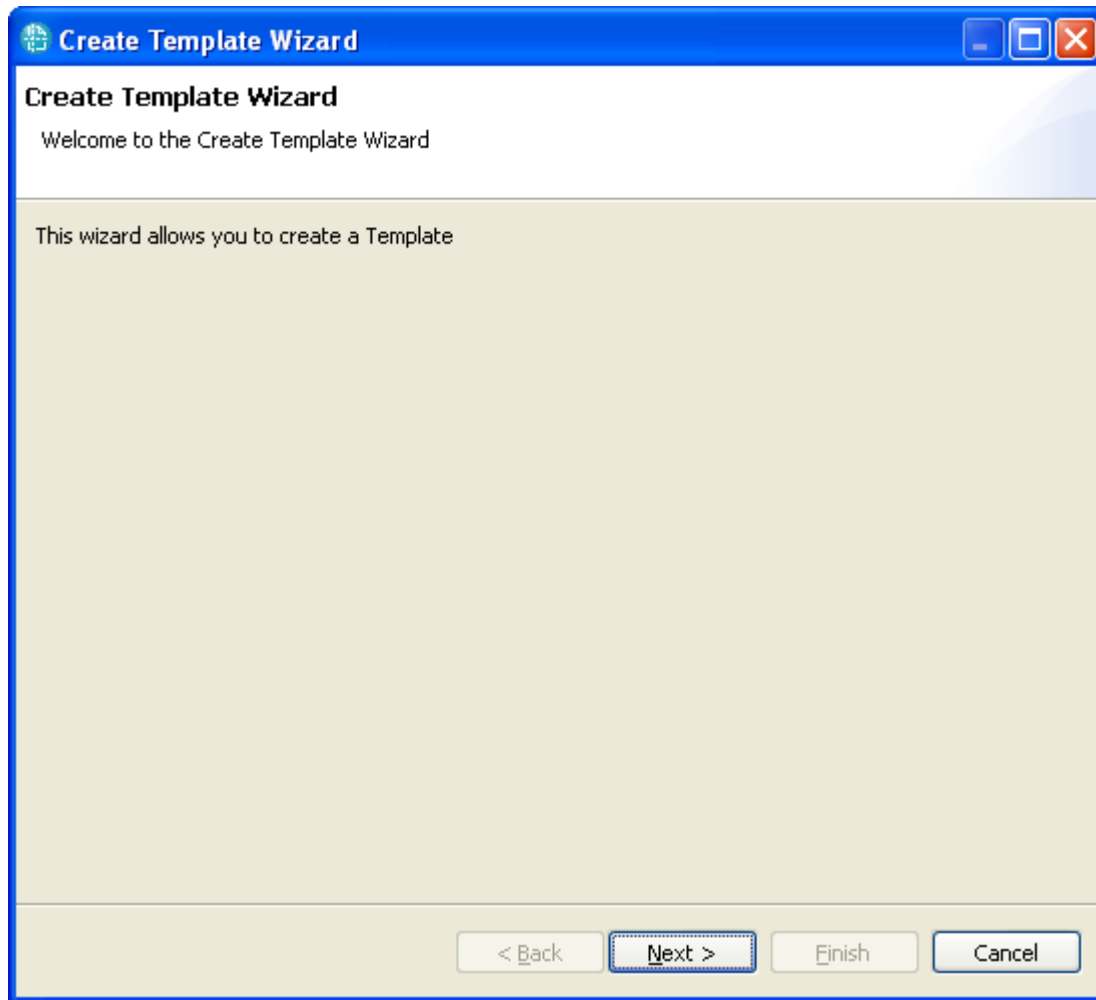


Figure 94 Welcome page

The second page provides options to select the base template. This option can be left empty (no base template used) or an existing Document Template can be selected. The name and descriptions can be edited, and the template elements can be removed from an existing template by checking the 'Clear template content' checkbox.

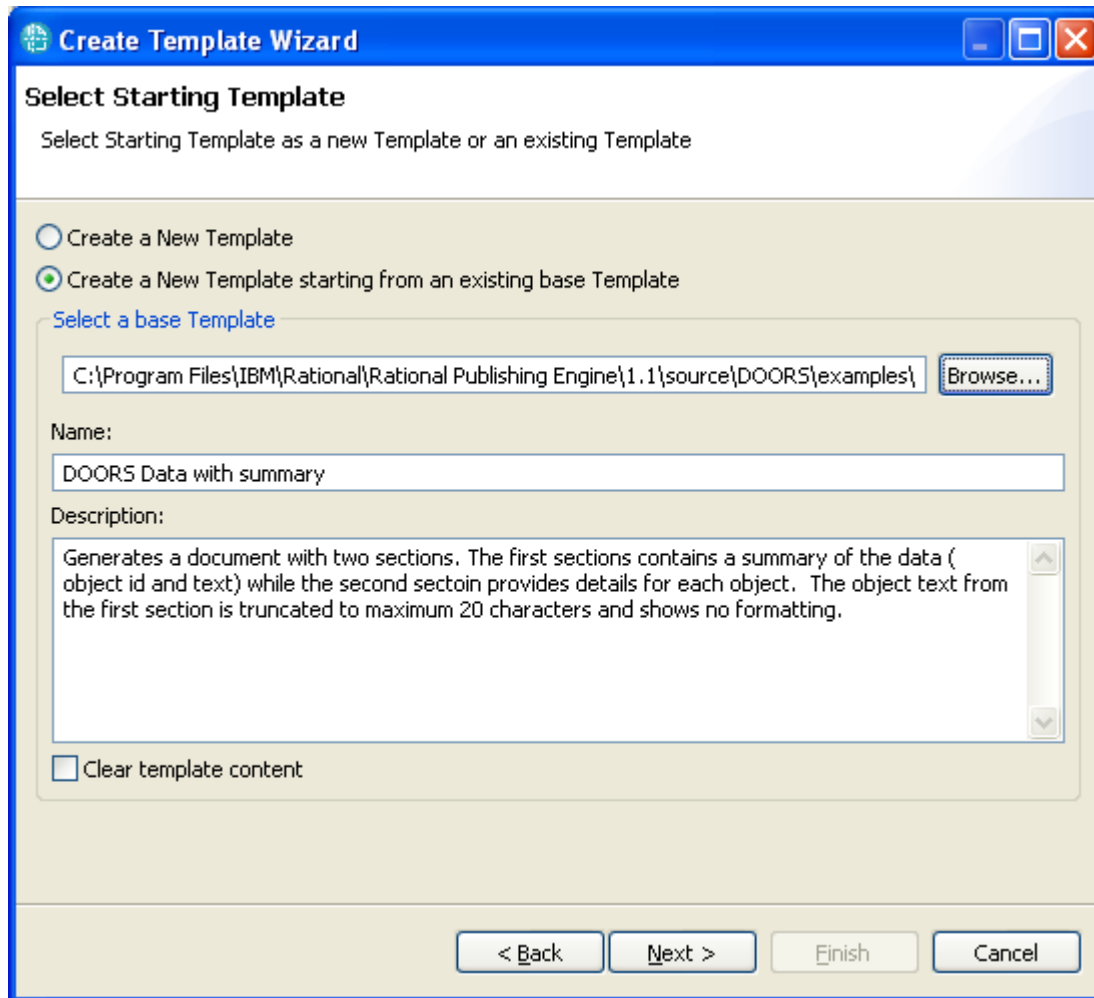


Figure 95 Base Template selection page

The next page provides options for constructing a generic Title page. Paragraph elements containing a Title, Subtitle, along with Table and Section fields (most useful in Word documents) can be chosen to be included in the Template.

Select the options for a Title page as needed and click 'Next'.

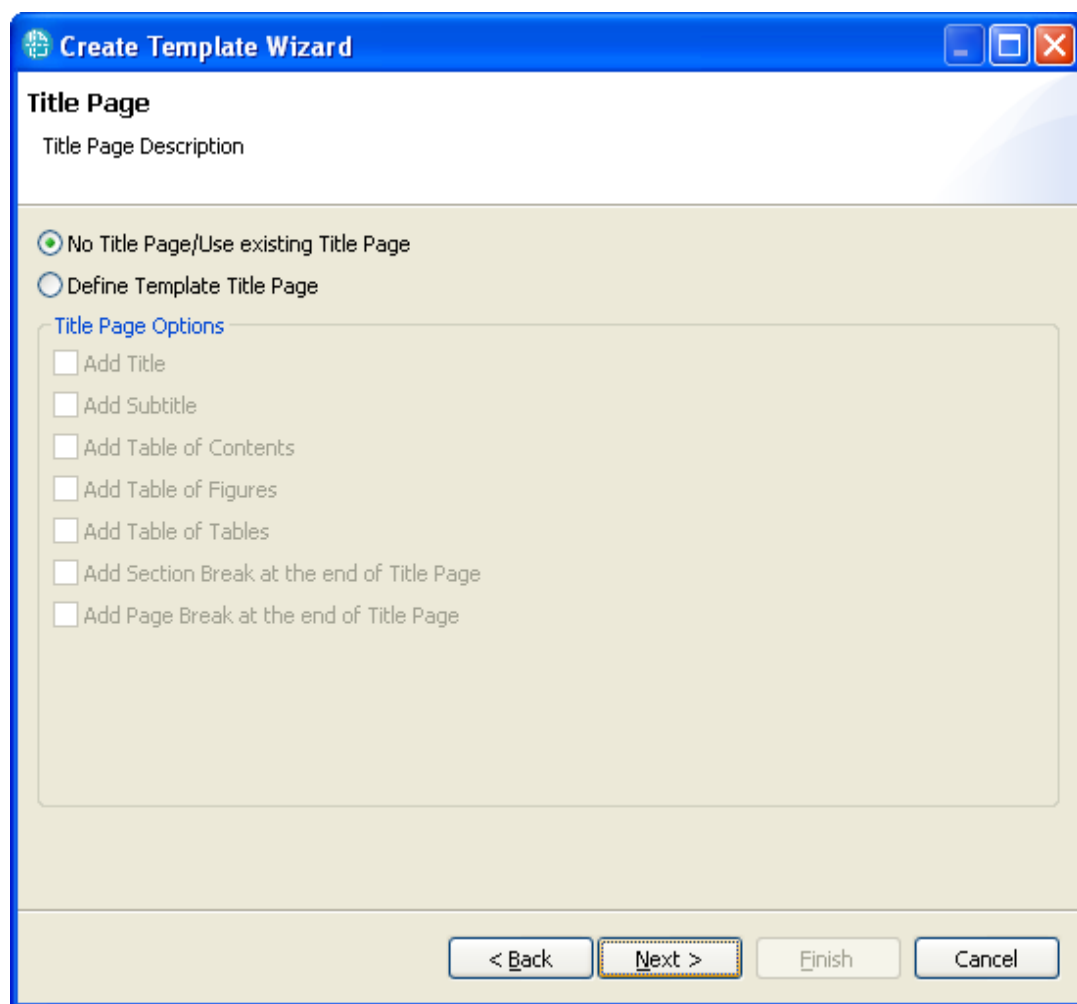


Figure 96 Title selection page

The next page, shown in Figure 97 Data source page97, provides options for editing data sources. This page contains all the data sources existing in the base template. Data sources can be added by clicking on the 'Add... ' button, which starts the 'New Data Source dialog, shown in Figure 98 The 'New Data Source' dialog98. Select a data source in the template and click 'Remove' to remove it from the Template, or click on the 'Edit... ' button to start the 'Edit Data Source' dialog.

NOTE A data source from an existing base Template cannot be edited.

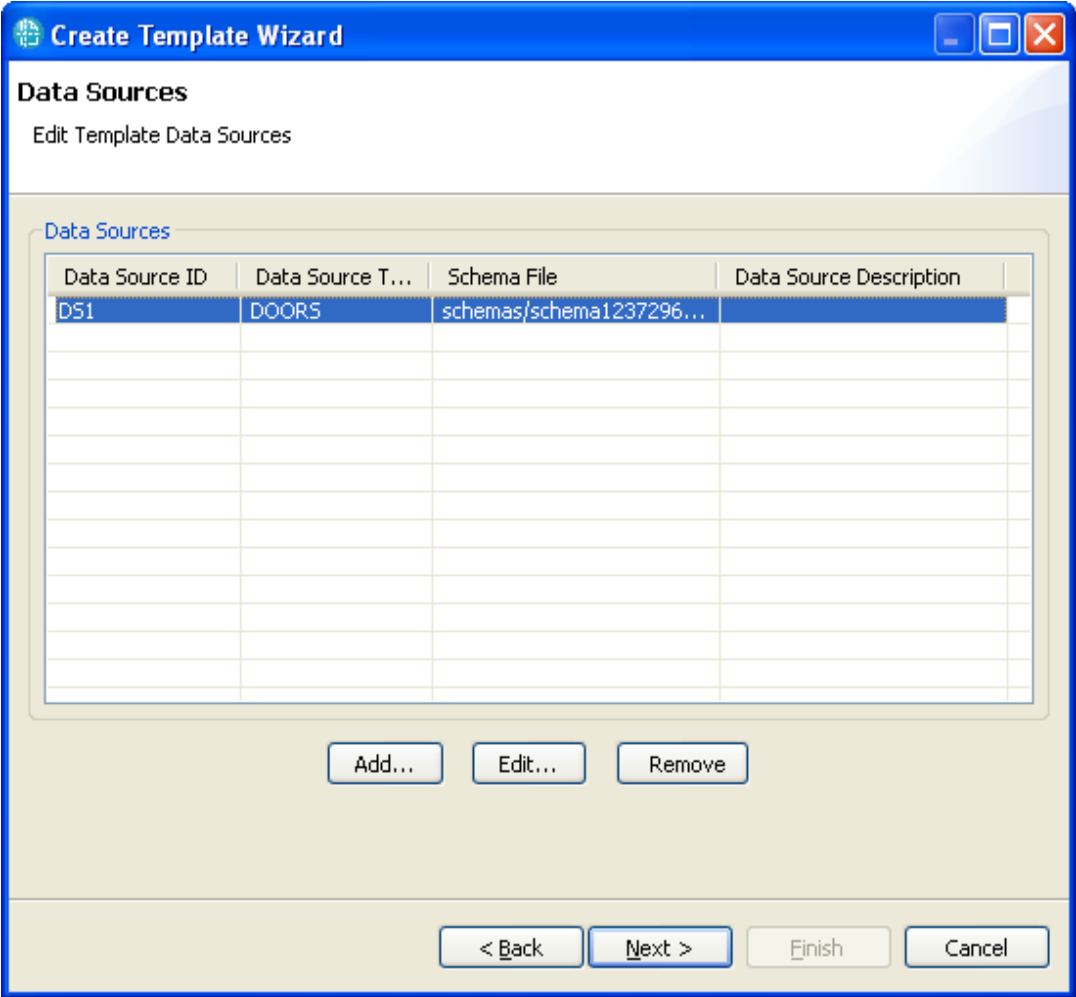


Figure 97 Data source page
Adding a new data source schema.

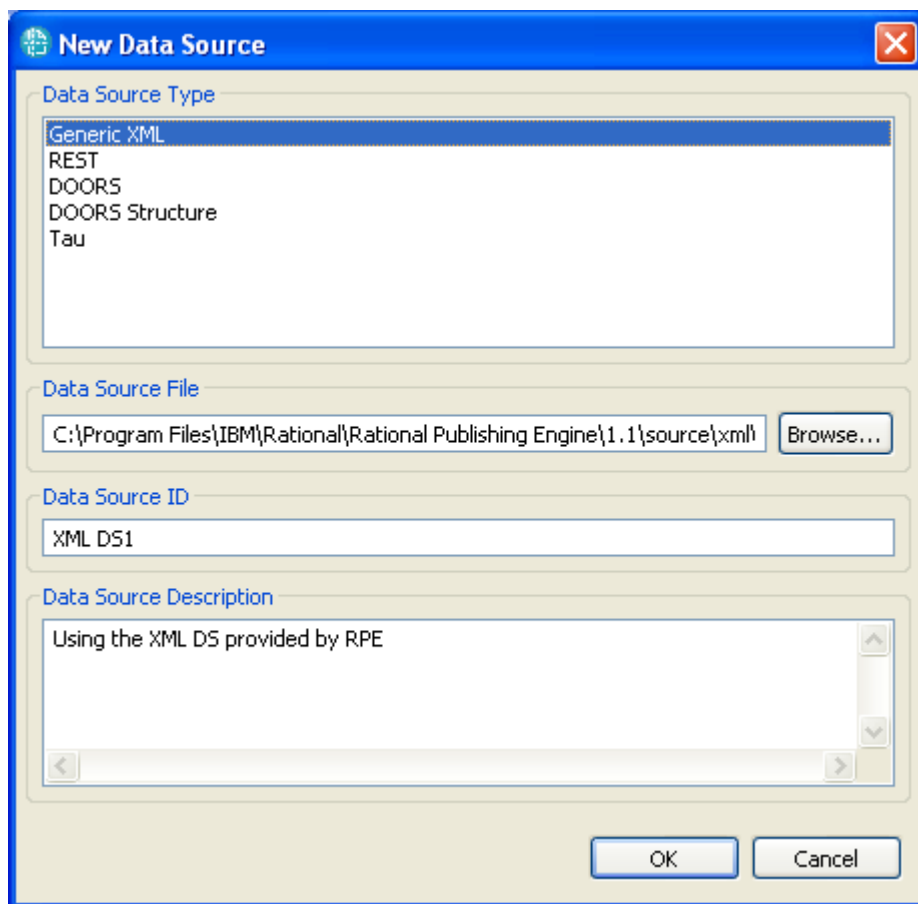


Figure 98 The 'New Data Source' dialog

The next page, shown in Figure 99 Master pages selection dialog99, provides options for editing master pages. By default, the page lists all the master pages existing in the base template. Master pages can be added by clicking on the 'Add...' button, which starts the 'New Master Page' dialog, shown in Figure 100 The 'New Master Page' dialog100. Select a master page in the template and click 'Remove' to remove it from the Template, or click on the 'Edit...' button to start the 'Edit Master Page' dialog.

Note that a master page from an existing base Template can only be edited or removed if the 'Clear template content' checkbox is checked on the base Template selection page (Figure 95 Base Template selection page95).

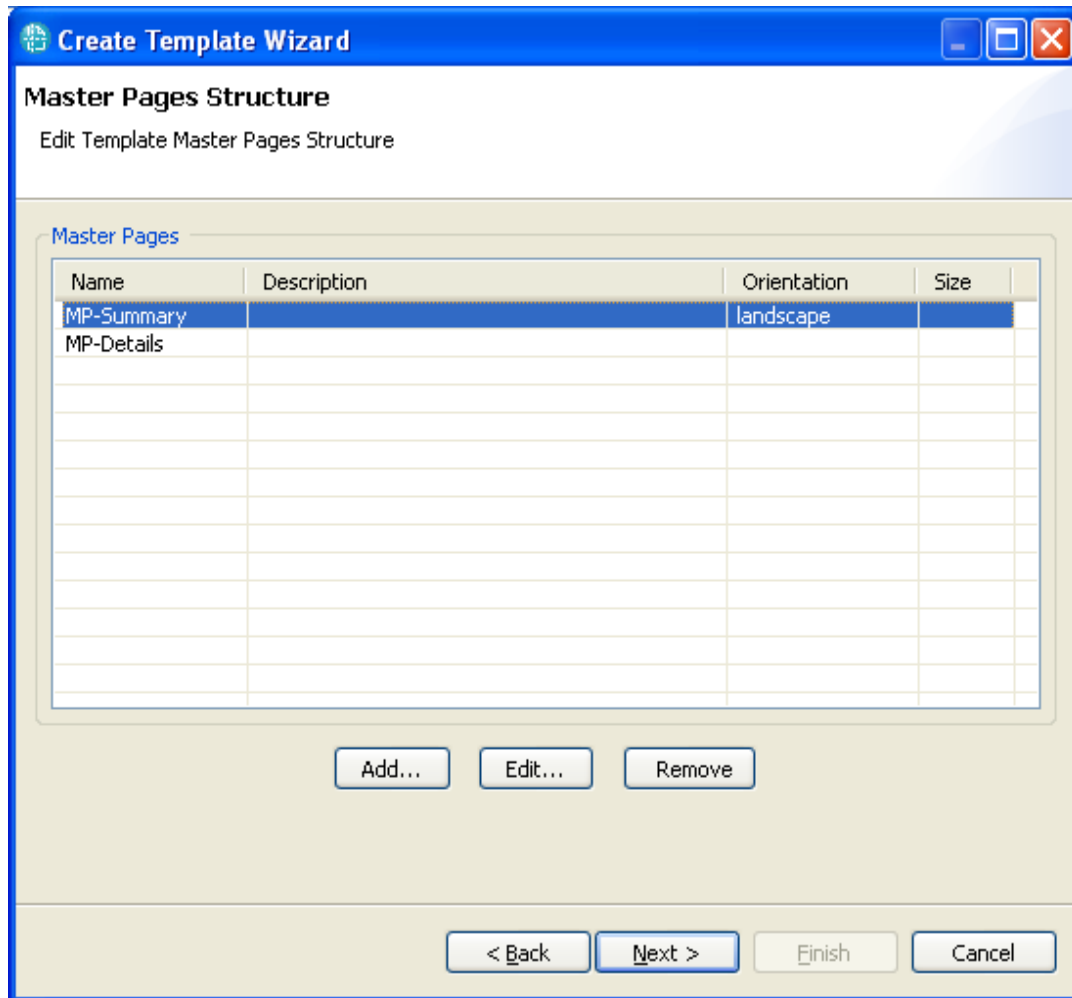


Figure 99 Master pages selection dialog

Creating/editing a master page brings up a new dialog.

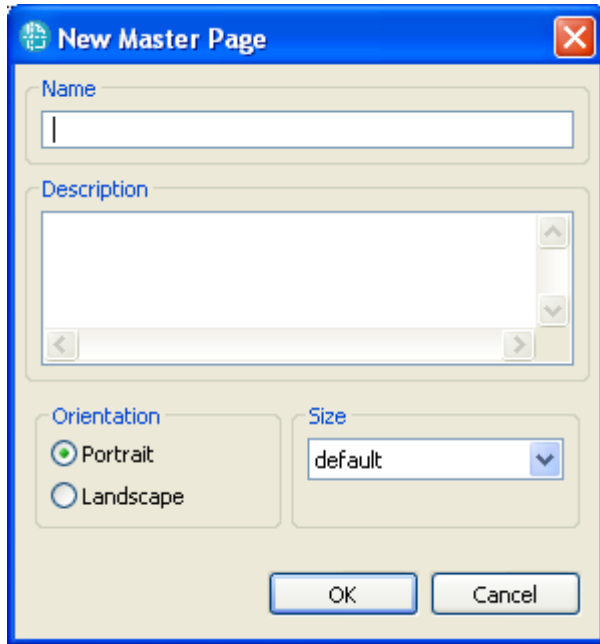


Figure 100 The 'New Master Page' dialog

The next page, shown in Figure 101 Variables page101, provides options for editing variables. By default, the page lists all the variables existing in the base template. Variables can be added by clicking on the 'Add... ' button, which starts the 'New Variable' dialog, shown in Figure 102 The 'New Variable' dialog102. Select a variable in the template and click 'Remove' to remove it from the Template, or click on the 'Edit...' button to start the 'Edit Variable' dialog.

Note that a variable from an existing base Template can only be edited or removed if the 'Clear template content' checkbox is checked on the base Template selection page (Figure 95 Base Template selection page95).

Click 'Next' to proceed to the Summary page.

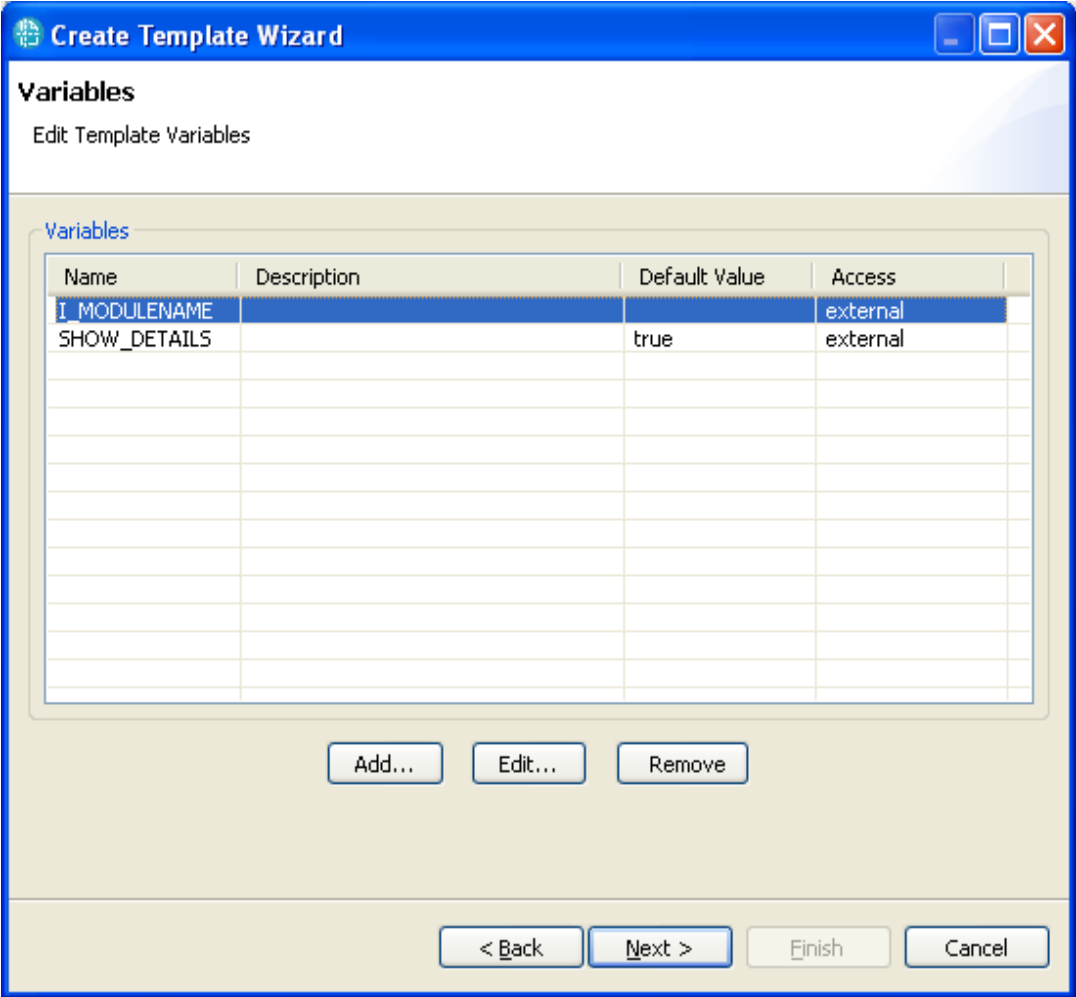


Figure 101 Variables page

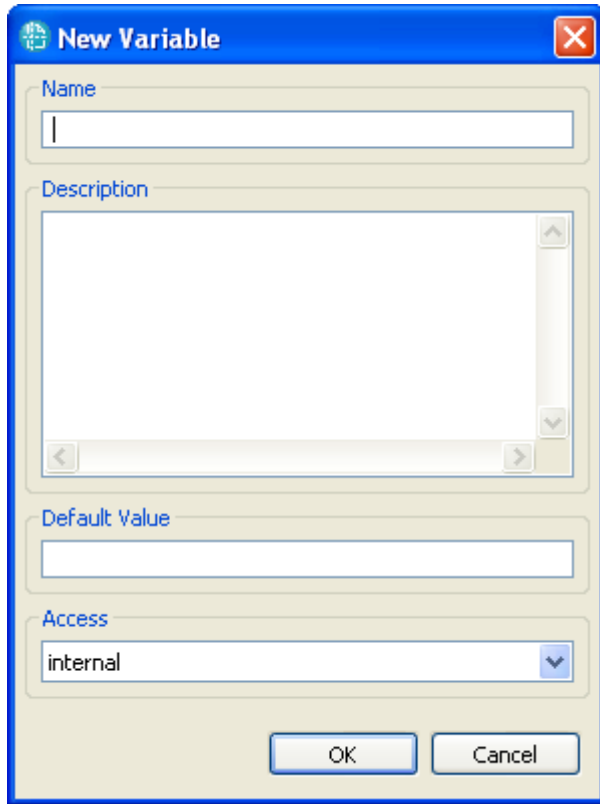


Figure 102 The 'New Variable' dialog

The last page is shows a summary of all the selections. Review this page and click 'Finish' to load the Document Template in the RPE Document Studio.

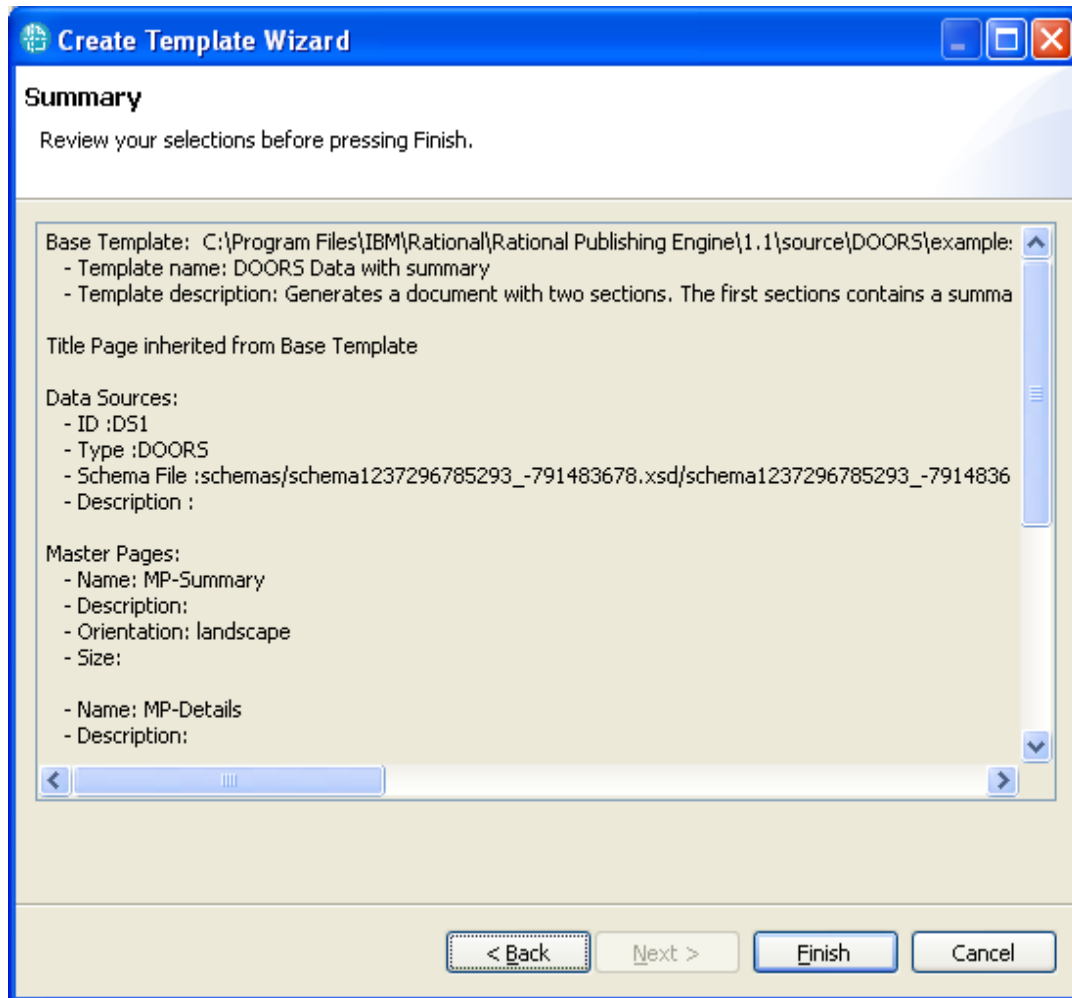


Figure 103 Summary page

Data

As previously stated RPE templates operate with data schemas and not with actual data sources.

Add Data Source Schema Wizard

Adding new schemas can be done in two ways.

Through the “New data source” operation available in the main menu, the context menu in the outline view or from the button in the schema view.

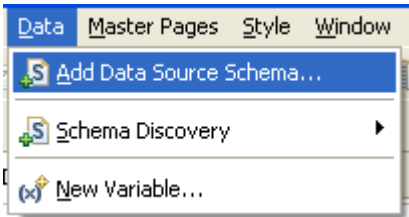


Figure 104 Add data source schema in the main menu

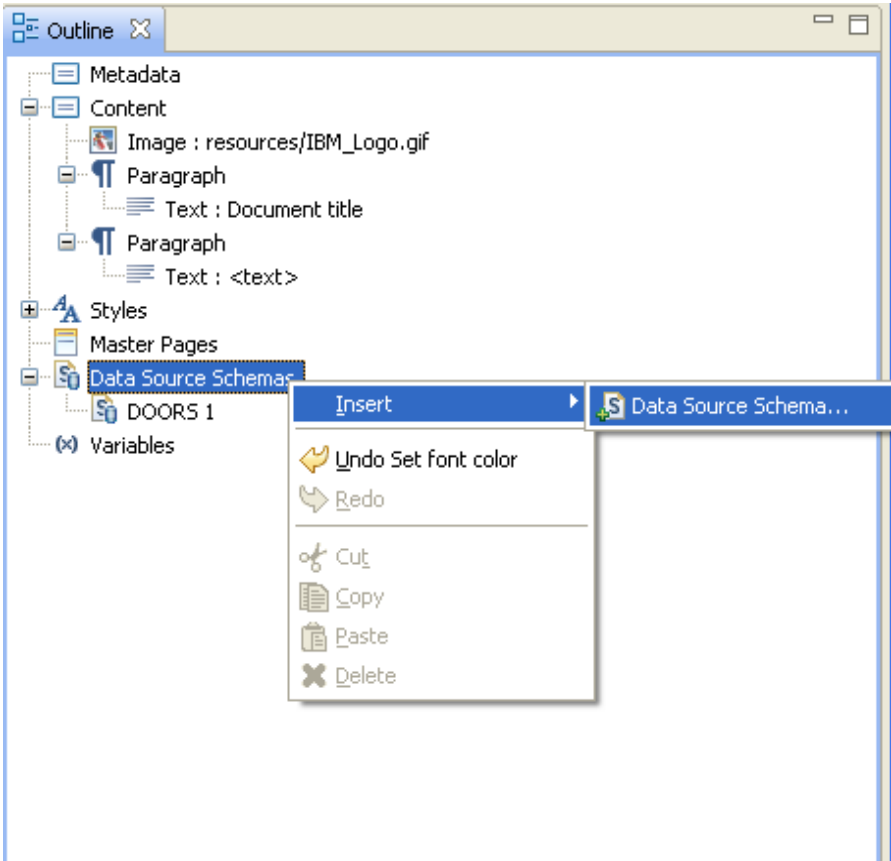


Figure 105 Insert Data Source Schema from Outline context menu

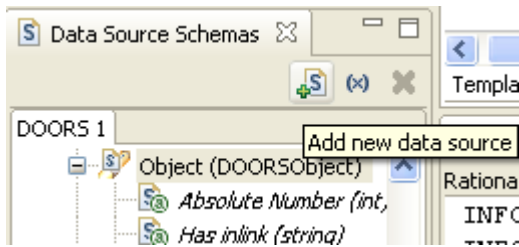


Figure 106 Adding a data source schema from schema view

Using any of the above options will start the “Add Data Source Schema wizard”.

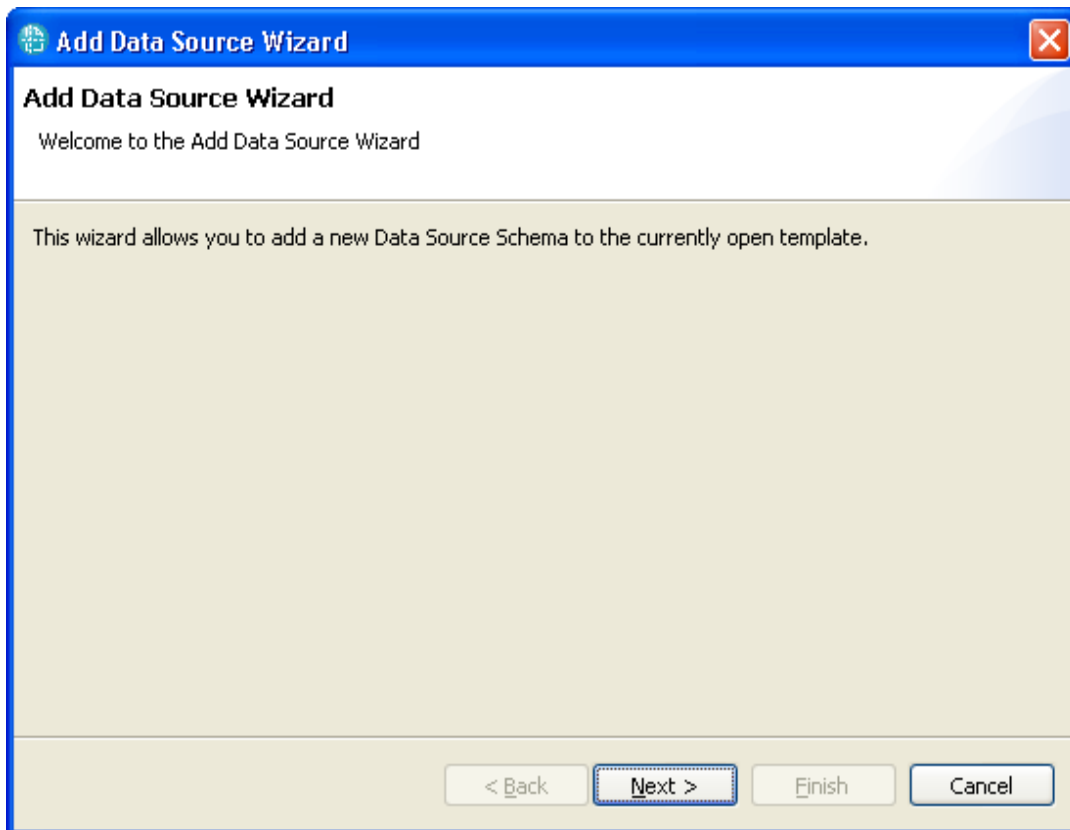


Figure 107 Add data source - Welcome screen

The second page of the wizard allows selecting the type of the schema to add and provide the name for the new data source along with the schema path or URL

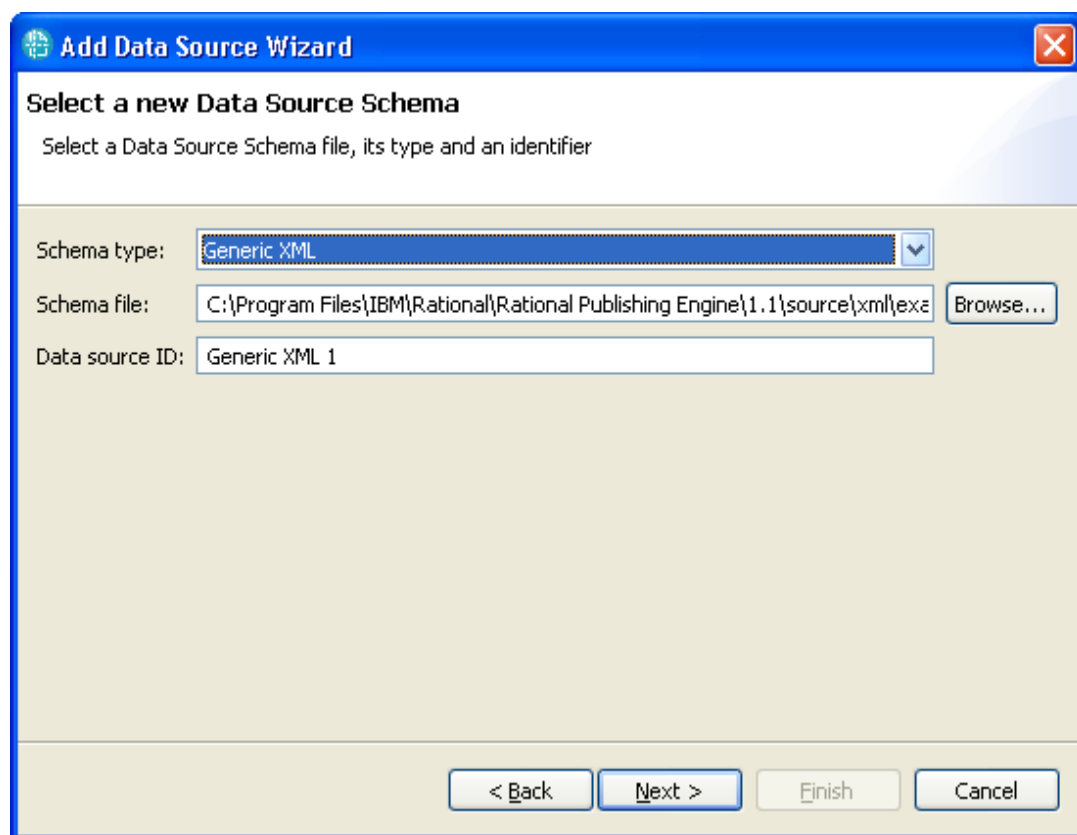


Figure 108 Data Source Schema Type Selection

NOTE You need to provide the full path to the schema. The schema can be on the local file system or on any URL accessible from the machine.

The data source types supported by RPE are listed in the combo box.

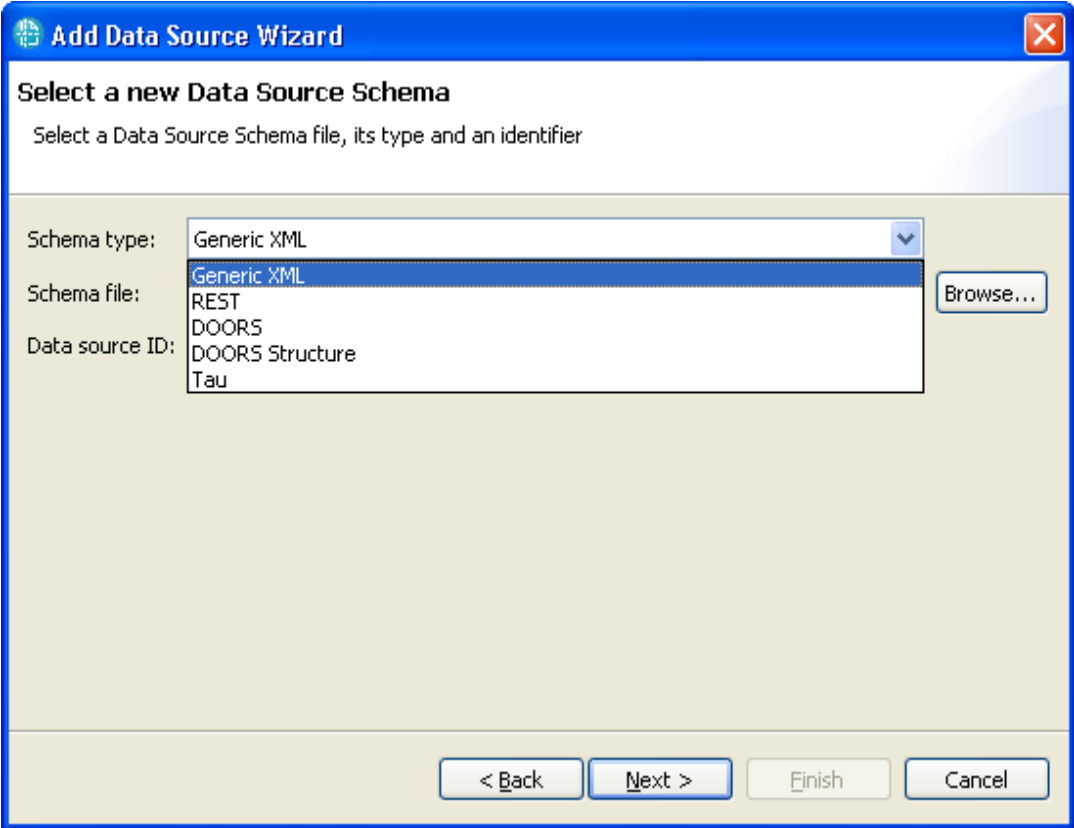


Figure 109 available schema types

A summary dialog displays the selections made.

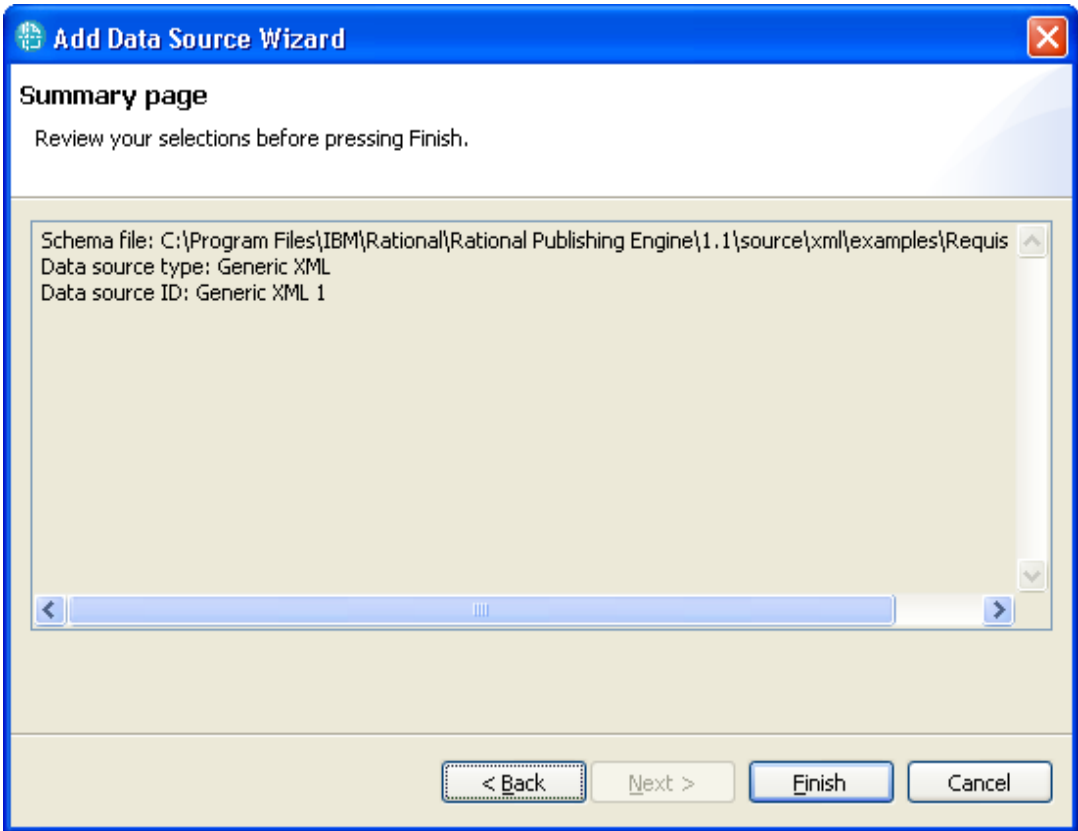


Figure 110 Summary dialog

Schema discovery

For some data sources it is possible to use Schema Discovery wizards to obtain schemas.



Figure 111 Schema discovery menu

Schema view

Once added, the data source schemas are visible in the outline view and in the schema view.

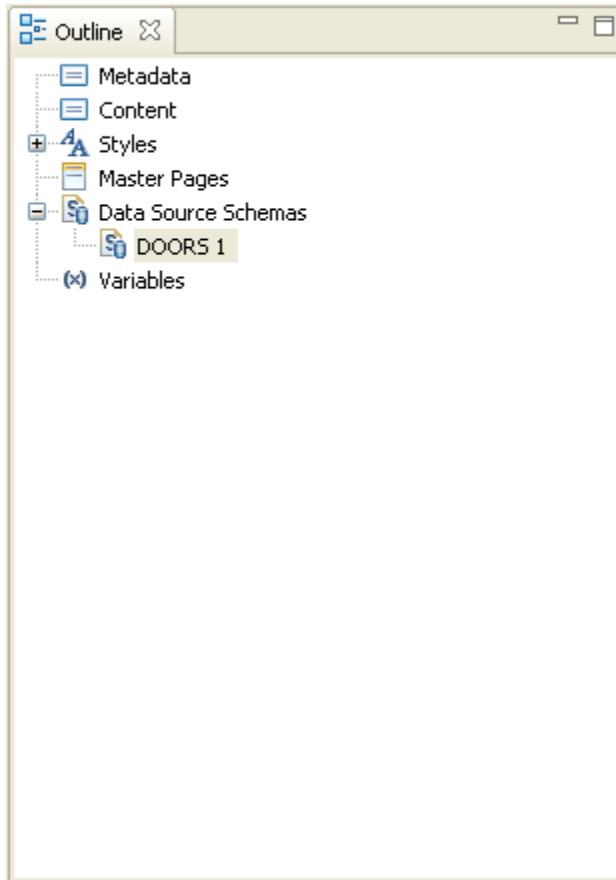


Figure 112 Data source schemas in the outline view

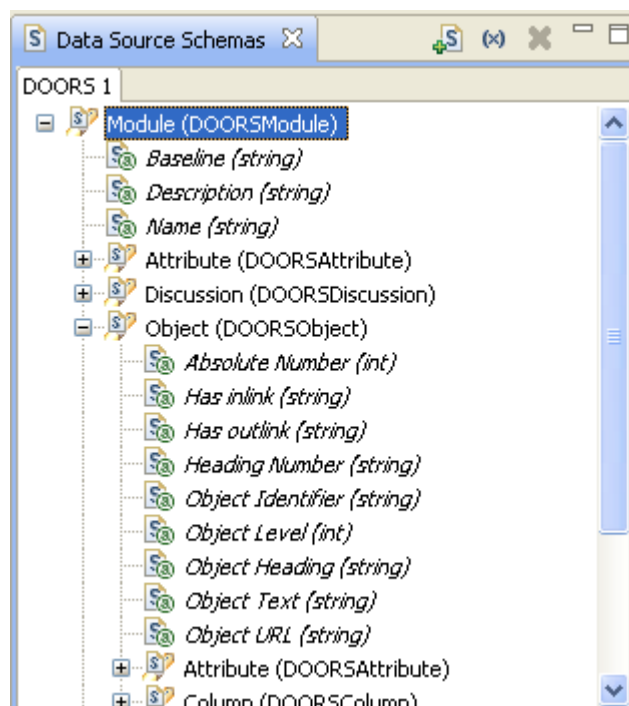


Figure 113 Data source schemas in the schema view

Queries

A query defines what data is to be extracted from the data source. You can assign a query to any template element. Template elements having queries will be generated for each data element extracted from the data source.

You assign a query to a template element by dragging an element from the schema view onto the template element.

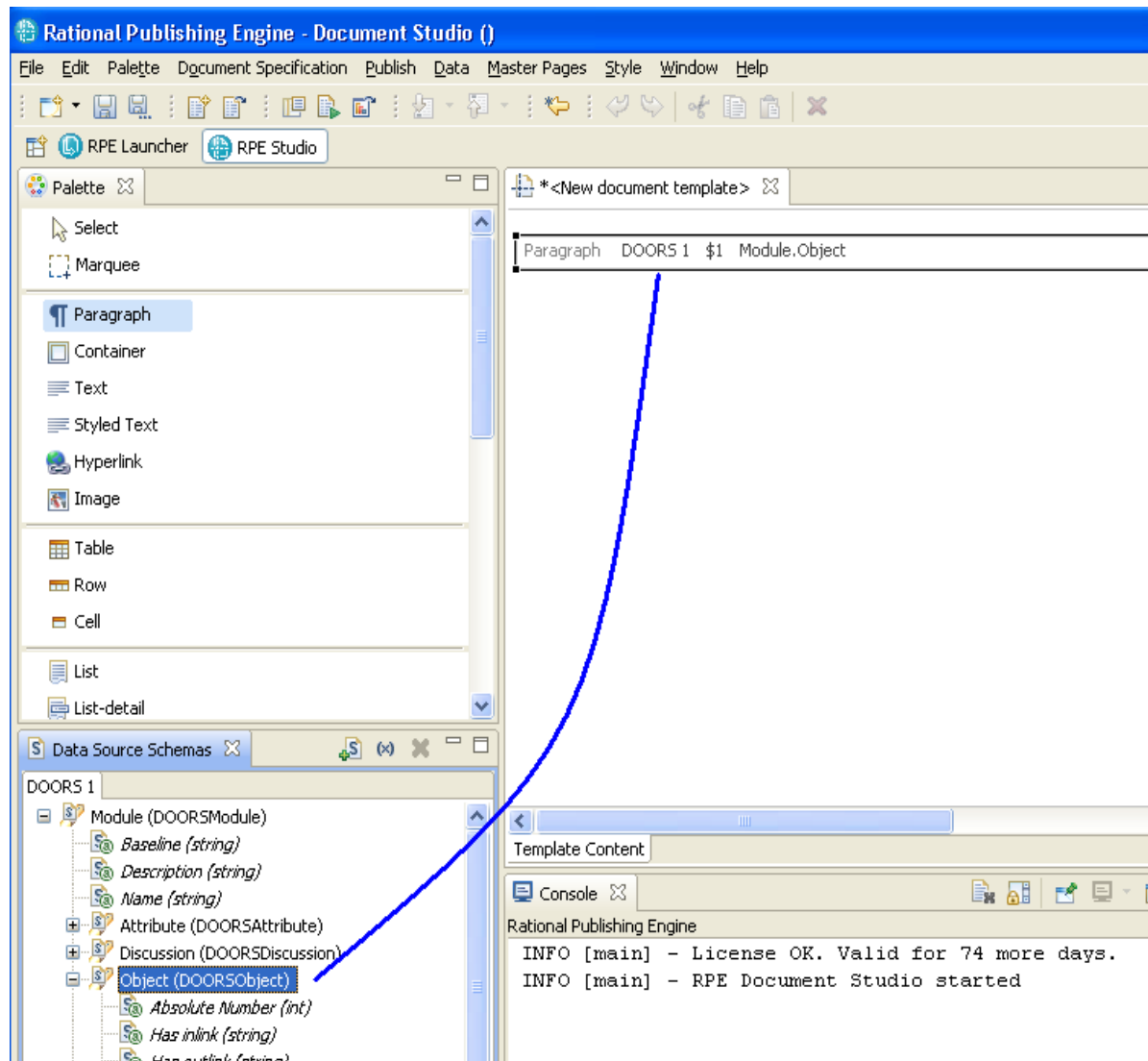


Figure 114 Assigning a query to a template element

Once a query is assigned to a template element, you can use the attributes of the query type (the element dragged from the schema view) anywhere in the template element, including its children.

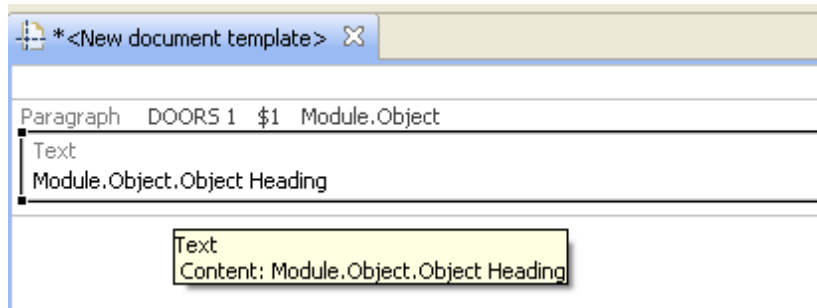


Figure 115 Using data attributes

Using data attributes in template elements

You can specify which data attributes to display by dragging them from the schema view or through the content editor.

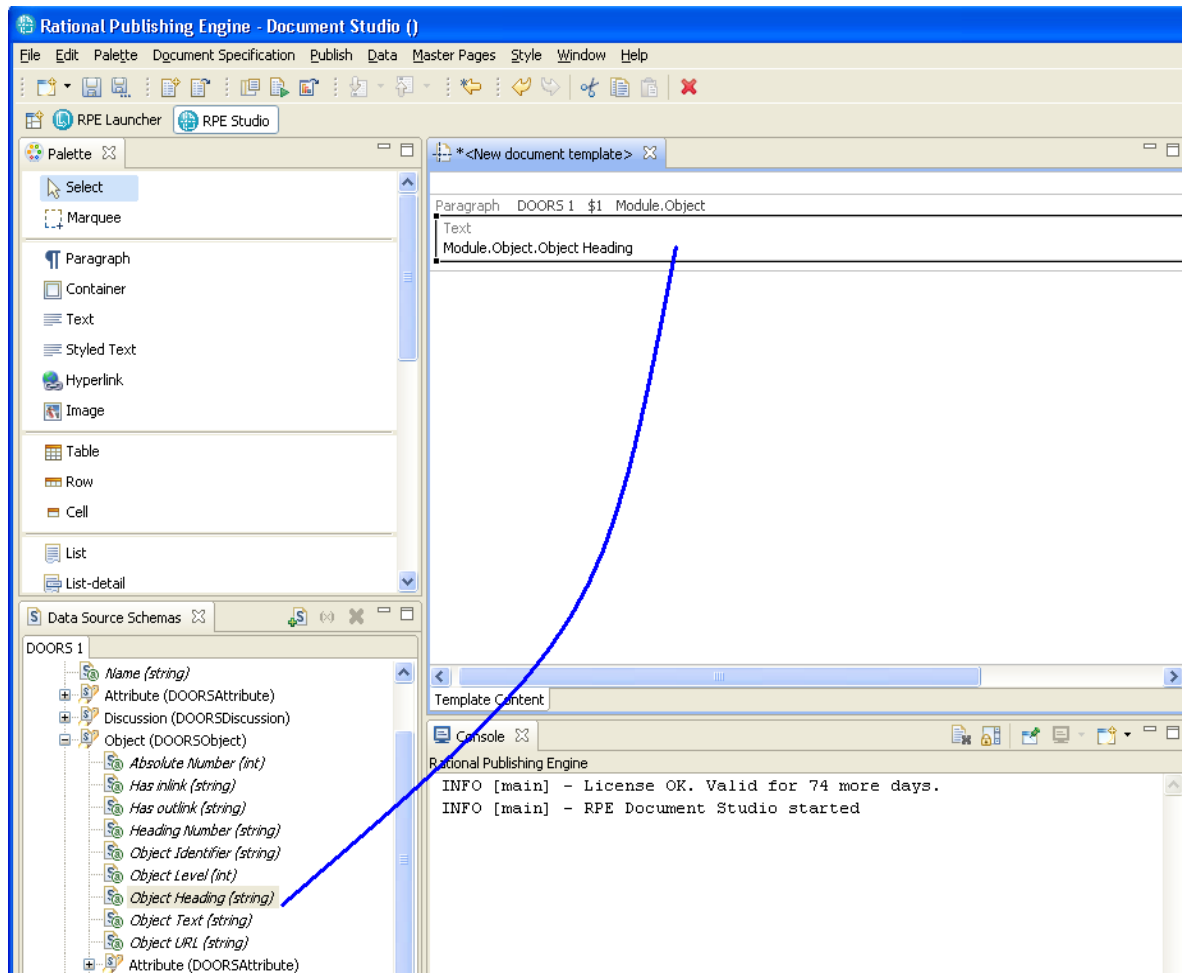


Figure 116 Dragging a data attribute

Edit the text element's content using the expression editor by double clicking the text element.

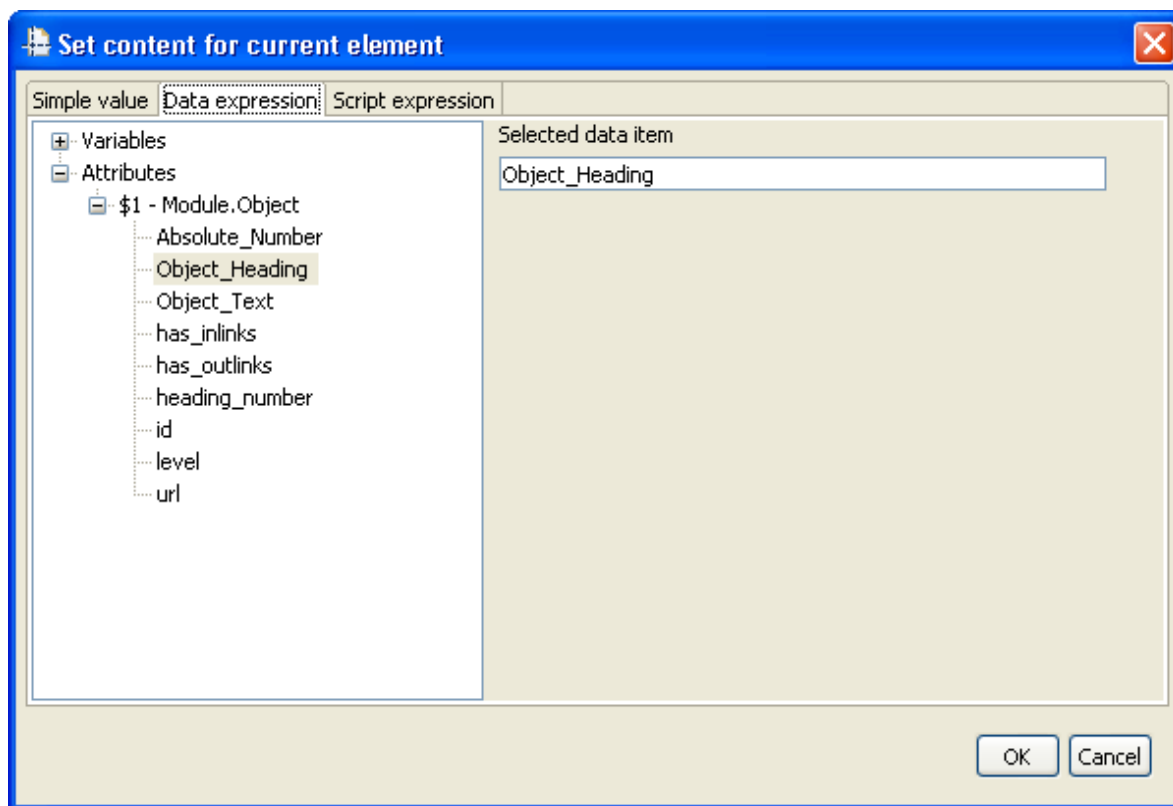


Figure 117 Expression Editor for the content of a text element

NOTE Only the **text** and **hyperlink** elements can be used to render data attributes. All the other template elements can use data attributes for conditional formatting only.

Content editor

To help define the content of a template element or values for formatting properties, RPE provides the Expression Editor. The editor has 3 tabs:

- Simple value
- Data expression
- Script expression

NOTE The content of the last used tab will be used for the value currently edited. The simple value tab allows entering static text.

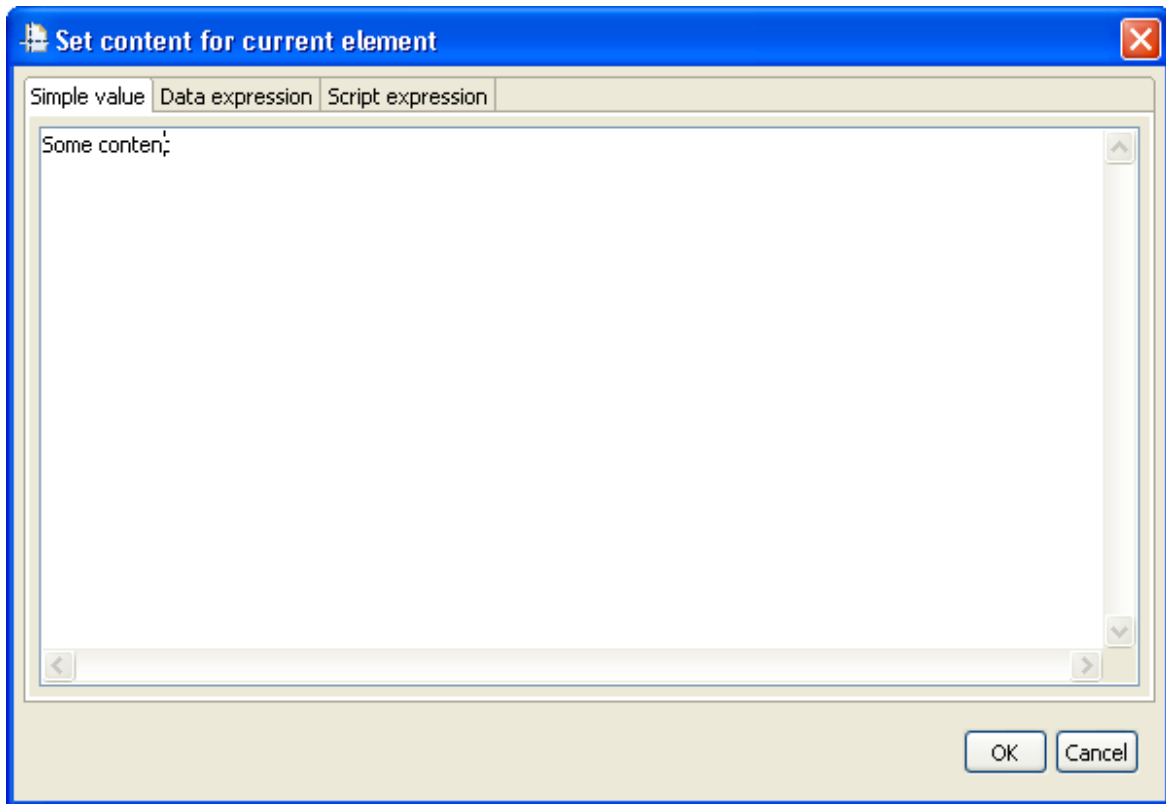


Figure 118 Content editor - simple value

The data expression tab allows selecting a data attributes and variables available in the current context.

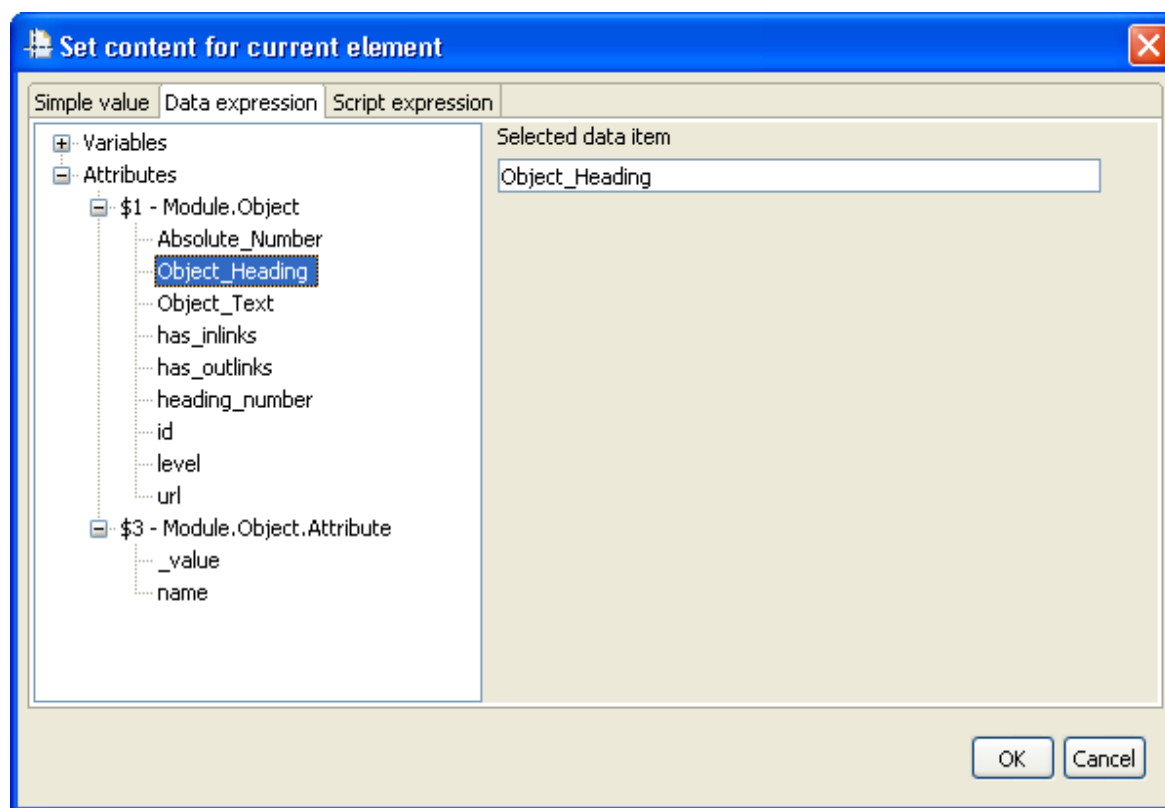


Figure 119 Content editor - data expression

The script expression tab allows entering Java Script code that can use the data attributes and variables available in the current context.

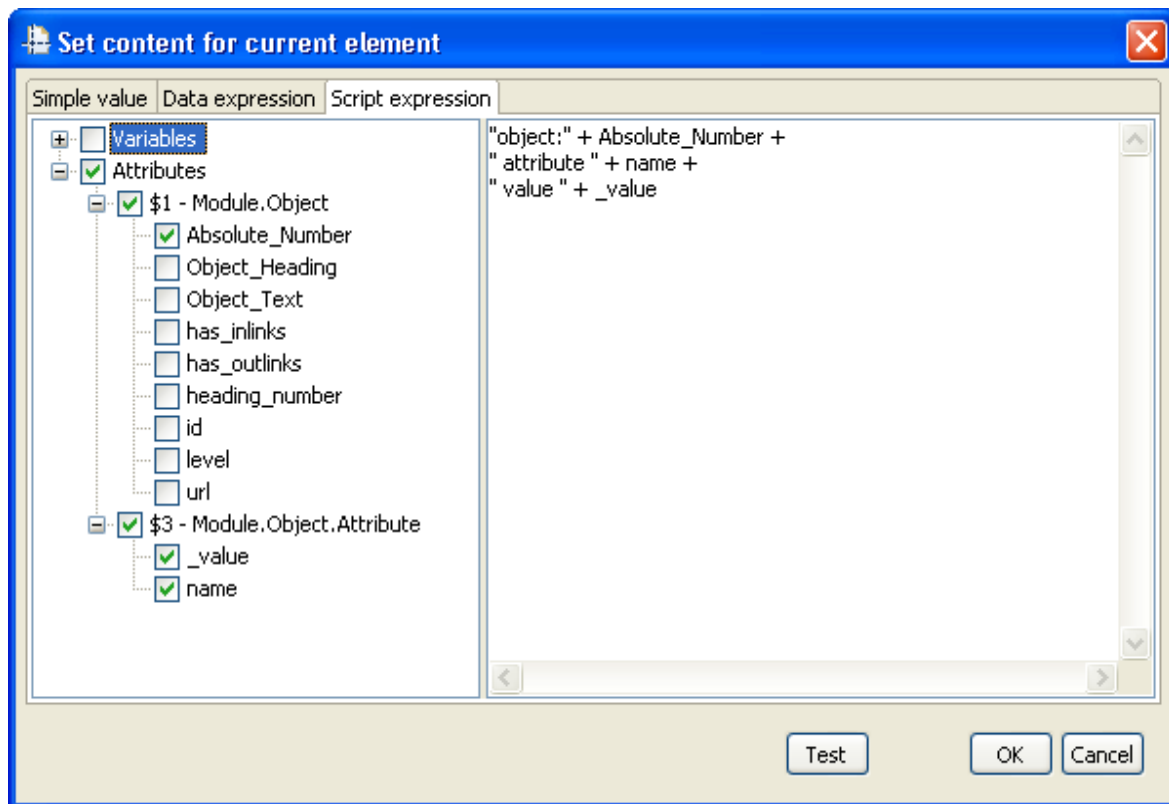


Figure 120 Content editor - script expression

Condition editor

For filters and conditions RPE provides a condition editor. The editor contains two tabs:

- native filter – available when defining filters
- scripted expression – available when defining filters and conditions

The native filter allows the user to enter filters in the format supported natively by the data source. RPE does not process that filter in any way. The user must consult the data source documentation for details on the appropriate syntax.

The scripted expression editor assists the user in defining a Java Script expression using the data attributes and variables accessible in the current context.

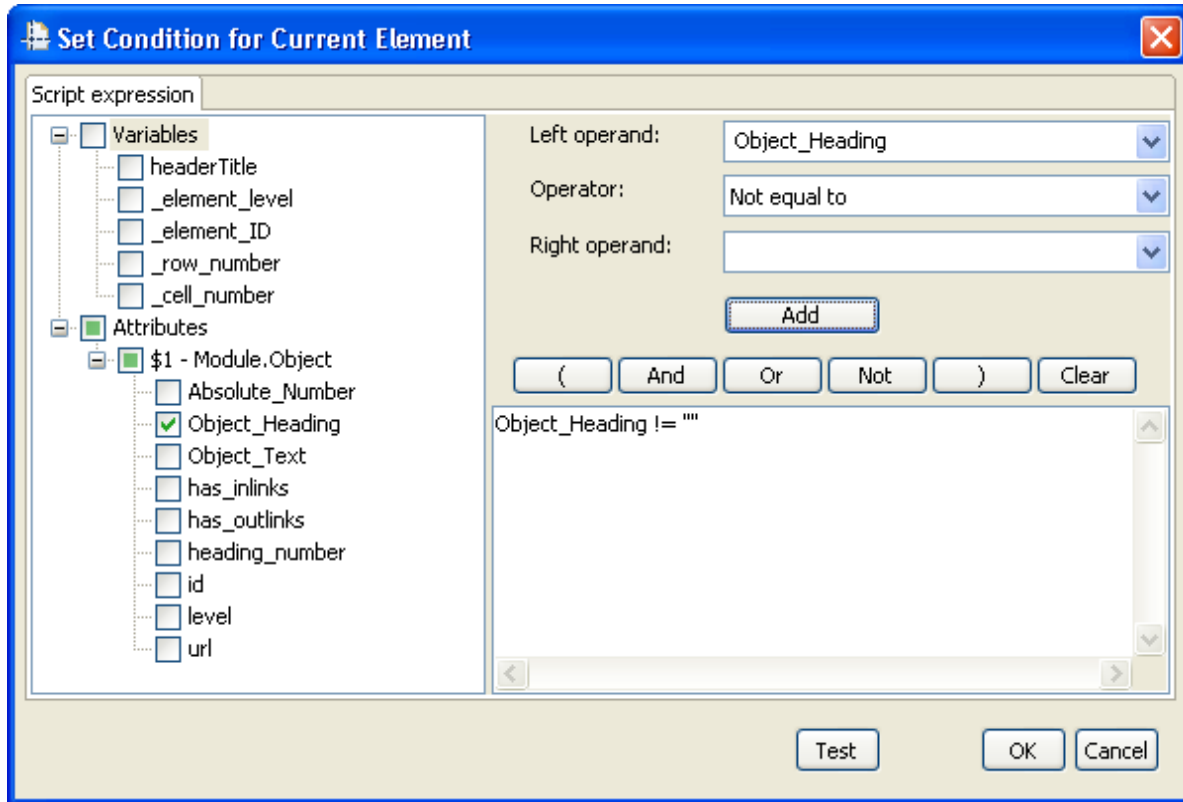


Figure 121 Condition editor - scripted expression

You can use the tools on the page to add and link expression parts but you can also manually type the condition. If the expression is built using the tools on the page then any data attribute or variable used is automatically selected in the left pane. If you enter the condition manually you must also manually select the used components.

Filter

A query can have a filter defined for it. A filter can be expressed using the native data source means (if available and textual) or via a RPE Filter. A native filter is interpreted by the data source. A RPE filter is processed by RPE as it extracts data from the data source.

See Data section for details on using filters.

TIP Whenever possible it is recommended to use native filter as it should yield better document generation times than when a RPE filter is used.

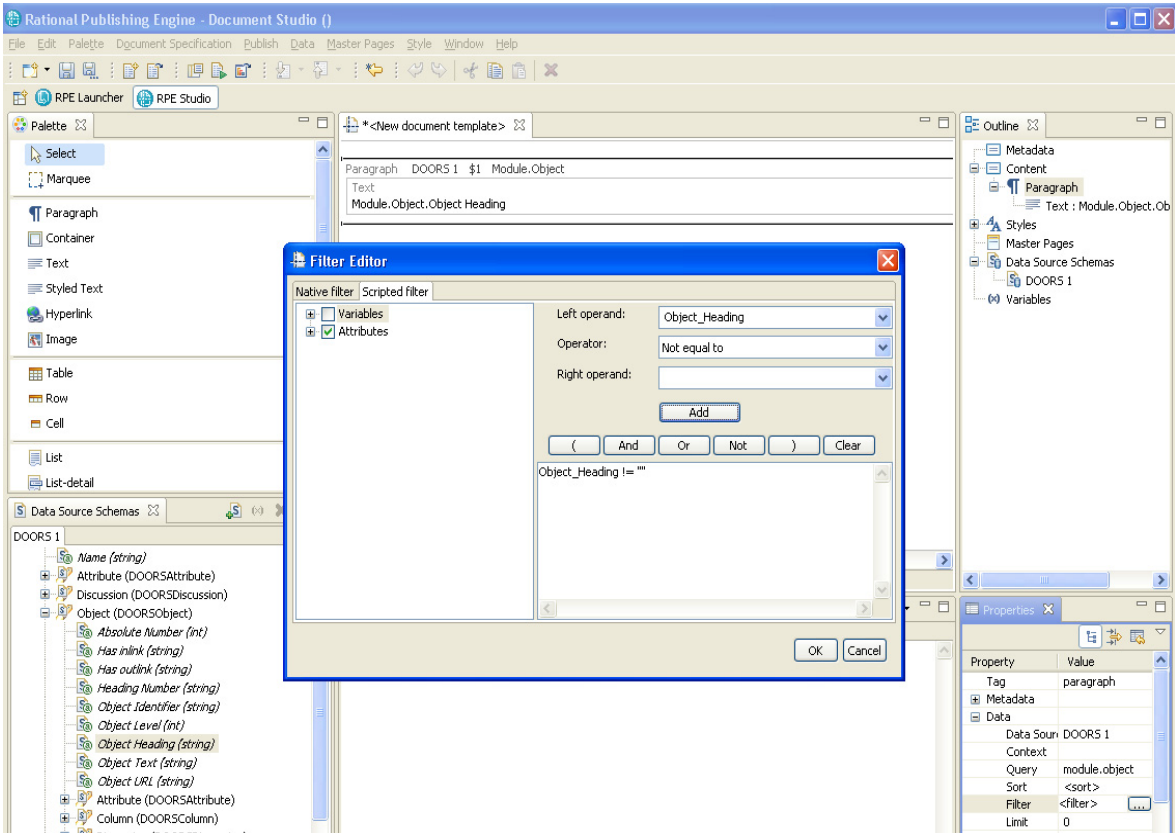


Figure 122 RPE Scripted Filter

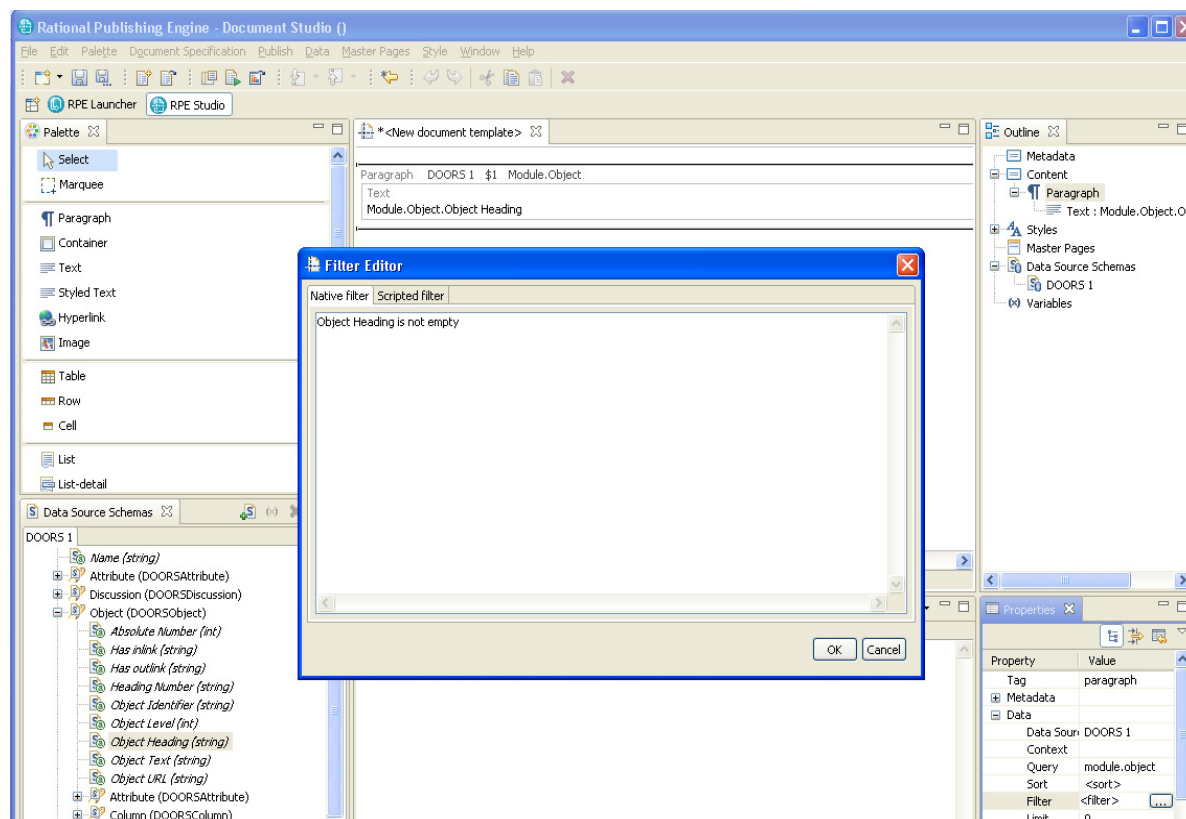


Figure 123 Native filter (DOORS)

NOTE RPE cannot check the native filter correctness. It is the user's responsibility to ensure the native filter is correct. For DOORS Data Sources the native filter should be first tested in DOORS then copied from the "Description" window of the Filter properties.

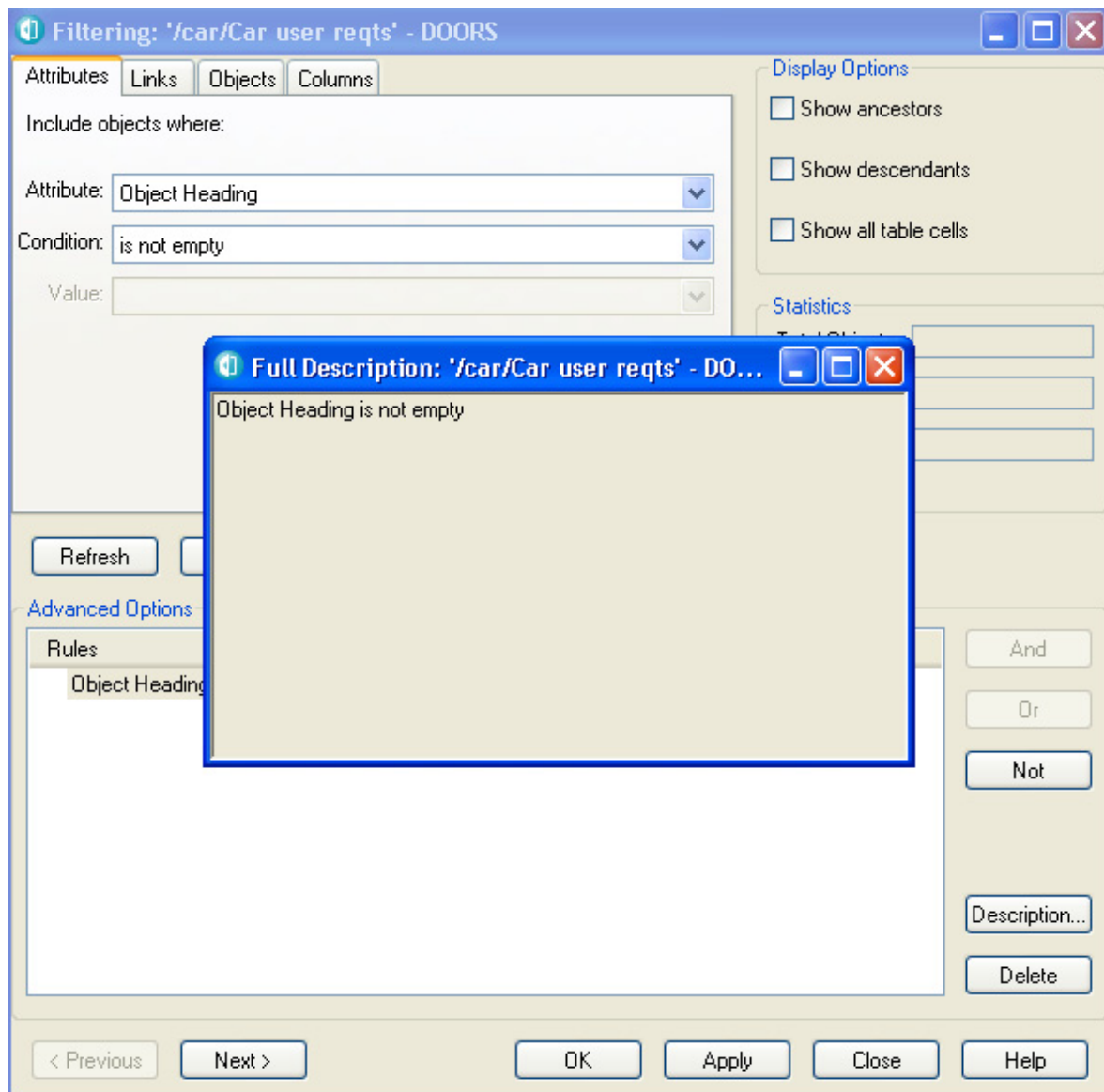


Figure 124 checking the native filter in the data source environment

Sort

A query can have a sort defined for it. A sort can be expressed using the data source's native sort (if available and textual) or as RPE Sort. A native sort is interpreted by the data source. A RPE sort is processed by RPE as it extracts data from the data source.

See Data section for details on using sort.

TIP Whenever possible it is recommended to use a native sort as it should yield better document generation times than when a RPE sort is used.

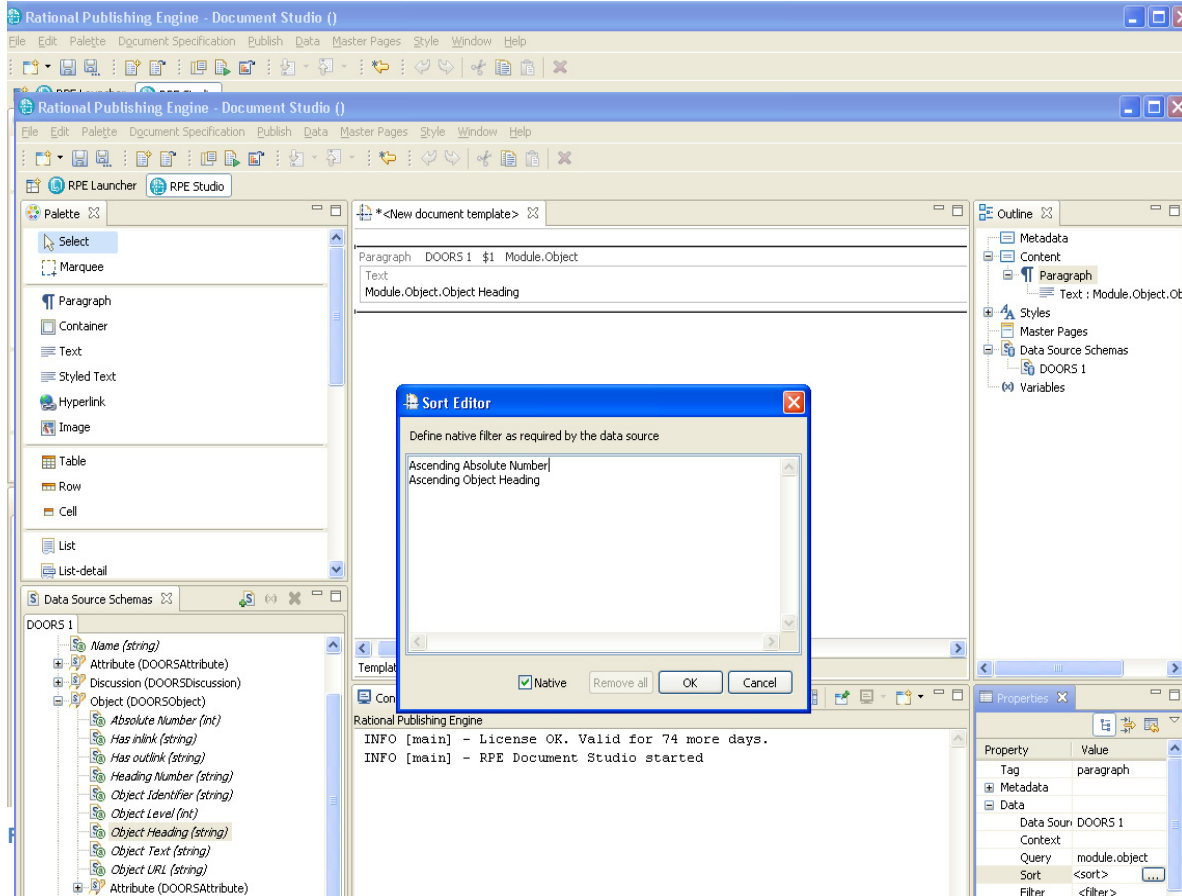


Figure 126 Native Sort (DOORS)

NOTE RPE cannot check the correctness of a native sort. It is the user's responsibility to ensure the native sort is correct. For DOORS Data Sources the native sort should be first tested in DOORS.

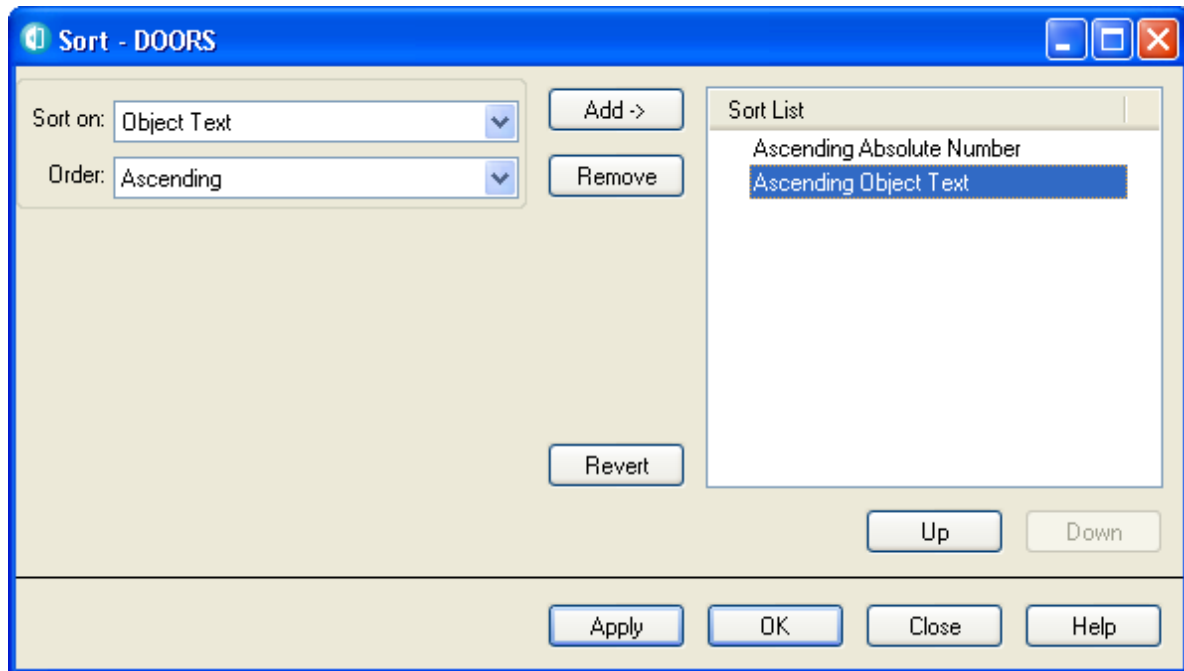


Figure 127 Native sort in data source (DOORS)

Data Contexts

You can use only the attributes valid in the current context (see the Data/Queries and contexts for details). RPE studio will prevent you from dragging attributes not visible in the current context.

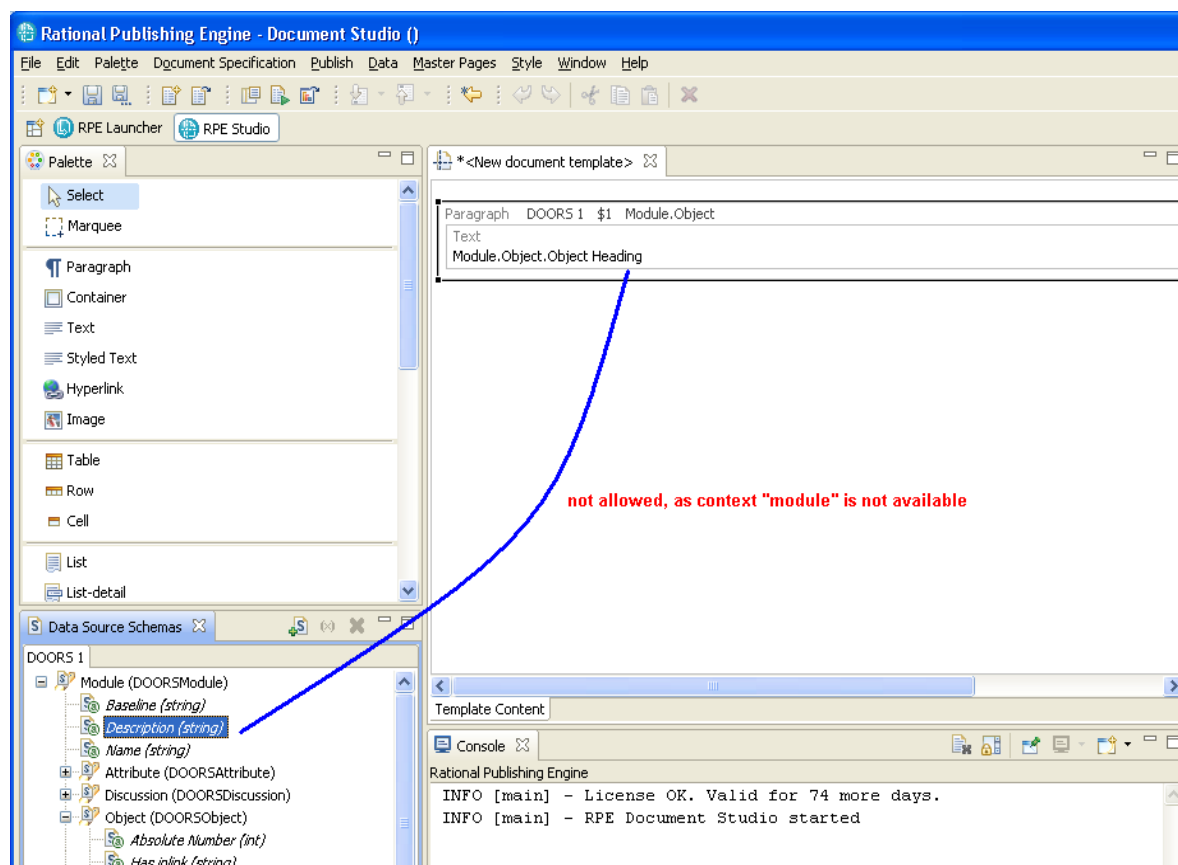


Figure 128 Disallowed attribute use

In the above example the “Description” attribute is not part of the “Object” type in the schema and “module” is not an available context. RPE will not allow dragging it in the selected text element from the editor area.

Nested queries

You can nest queries by assigning queries to children template elements.

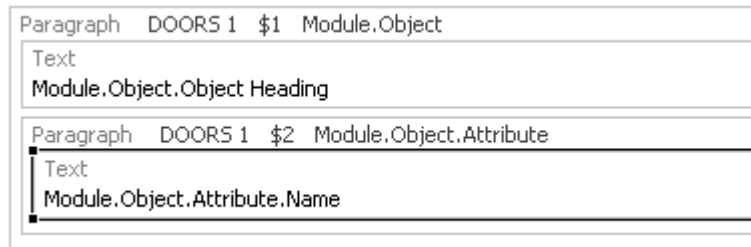


Figure 129 nested queries

When a query is dragged onto an element that has at least 1 parent element with a query, RPE will calculate if the dragged query can be executed in the context parent's element query. All the queries that can serve as context are displayed for selection. The list displays the id of the query as well as its textual representation.

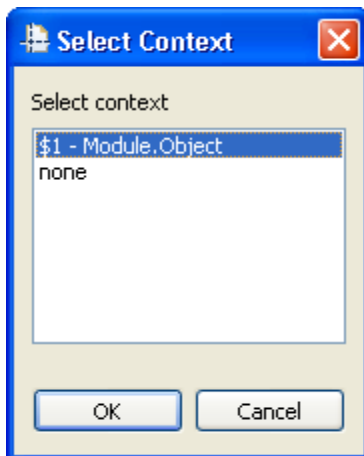


Figure 130 Context selection

In the example above module.object can serve as context for module.object.attribute hence the identified of the query is displayed in the list.

Selecting **none** for the context will result in two nested but unrelated queries.

In the above example, setting the second query's context to \$1 will produce the following output:

- a set of paragraphs containing the heading of each object in the module
 - a list of paragraphs with the attribute names for the *current object* in query \$1

In the above example, setting the second query's context to **none** will produce the following output:

- a set of paragraphs containing the heading of each object in the module
 - a list of paragraphs with the attribute names for *all the objects*

Advanced usage of data attributes

The data attributes can be used for more than displaying the raw data extracted from the data source.

Calculated values

Using the expression editor you can create JavaScript snippets to process the data attributes as needed. Examples of such processing is combining data attributes, trimming white spaces, transforming numeric values into textual descriptions etc.

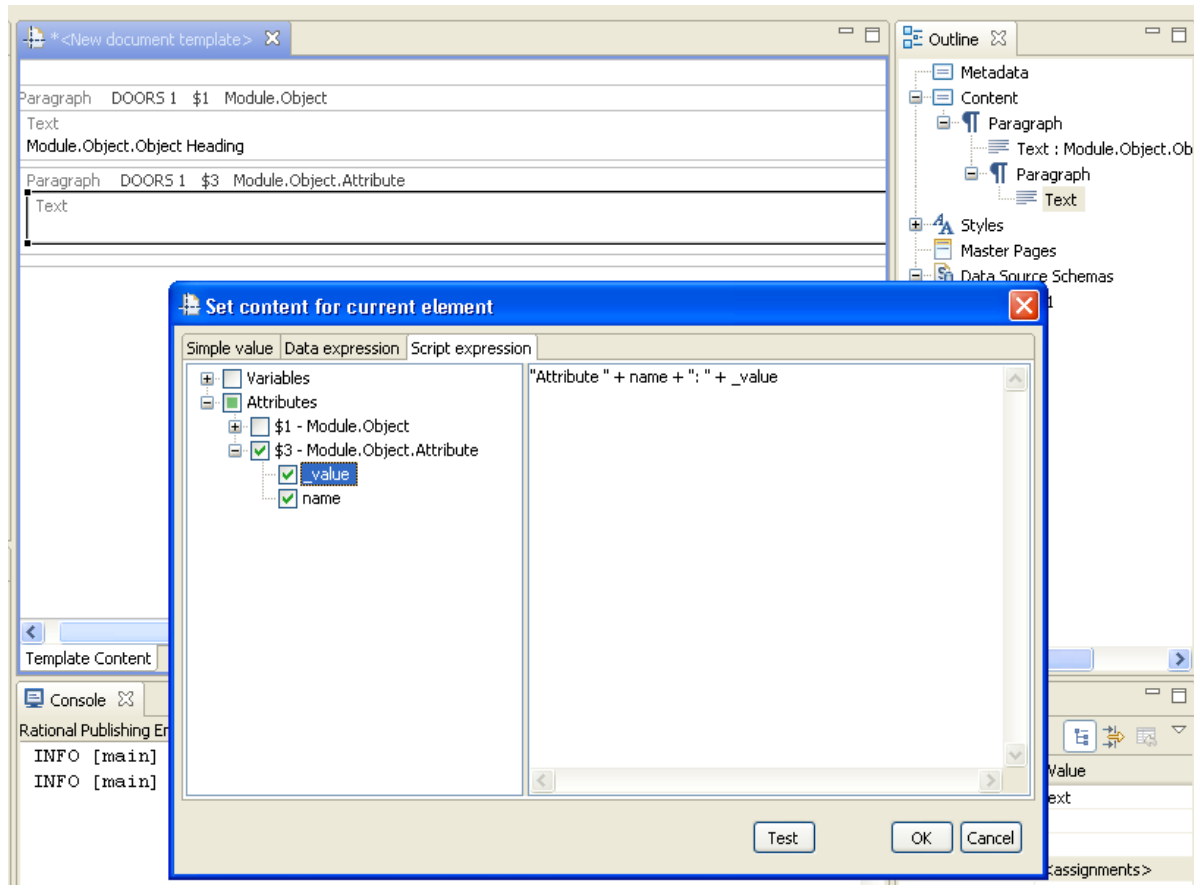


Figure 131 Using Java Script to build the output value

You can use any data attribute from the current context (the element's query attributes and the attributes from all the parent elements' queries).

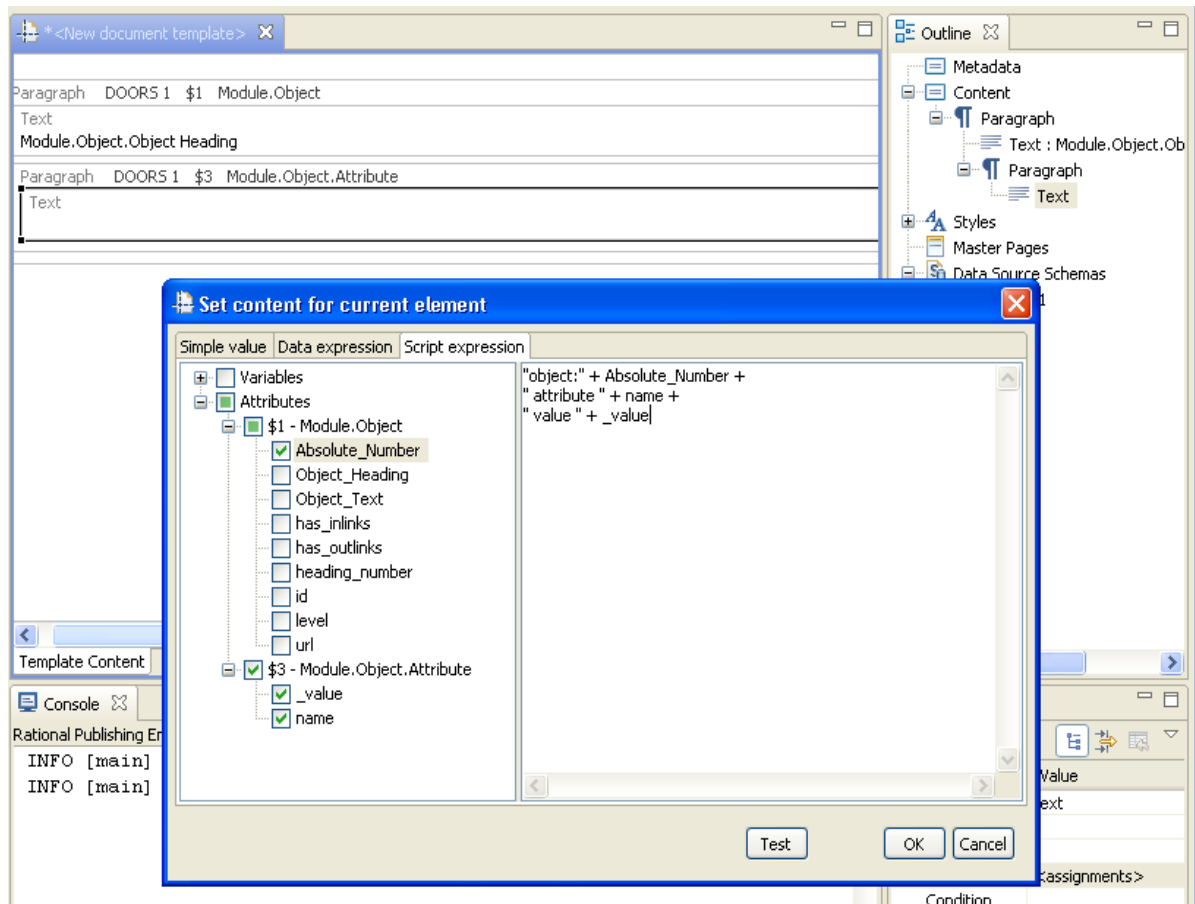


Figure 132 Combining attribute values

NOTE When using an attribute in a script expression you need to mark it as used in the attributes tree in the Expression Editor. The expression editor will report errors for attributes used but not declared as used.

Conditions

Using expressions based on data attributes or template variables you can define conditions for when an element should be rendered. The condition is a JavaScript expression that evaluates to a *Boolean* value.

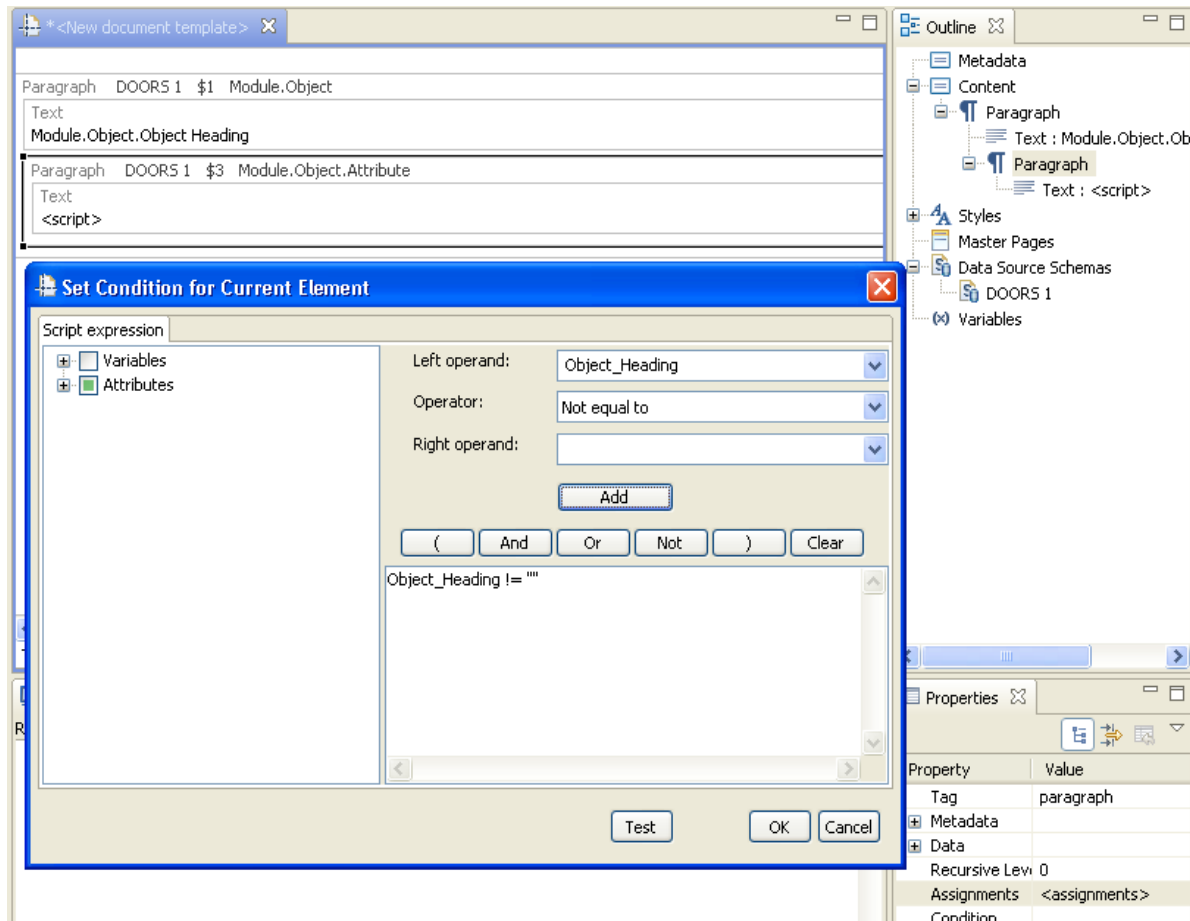


Figure 133 a condition for a template element

In the above example the second paragraph and its contents are processed only if the “Object Heading” attribute is not empty

Filter vs. Conditions

While looking similar in purpose the Filters and Conditions are two different mechanisms that server different purposes.

A **filter** can be evaluated as data is retrieved from the data source.

A **condition** is evaluated only after the data has been extracted from the data source. So while you can use conditions instead of filters, using filters yields better performance as RPE will only process a subset of the data.

A **condition** is evaluated only once for an element even if that element is a query. In consequence you cannot use the current query as a context for a condition.

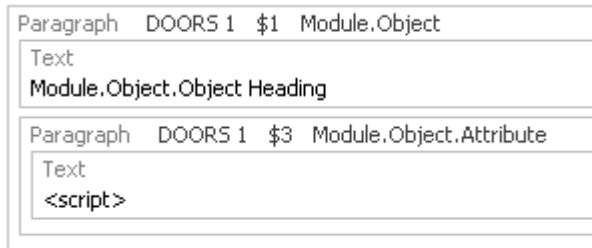


Figure 134 Nested contexts

In the above example the 3 elements can use the following contexts in their conditions:

- **paragraph 1**– none
- **paragraph 2**– attributes of the *module.object* query
- **text** - attributes of the *module.object* query and attributes of the *module.object.attribute* query

The first paragraph cannot use attributes of the *module.object* to define its condition. If that kind of behavior is needed you need to apply the filter (scripted or native) on the container element.

Conditional formatting

You can use expressions to define formatting properties based on data attribute values. The conditional formatting are similar to element conditions but their return values depend on the property type.

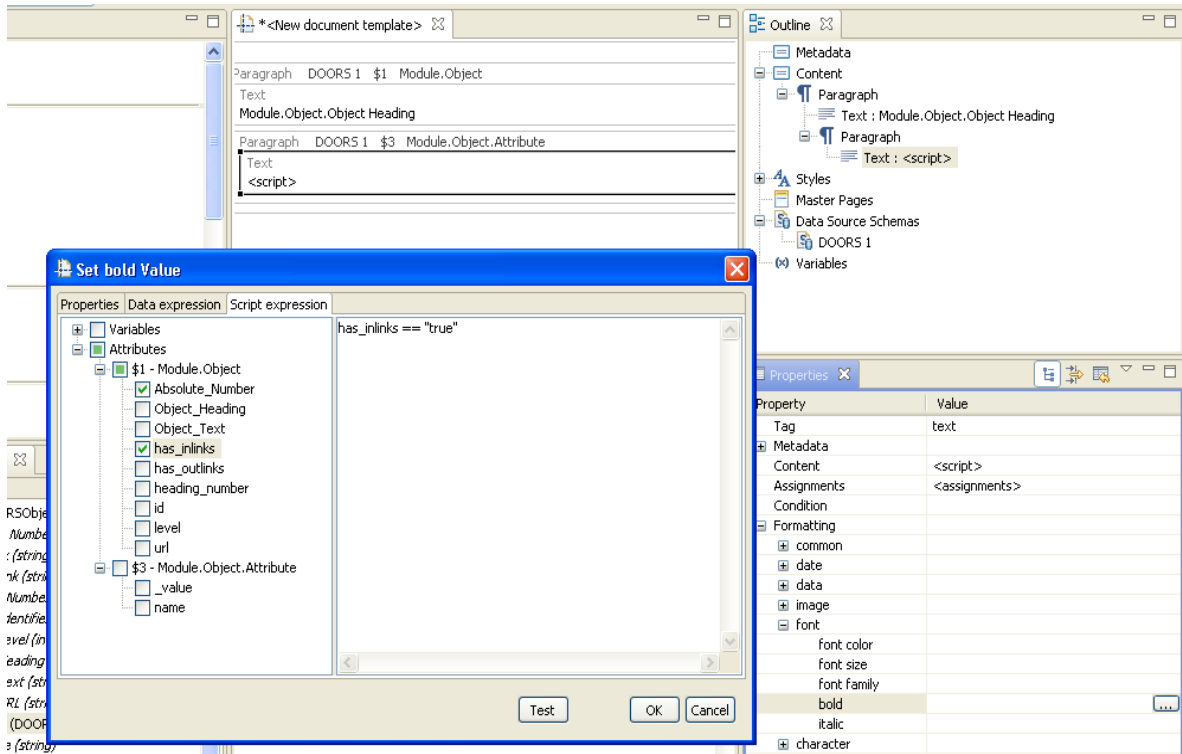


Figure 135 conditional "bold" property

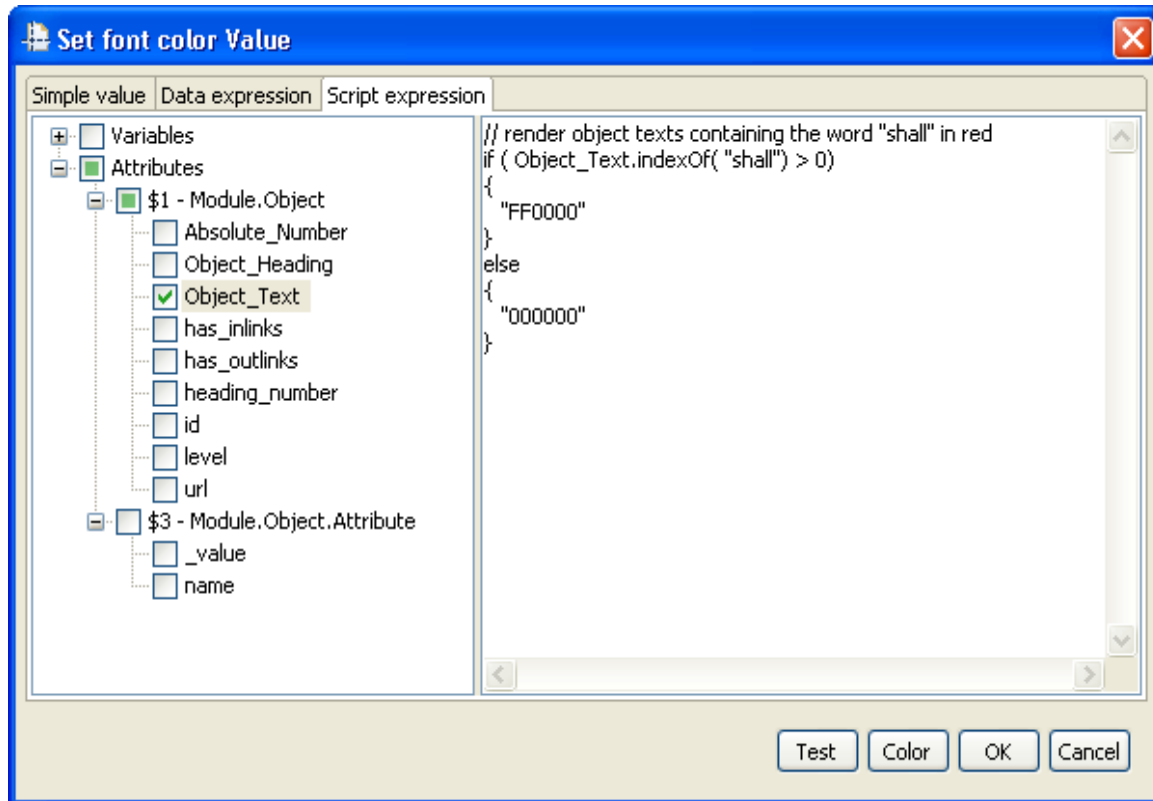


Figure 136 conditional “color” property

NOTE The condition verifies if the object text equals requirement and not if it contains the object requirement.

NOTE Assuming the condition is changed to check if the object text contains “shall”, the formatting will still be applied to the whole object text and not only to the requirement word.

NOTE For color properties you can use the color picker to get the color code.

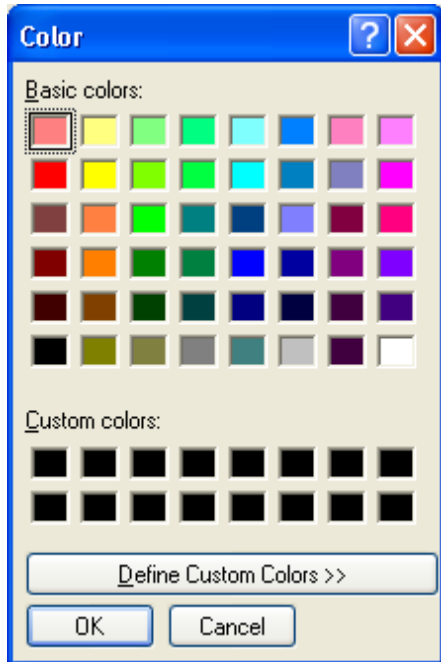


Figure 137 Color picker

NOTE The color picker is available in the Simple value and Script value tabs of the expression editor.

Styles

RPE allows creating styles and applying them to template elements. Unlike the styles available in a word processor application such as Microsoft Word, the RPE styles can combine properties for any given elements. When creating a style you need to define the list of properties that make the style and then provide values for them.

Creating a style

You can create styles using the style wizard. The wizard is accesible from the main menu or from the context menu of the Outline View.

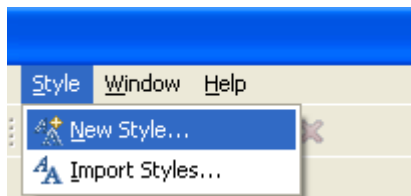


Figure 138 Style menu

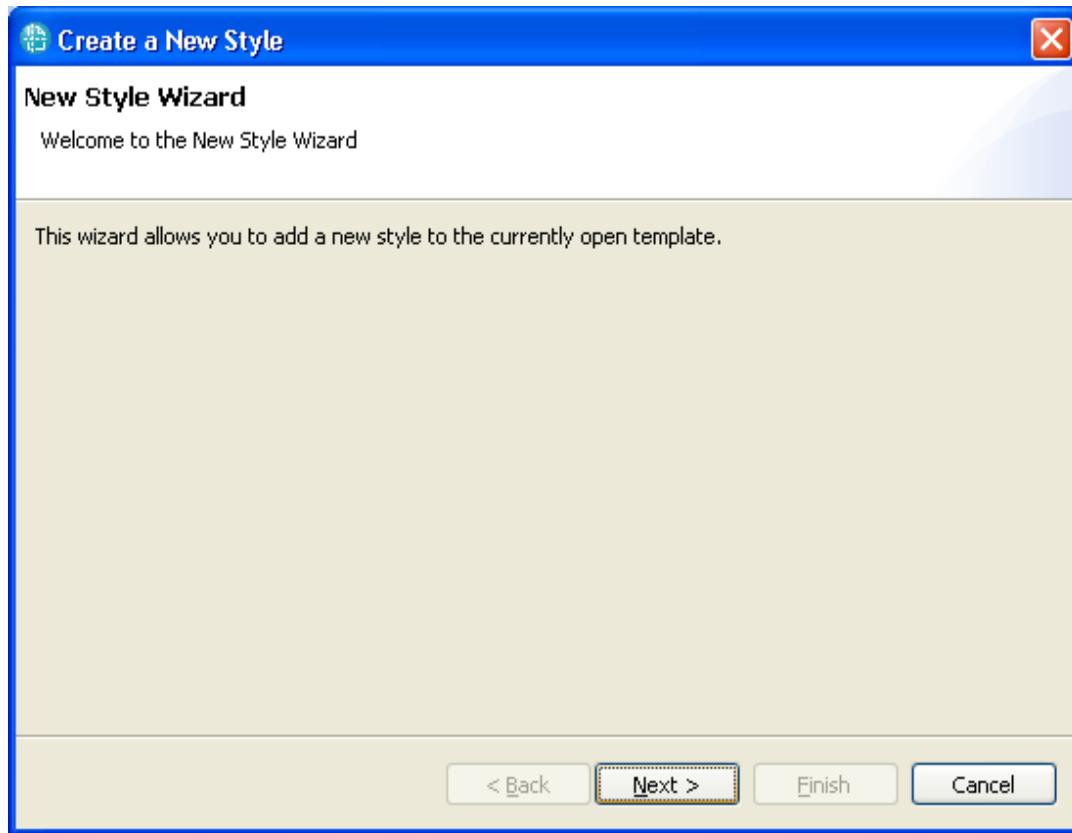


Figure 139 style wizard - welcome screen

The first step is to provide a name for the style. RPE will check if that name is already in use and will prevent continuing until a unique name is entered.

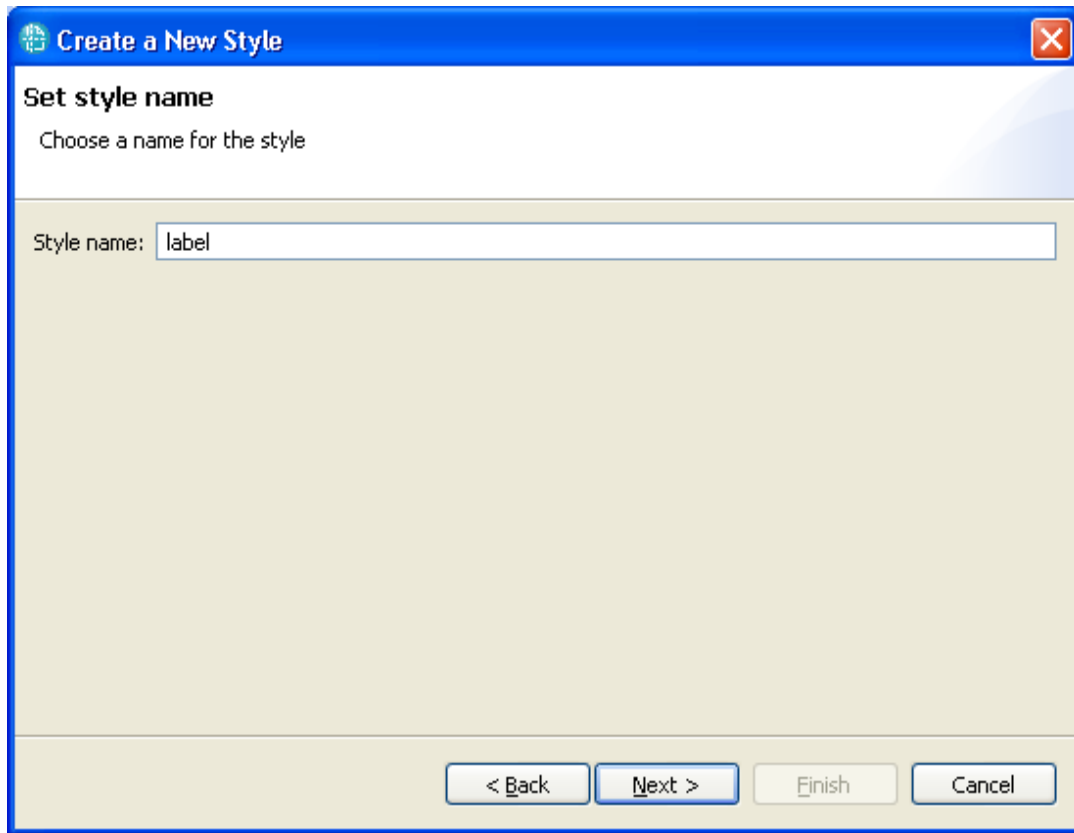


Figure 140 style wizard - style name

Once the style name is selected you can define the properties that this style will apply to template elements. The left pane contains all the available formatting properties

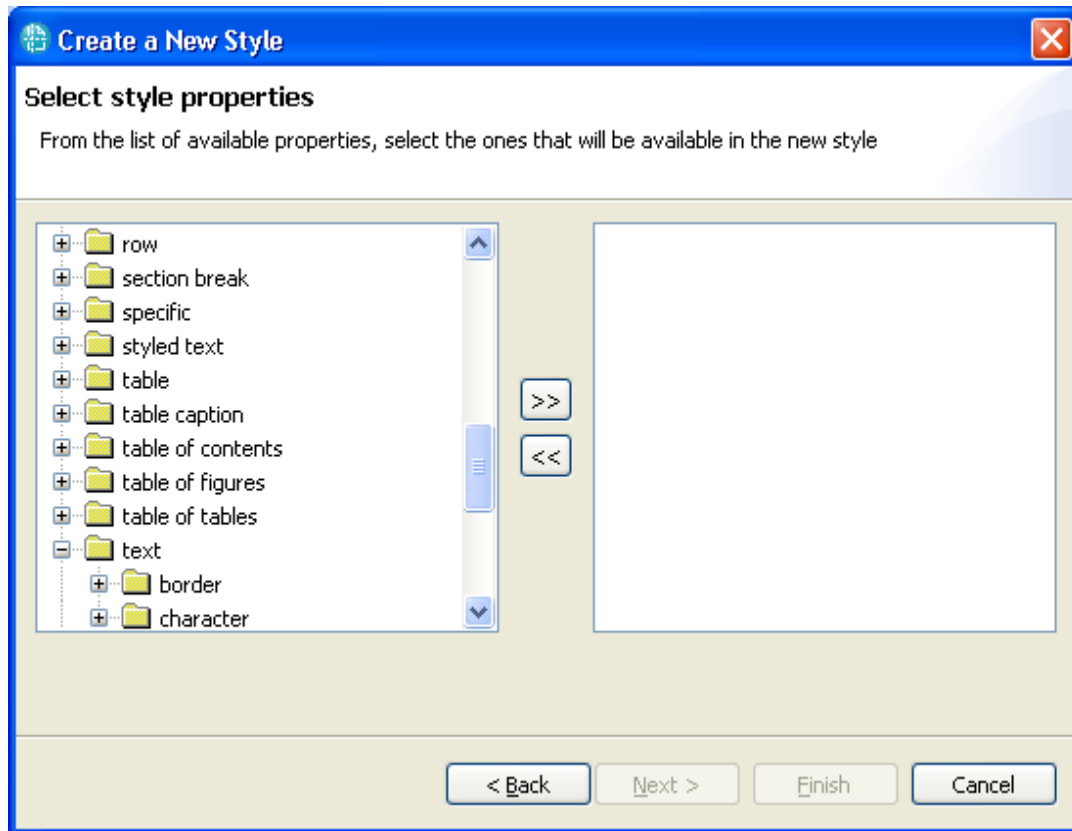


Figure 141 style wizard – style properties

To define which properties the style will apply you need to select properties or groups of properties from the left pane and move them in the second pane.

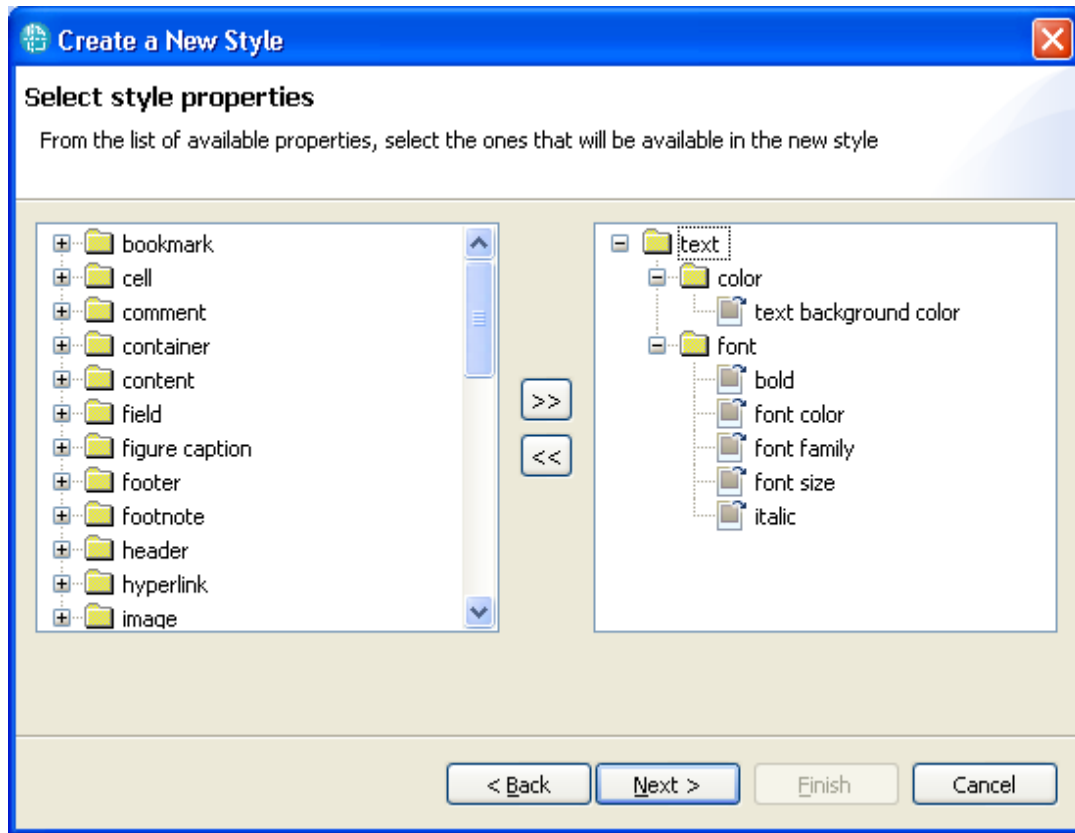


Figure 142 style wizard – select properties

Once the properties are selected you can provide the values for them.

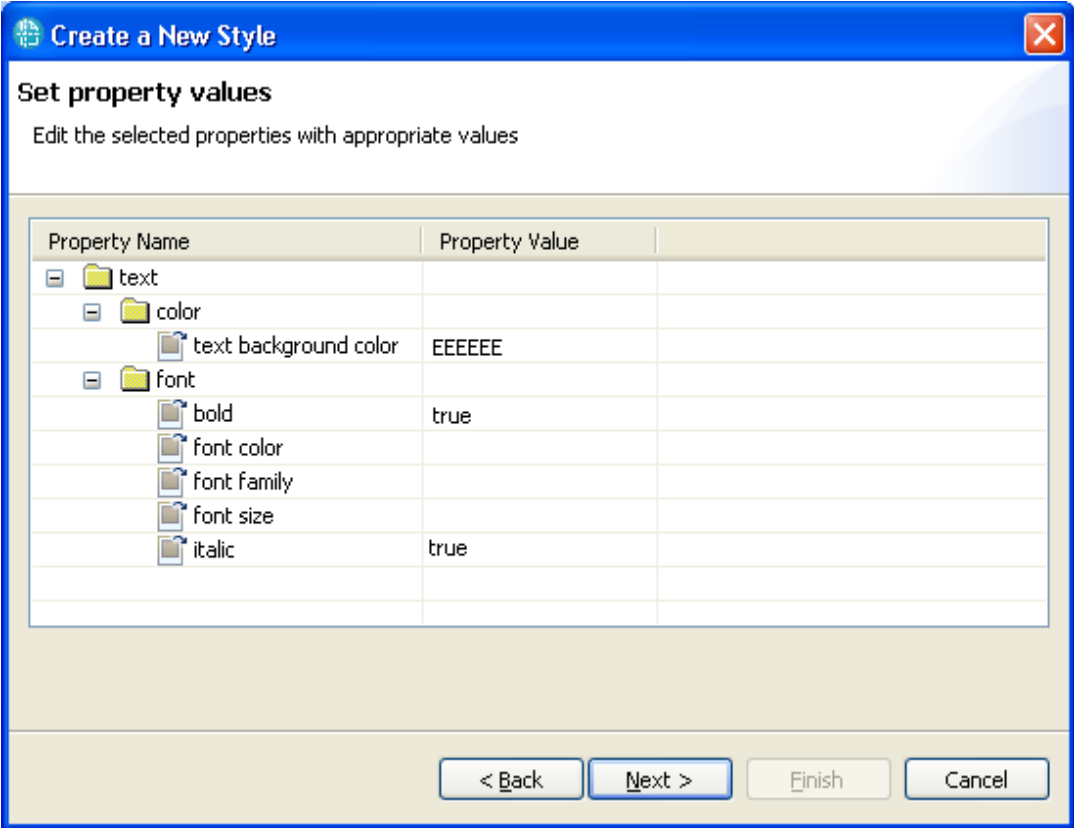


Figure 143 providing values for style properties

Before closing the wizard will display a summary page with the selections made.

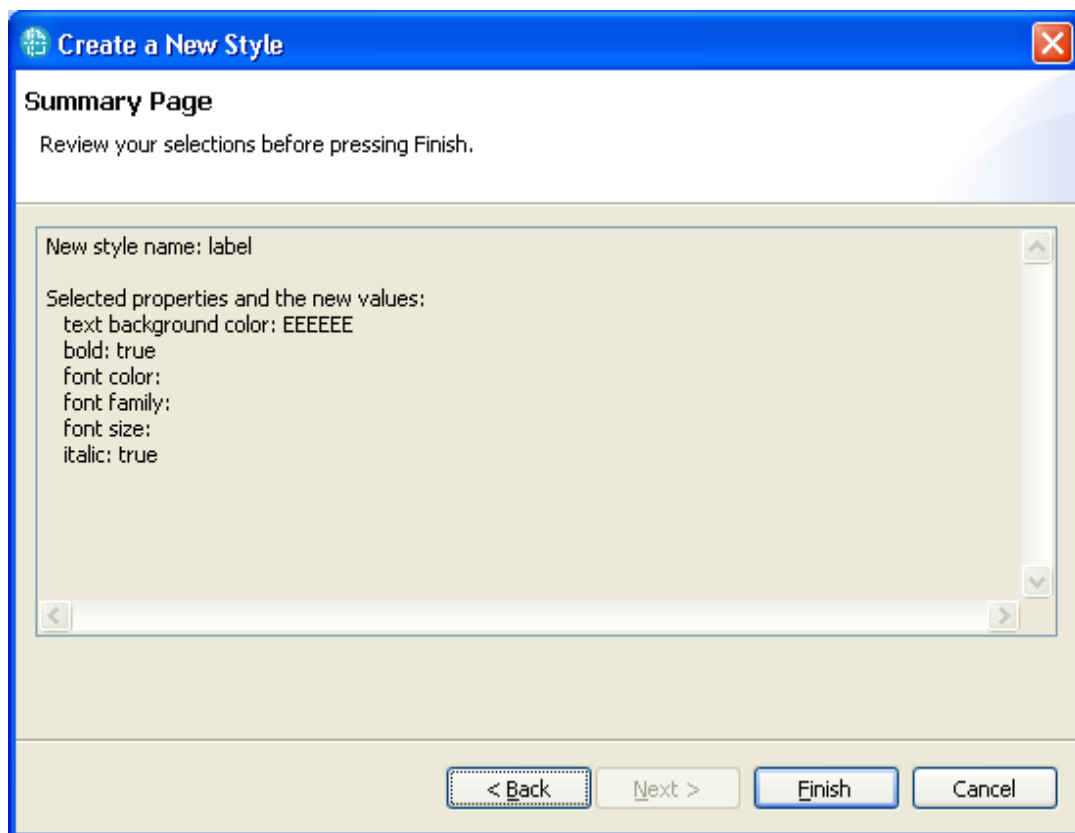


Figure 144 style wizard – summary page

Editing a style

Once you have created a style you can add new properties to it at any time using the “Edit style ...” function from the context menu of the Outline view. The previous wizard will open once again allowing you to add/remove properties from the style.

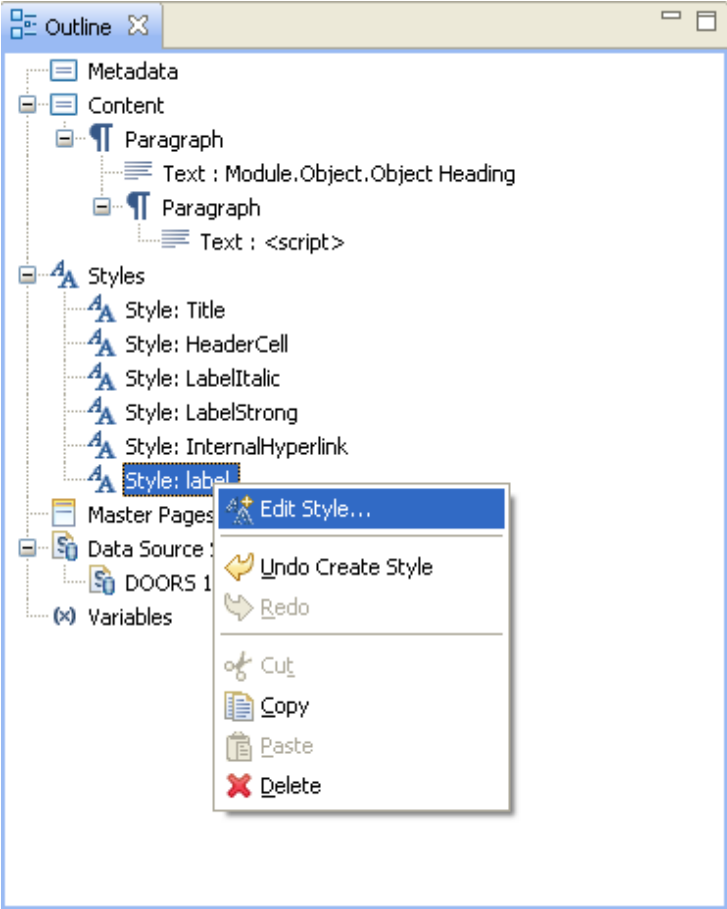


Figure 145 context menu entry for editing a style

Changing style properties

You can change the values for the style’s properties using the wizard or through the properties page.

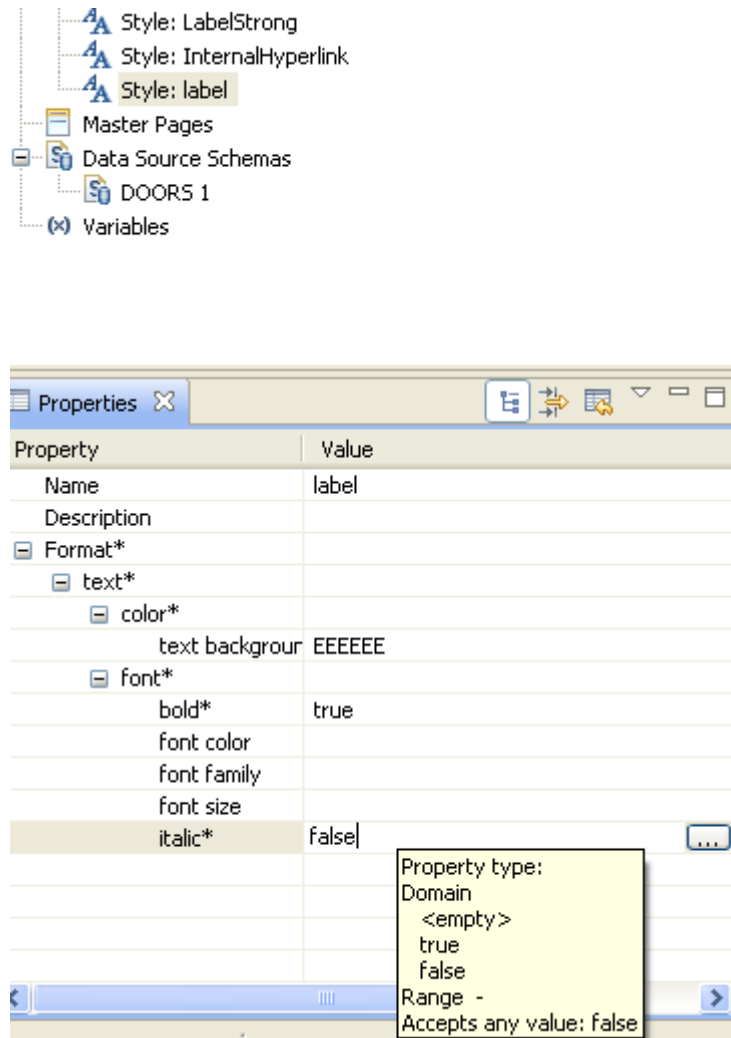


Figure 146 Editing style property values

Applying a style

To apply a style you can drag it from the outline page and drop it on the desired element.

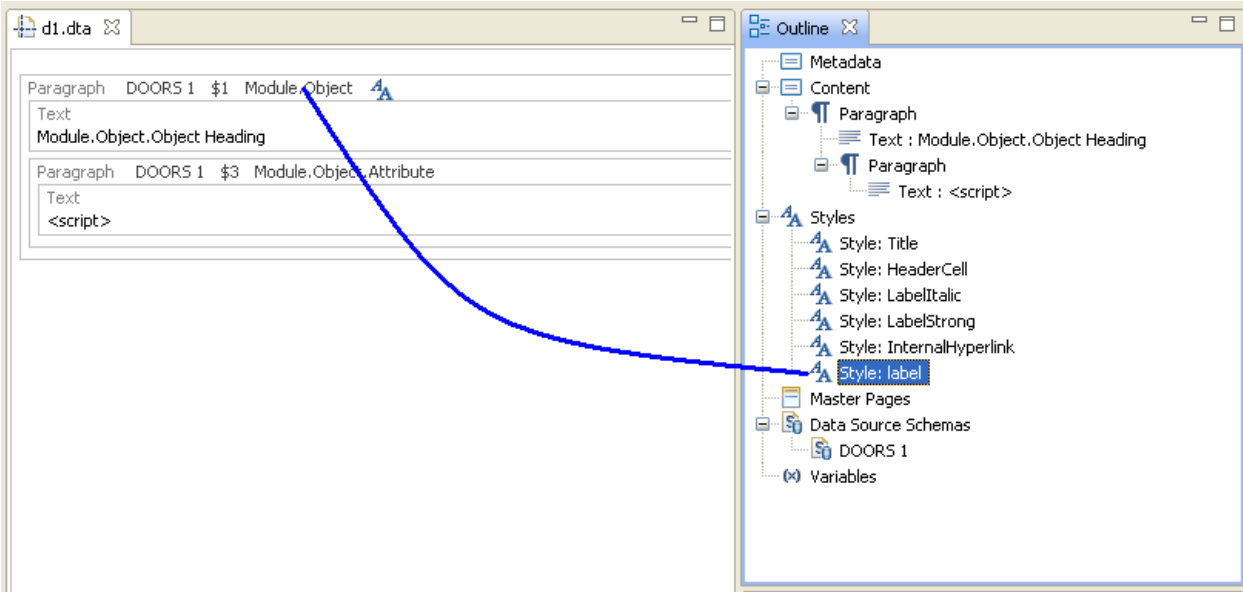


Figure 147 dragging a style on a template element

You can also type the style's name in the "style name" property of the desired element.

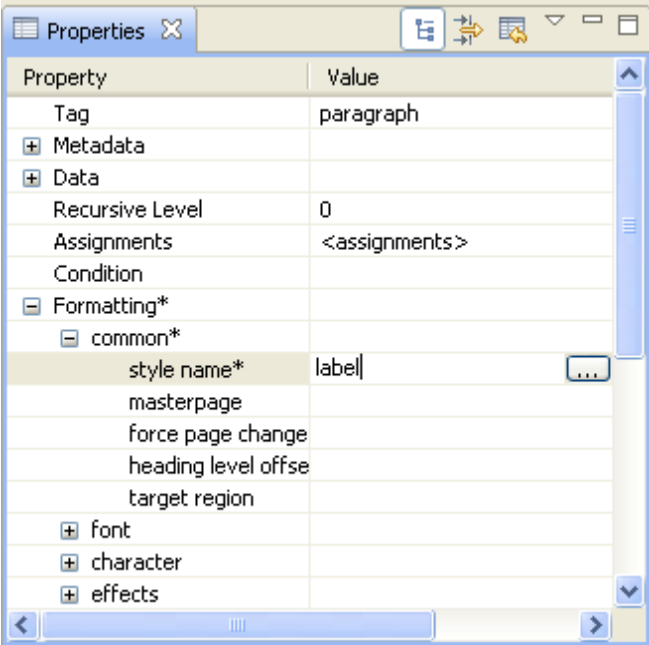


Figure 148 Set the style name through the properties page

Using external styles

RPE can use styles that are not defined in the document template as long as they are provided in the stylesheet. All you need to do is type the name of the style in the element's property page as you would do with a RPE style. When the output is generated RPE will try to use that style if present in the stylesheet. If a style with that name is not found, no style will be applied to that element.

Style precedence

When the document is generated, if RPE finds a style with the same name as a style defined in the document template, the stylesheet style is used. This applies to Word output only.

For HTML output you can manually remove the RPE style from the generated stylesheet so that the provided stylesheet is used.

Style inheritance

If a style is applied to a template element that also has values set for individual formatting properties, the element's changes will override the style ones if such an overlap exists.

Reserved style names

You can use any name for the style you define in RPE excepting styles named 1,2,3,4,5,6,7,8,9. These are reserved style names in RPE that are matched to heading styles in the output formats. 1 is translated to "Heading 1" for Word output and "H1" for HTML.

Master pages

A document template has no concept of pages; it is a single continuous flow. This is because a document template defines data placeholders so it is not possible to know at design time when a page will start.

But that does not mean RPE doesn't support the page concept. RPE handles pagination through "Master Pages". A master page is defined by a header element, a footer element and the page's properties such as page orientation, page borders etc.

A master page is edited in the same way as you edit the template's content with two major restrictions:

- You cannot put queries or data attributes in the master page (though the same result is possible via alternate means)
- You can add elements only inside the Header/Footer elements

NOTE A master page is not applied if it's the same as the current master page and the "Force page change" property is set to false.

NOTE For Word output, every Master Page change will generate a section break.

Adding a master page

You can add a masterpage from the main menu or from the outline view's context menu.

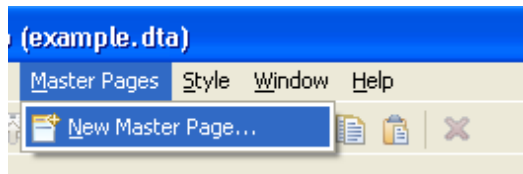


Figure 149 adding a master page from the main menu

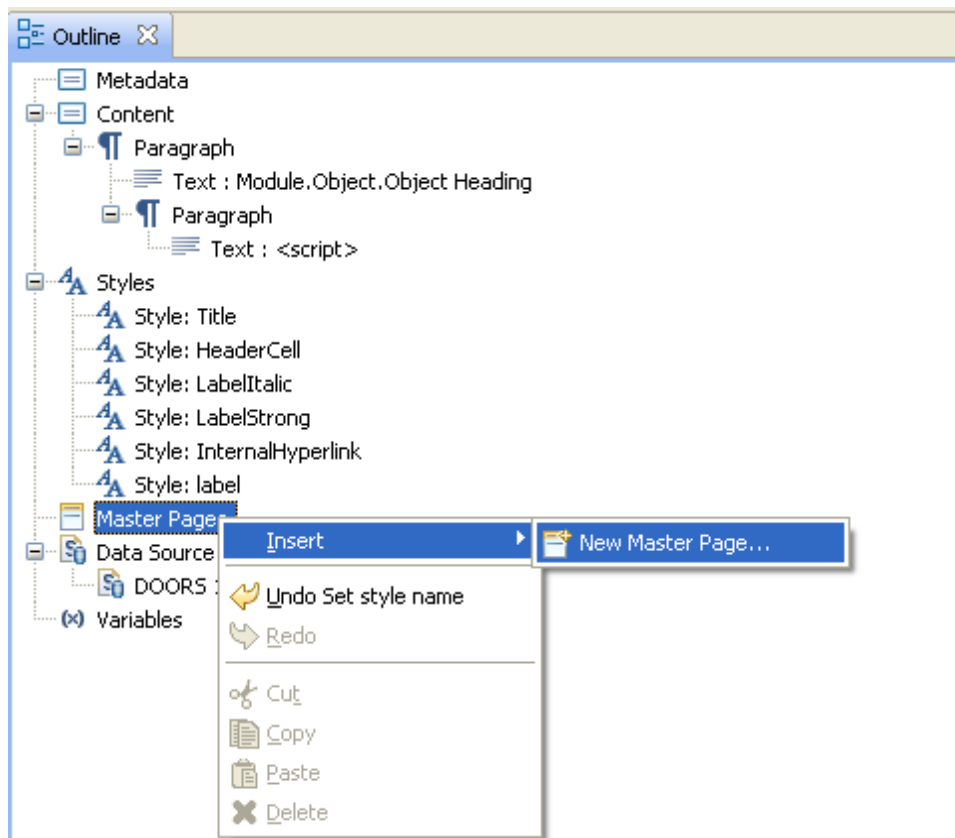


Figure 150 adding a master page from outline view

RPE will display a simple dialog for you to enter the name of the new master page.

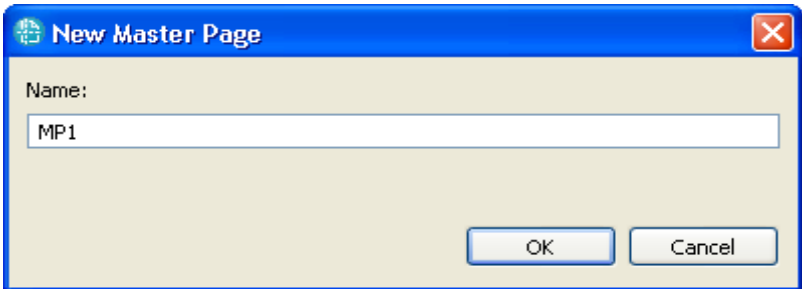


Figure 151 New master page dialog

NOTE RPE will not allow you to create master pages with identical names. Once added the master page is visible in the outline view. Its properties can be edited through the property page and a new tab is added in the editor area.

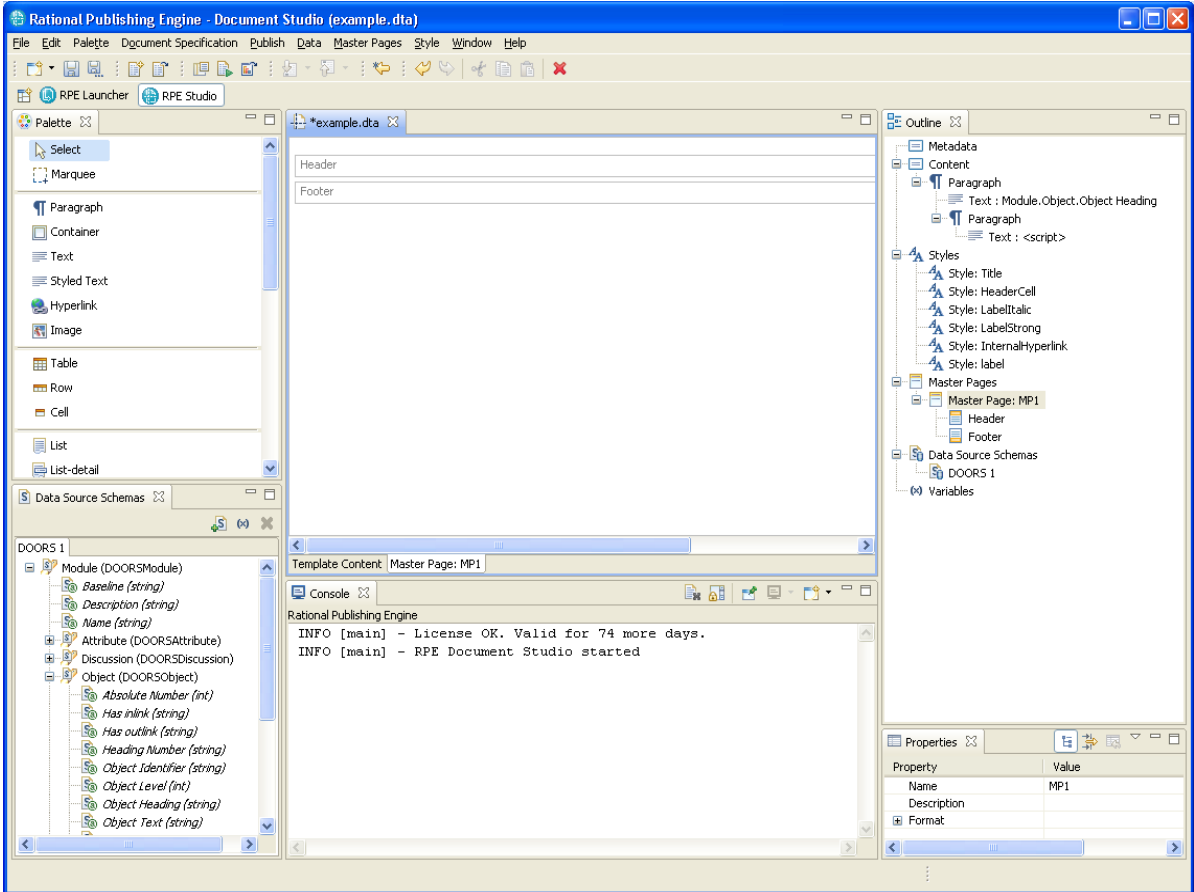


Figure 152 template with master page

Using master pages

A master page is used in same way as a style: you assign the master page to an element.

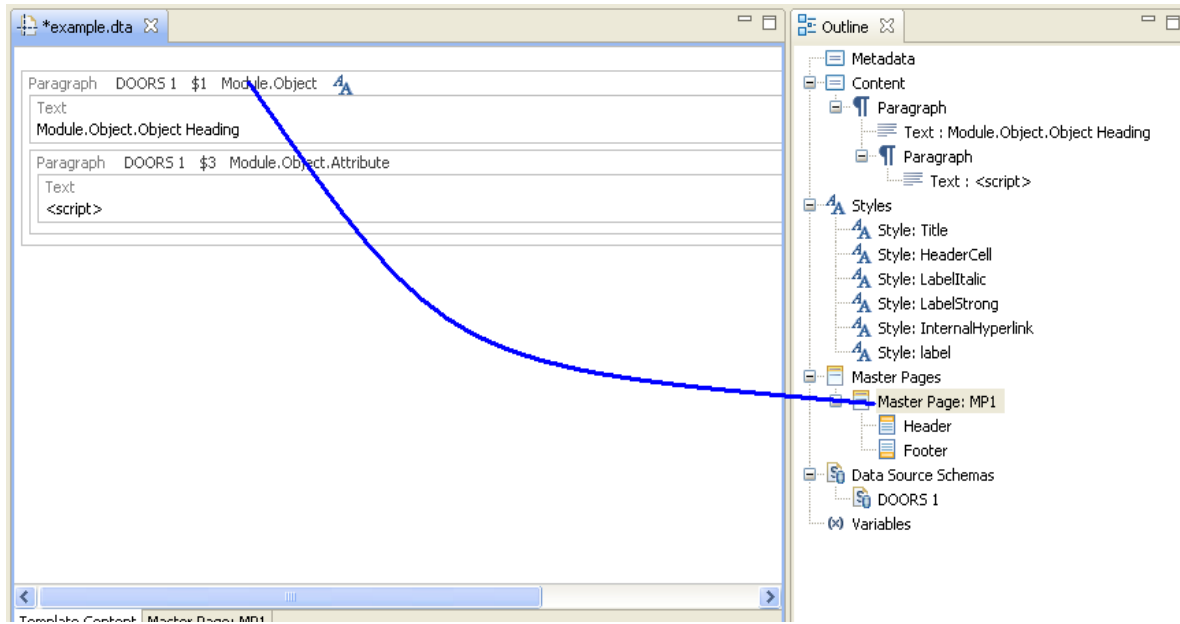


Figure 153 assigning a master page using drag&drop

Alternatively you can enter the name of the master page in the “masterpage” property of the desired element.

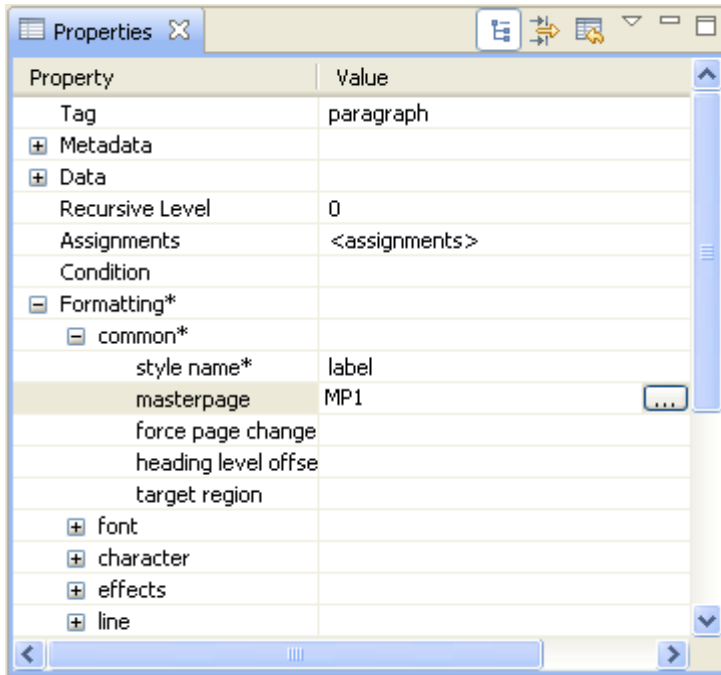


Figure 154 assigning a master page from the properties page

NOTE When the output is generated, RPE will apply a master page only if different from the last used master page. This is to prevent having a new section created in the output document for each element in a query. If that behavior is needed you have to set the “force page change” property to true for the element having assigned the master page.

NOTE For Word output, every Master Page change will generate a section break.

Data attributes in master pages

While you cannot use data attributes directly in master pages, it is possible to have data rendered in there. This is accomplished by:

- Creating a variable in the template
- Using the variable in the master page content
- Assigning a data value to the variable for each element that changes a page

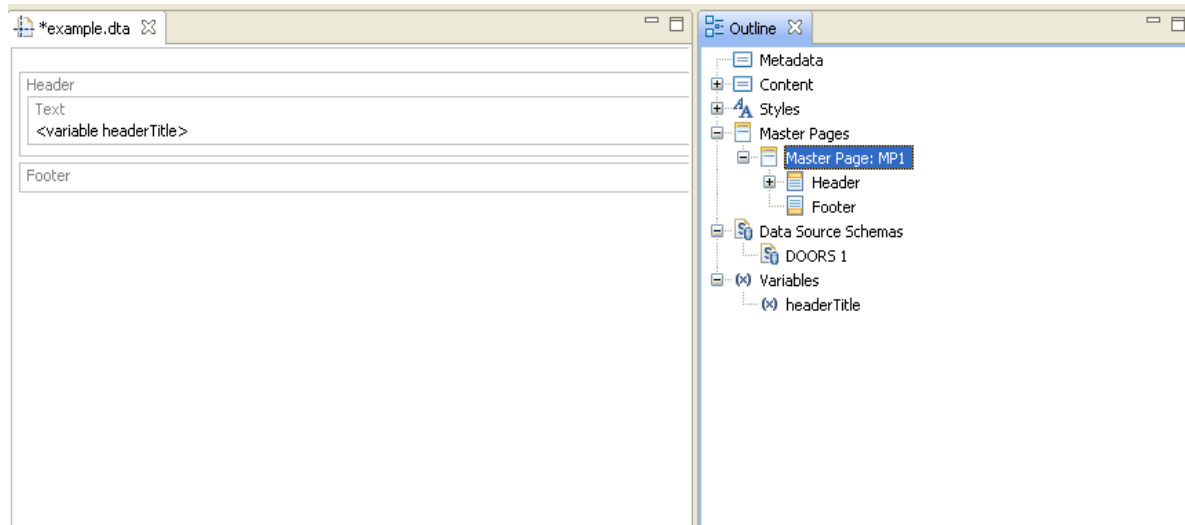


Figure 155 Variable used in the master page content

So that the correct value is used in the header you need to assign the variables value before the master page is used (before the element changing the page).

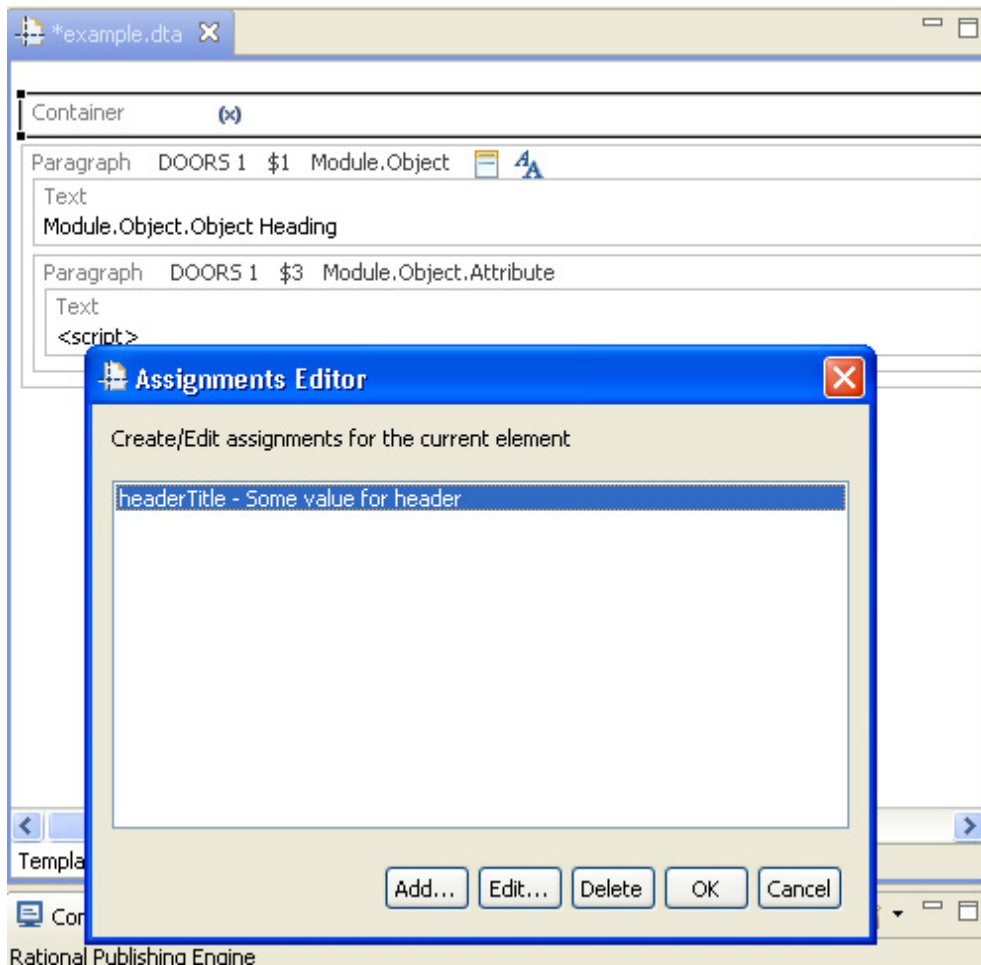


Figure 156 assigning the value for the template variable

With the assignment in place the header of the master page will be rendered using the data content.

NOTE The variable assignment is calculated only when the master page is changed. If a header/footer contains a variable, the header and footer will display the value calculated when a page change occurred. In the above example, if the “force page change” is set to false then the header will display the text for the first object heading, regardless of how many objects with heading are in the module. This is caused by RPE attempting to minimize the number of page changes.

Advanced template concepts

Bookmarks

The RPE bookmark concept is identical to the bookmark concept in Microsoft Word documents or HTML documents. You can use data/script expressions to define the name of a bookmark.

Name mangling

RPE will generate a unique name for each bookmark defined in the template. The unique name is based on the name defined for the bookmark and a random string generated by RPE. RPE will ensure internal hyperlinks use the unique name.

NOTE The unique name you provide must not exceed 20 characters.

If you want to use the bookmark from outside the generated documents you need to set the “ensure unique” property of the bookmark to false. This way, RPE will use the bookmark you’ve provided.

Hyperlinks

The hyperlinks defined in RPE Document Studio are of two types:

- *internal* – to a bookmark defined in the document template
- *external* – to an external URL (<http://www.ibm.com>)

The type of a hyperlink is defined by the *internal* property.

The link location is defined in the *content* of the hyperlink.

The text displayed by the hyperlink is defined by the *display* property and has the default value of “link”.

NOTE You can use data/script expressions to define the content and display of a hyperlink.

Internal hyperlinks

When using internal hyperlinks it is mandatory that the value (content) of the link is identical to the name (content) of the bookmark. This holds true even if the two values are data/calculated values. If the two values are not identical the internal hyperlink will not take you to the expected bookmark.

Variables

RPE template variables are available to be used as placeholders for data calculated at runtime (variable assignments) or provided in the document specification. Unlike data attributes you can also use variables in master pages.

A variable can be added from the outline context menu or from RPE’s main menu.

NOTE The variable name needs to respect Java Script naming conventions.

User variables

User variables are defined by the template designer. You can specify if a variable is “internal” or “external”.

Internal variables are not displayed in the document specification; hence they cannot have values provided by the user. Internal variables are used for calculations or to temporarily store information (such as the variables used to bring data in the master pages).

External variables are displayed in the document specification and they are to be used when their values are provided by the user rather than calculated.

RPE Variables

Besides the user defined variables, RPE maintains a few internal variables that can be used in the template.

Variable	Description
_element_id	The element id as kept internally in the template. Usable for debugging purposes.
_element_level	The recursive level of the current element.
_row_number	The row number in the current table. 1 based. If the element is not a row the value is 0
_cell_number	The cell number in the current row. 1 based. If the element is not a cell the value is 0

Variable assignments

When designing a template it might be necessary to calculate values or make available data attributes in contexts where normally they would not be available. This can be achieved through the mechanism of variable assignment.

The variable assignment allows the template designer to change the value of a variable at any given time. The new value can be static data or data available in the current context.

To create a variable assignment you need to first define a variable. Once the variable is defined you can access the variable assignment function from the “Data” context menu from the editor’s content area.

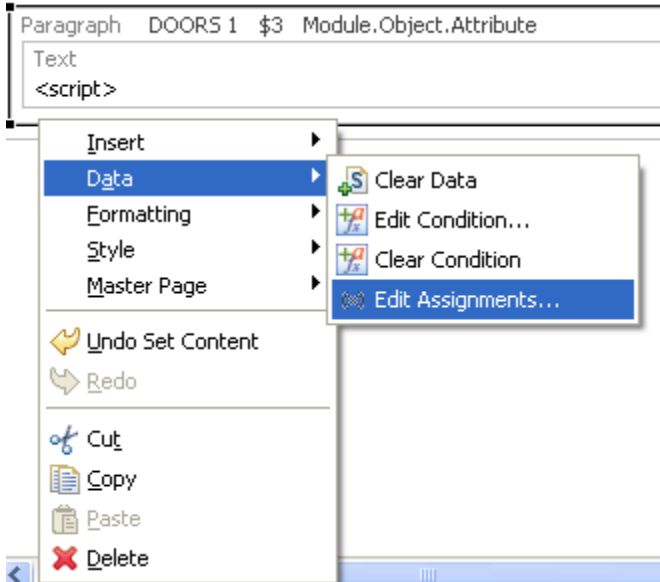


Figure 157 Variable assignments

When used, this function will display a dialog that lists all the assignments for the current element and allows defining new ones.

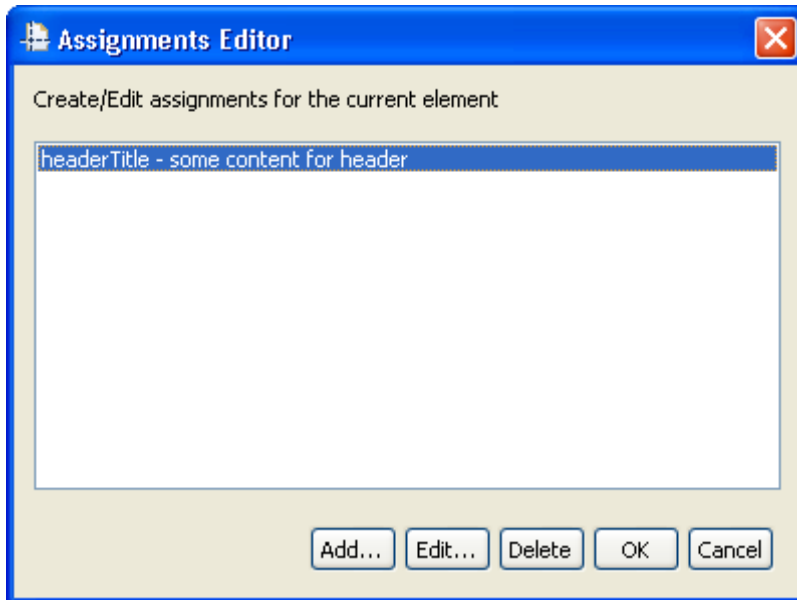


Figure 158 Variable assignments

You can edit or add new assignments using the Content Editor.

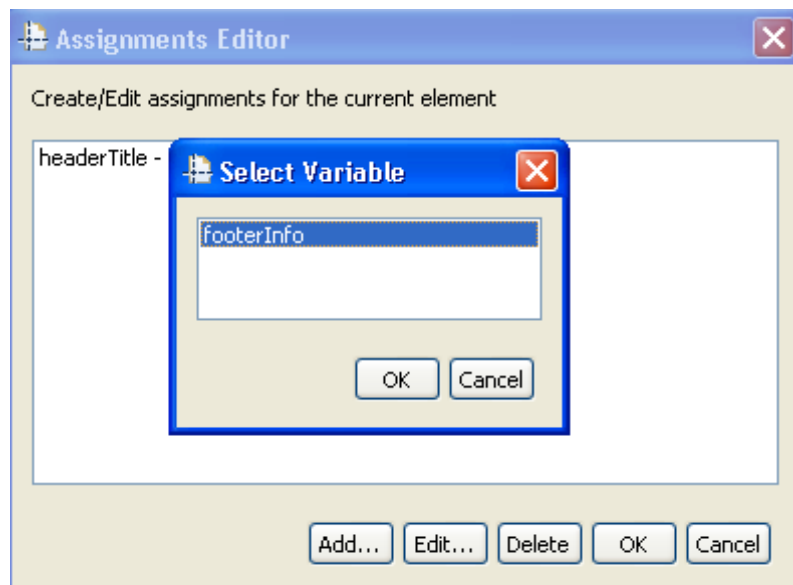


Figure 159 Variable assignments – creating a new variable assignment

When editing the value for the assignment the same rules apply as for defining the content of an element. In other words you can use only the data attributes and variables visible in the current context.

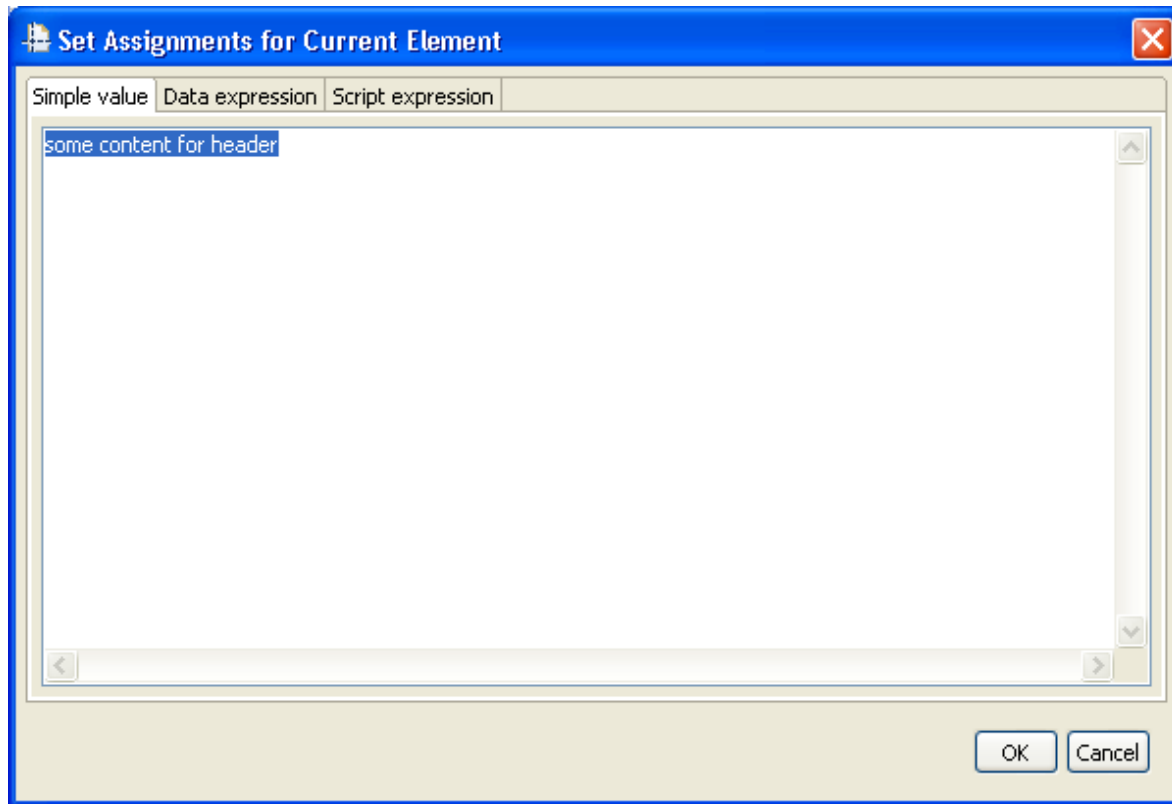


Figure 160 Variable assignment.

RPE Predefined Template

RPE Document Studio comes with a predefined template named “template.dta” located in the “utils” folder. You can consider this template the equivalent for Microsoft Word’s normal.dot. RPE Document Studio uses “template.dta” each time it creates a new document.

NOTE You can customize template.dta to match your organization’s needs. You can define styles, add header content (company logo, legal info etc), define master pages etc.

RPE launcher integration

The RPE launcher comes integrated in RPE Document Studio. There are two run modes for the launcher in this case:

- Synchronized with studio
- Not synchronized with studio

The run mode can be specified in the preference page of RPE Document Studio.

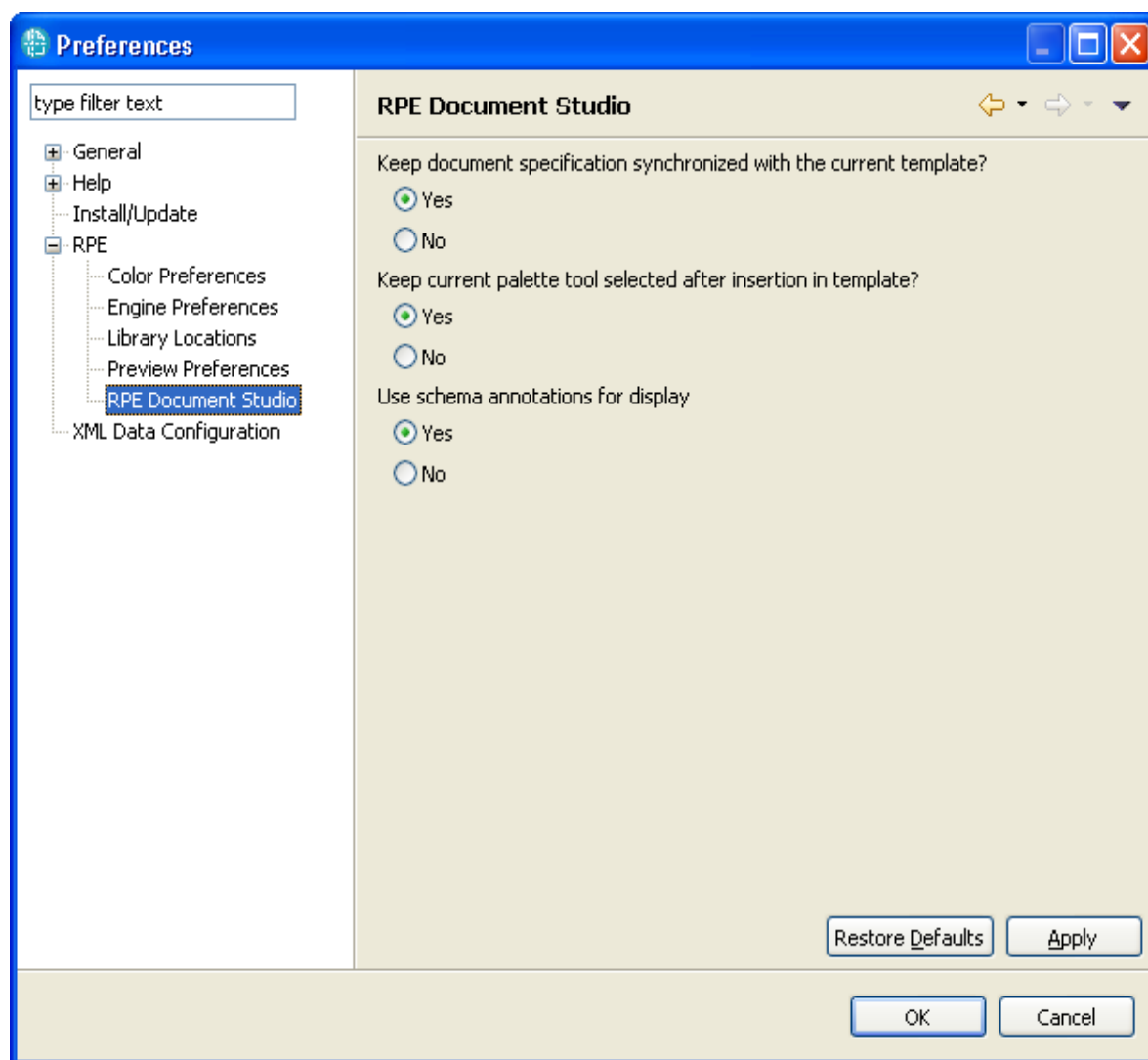


Figure 161 RPE preference page

Synchronized mode

This run mode is meant to support testing of the current template in RPE Studio. While running in this mode it is not recommended to manually load document specifications as RPE Studio actions will modify the document template

RPE Studio Action	Effect in RPE launcher
New template	The existing document specification is closed and a new one is created
Open template	The existing document specification is closed and a new one is created. The newly loaded template is assigned to the document specification
Save template	If this was a newly created template, the template is assigned to the document specification. The document specification is synchronized with the template.
Save as	If this was a newly created template, the template is assigned to the document specification. Otherwise the current template in the document specification is replaced by the new one (with the new path).

Generate & preview

In synchronized run mode RPE will detect if the current template is unsaved and will prompt you to save it before generating the output or the preview. Cancelling the save operation will also cancel the document generation.

Unsynchronized mode

In this run mode, the RPE Launcher operates as it would do as a standalone application and is not affected by operations in Document Studio.

Schema Discovery

RPE provides tools to assist the user to build data source schemas for some data sources. In the current build the DOORS and REST schema discovery mechanisms are provided.

DOORS Schema Discovery

The wizard guides you in building a schema for a specific DOORS module. This will greatly simplify authoring document templates for modules with the same structure (same or similar attribute list).

You can access the function from the main menu.

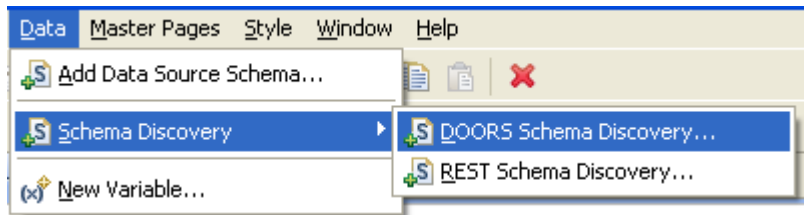


Figure 162 DOORS Schema Discovery

The wizard starts with a welcome screen.

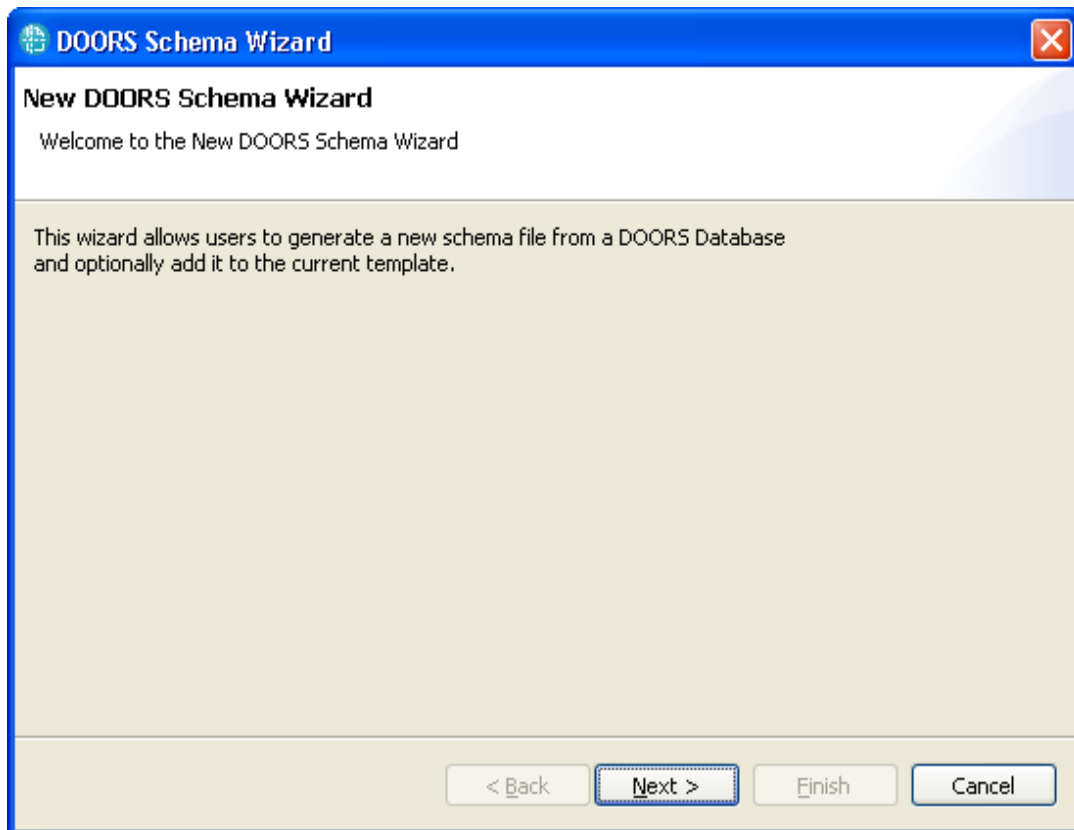


Figure 163 Welcome screen

In order to discover attributes, RPE needs to be able to connect to DOORS. The second page of the wizard does that

DOORS Schema Wizard

DOORS Connection options

Fill the information below

Select the DOORS instance to connect to:

Use running DOORS instance

Run a new background DOORS process

Username: spurlos

Password:

Database: 36677@giediprime

Path to doors.exe: C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors.exe

< Back Next > Finish Cancel

Figure 164 DOORS connection

In the next step of the wizard you can decide if you want to add attributes to the “main” objects (Module.Object schema element) or attributes for linked objects (Module.Object.Link.Linked Object schema element)

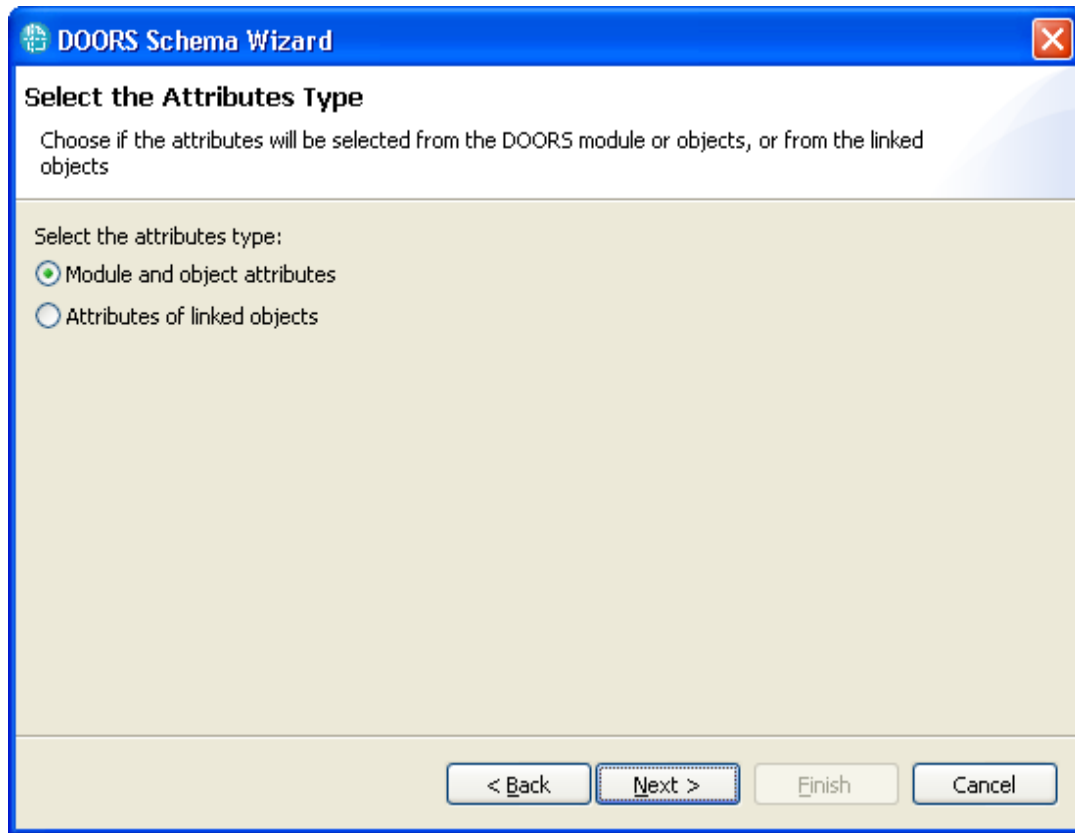


Figure 165

Schema discovery for current module

Select for which module you want to discover its schema

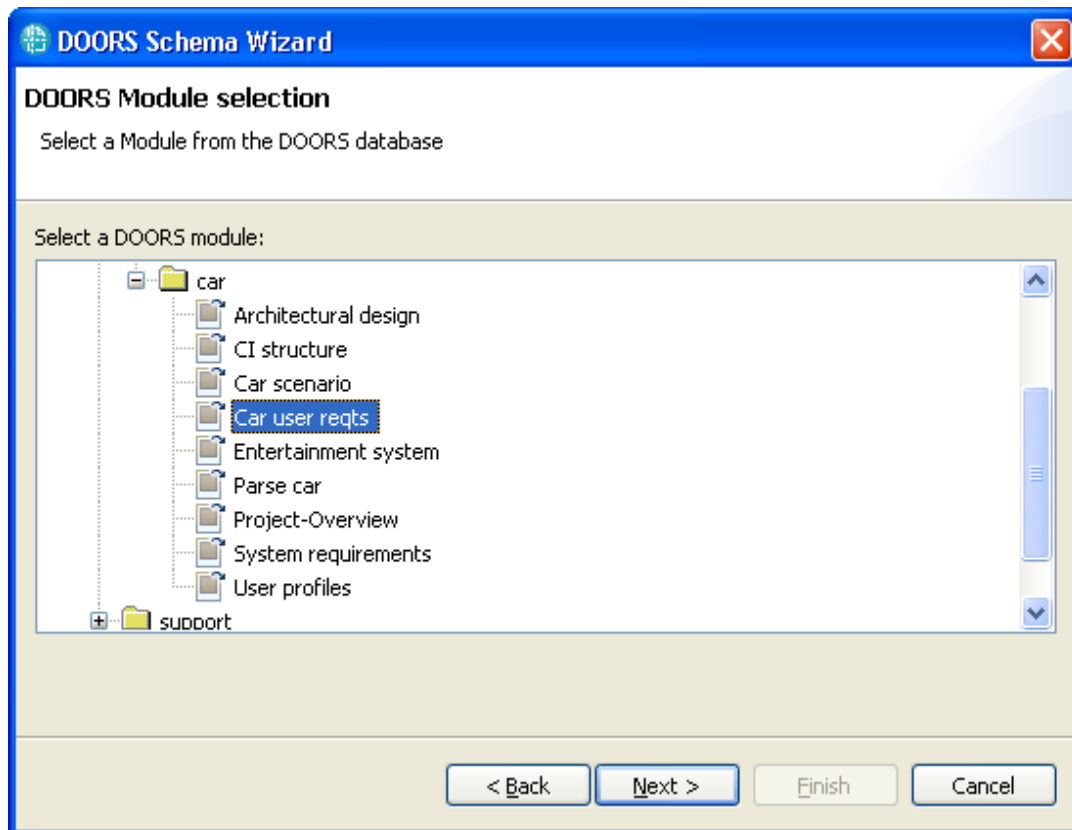


Figure 166 Module selection

Once you've selected the module you need to specify the module's baseline you want to use. The selected baseline will determine which attribute set to use.

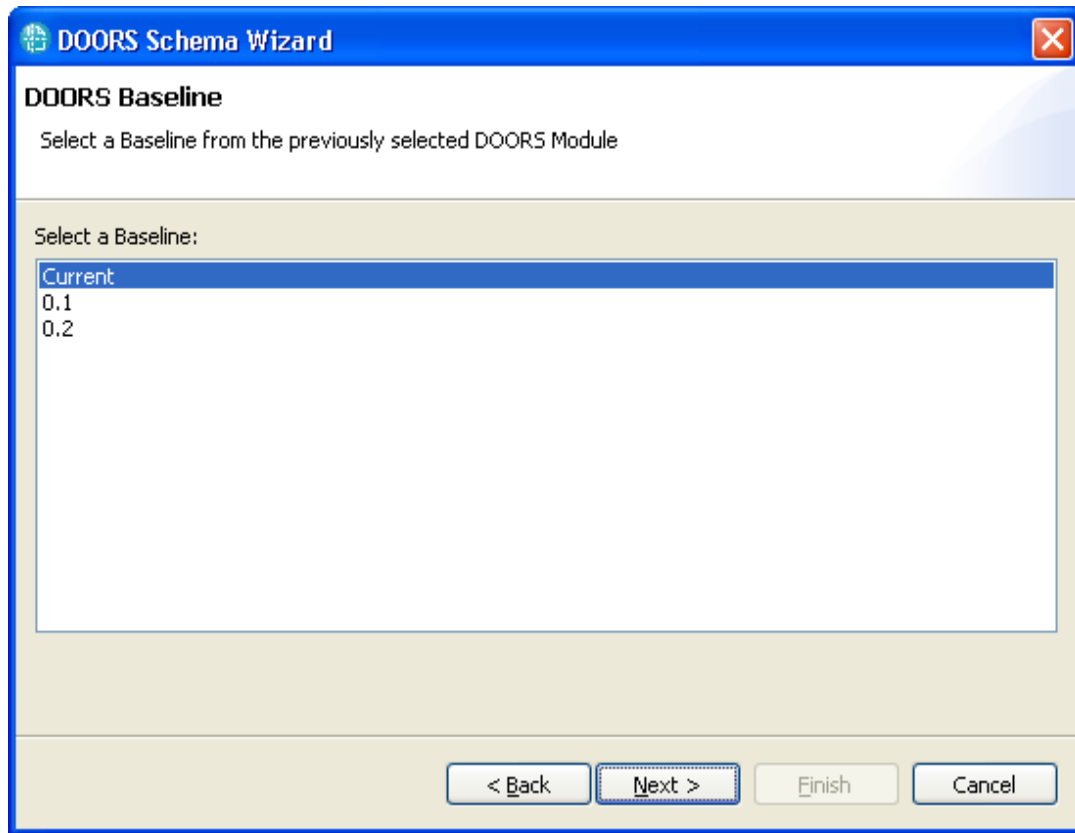


Figure 167 selecting the module version to use

This screen allows you to select the attributes to use. The attribute set is taken from the baseline selected in the previous screen.

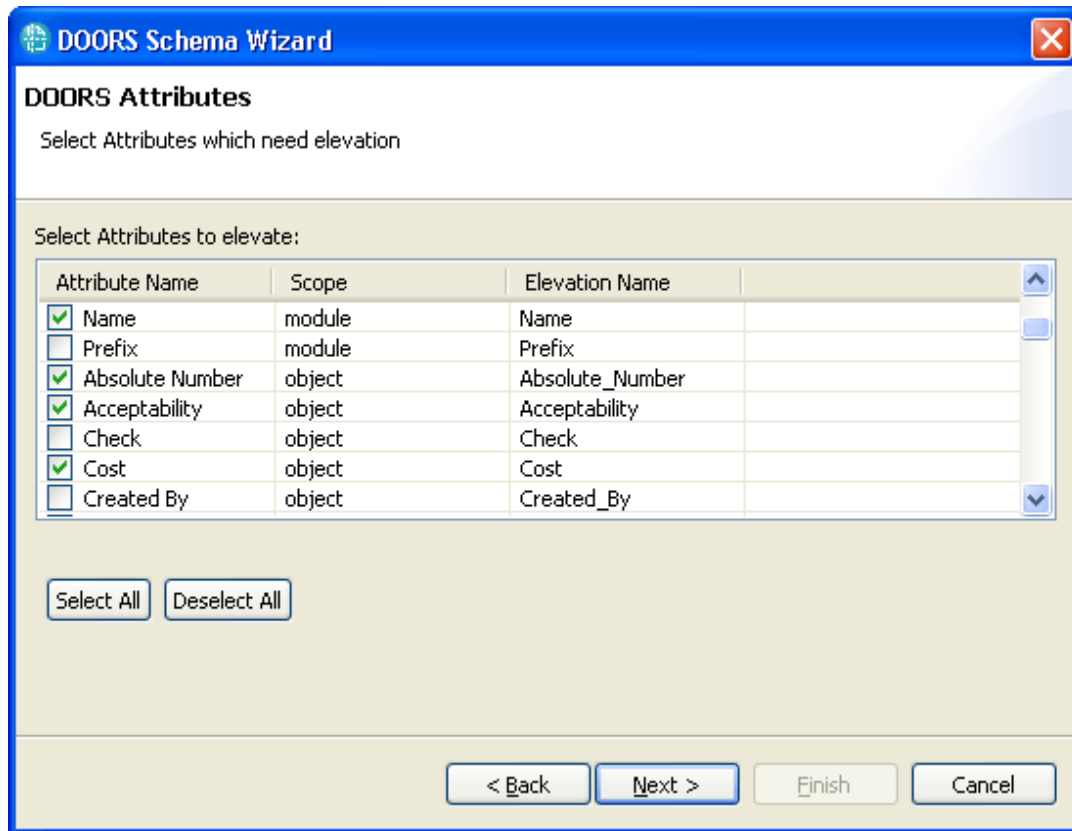


Figure 168 Selecting attributes

An elevated attribute allows direct access to that attribute's value from a *module.object* context. Non elevated attribute values are available only from a *module.object.attribute* context.

NOTE You can select the attribute set from any baseline of the module but you cannot have attributes selected from 2 different baselines.

NOTE The elevated name is the name used in script expressions. Hence it must be a valid JavaScript identifier. RPE will generate a valid name out of the DOORS attribute name and will prevent you from changing it to an invalid name.

NOTE The selected baseline is used for the sole purpose of defining the attribute shown to the user. This information (the baseline used to elevate the attributes) is not used at document generation time. If one attribute does not exist in the baseline used for document generation nothing will be rendered for it in the output.

Select the columns you want to elevate

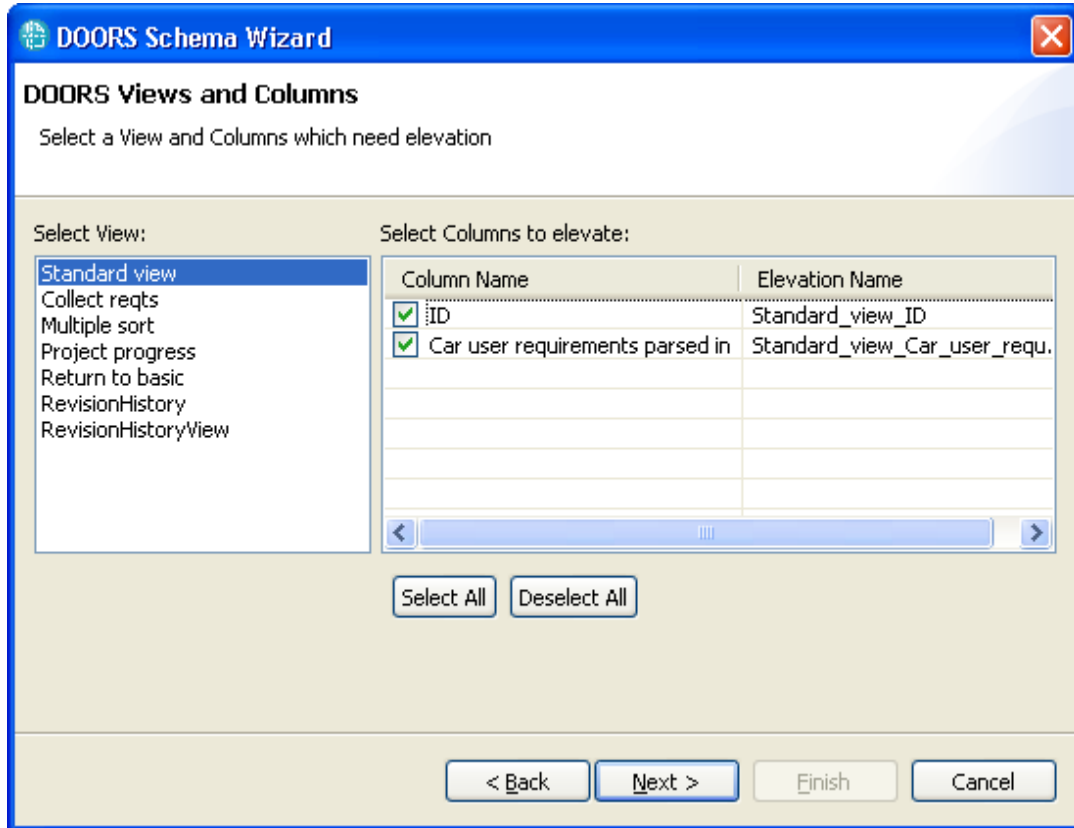


Figure 169 selecting columns

An elevated attribute allows direct access to that attribute's value form a *module.object* context. Non elevated attribute values are available only from a *module.object.column* context.

NOTE You can elevate columns from any number of views.

NOTE With the current RPE version you can elevate only columns that do not contain < or >.

NOTE The elevated name is the name used in script expressions. Hence it must be a valid JavaScript identifier. RPE will generate a valid name out of the DOORS column name and will prevent you from changing it to an invalid name.

Save schema and add it to the current template

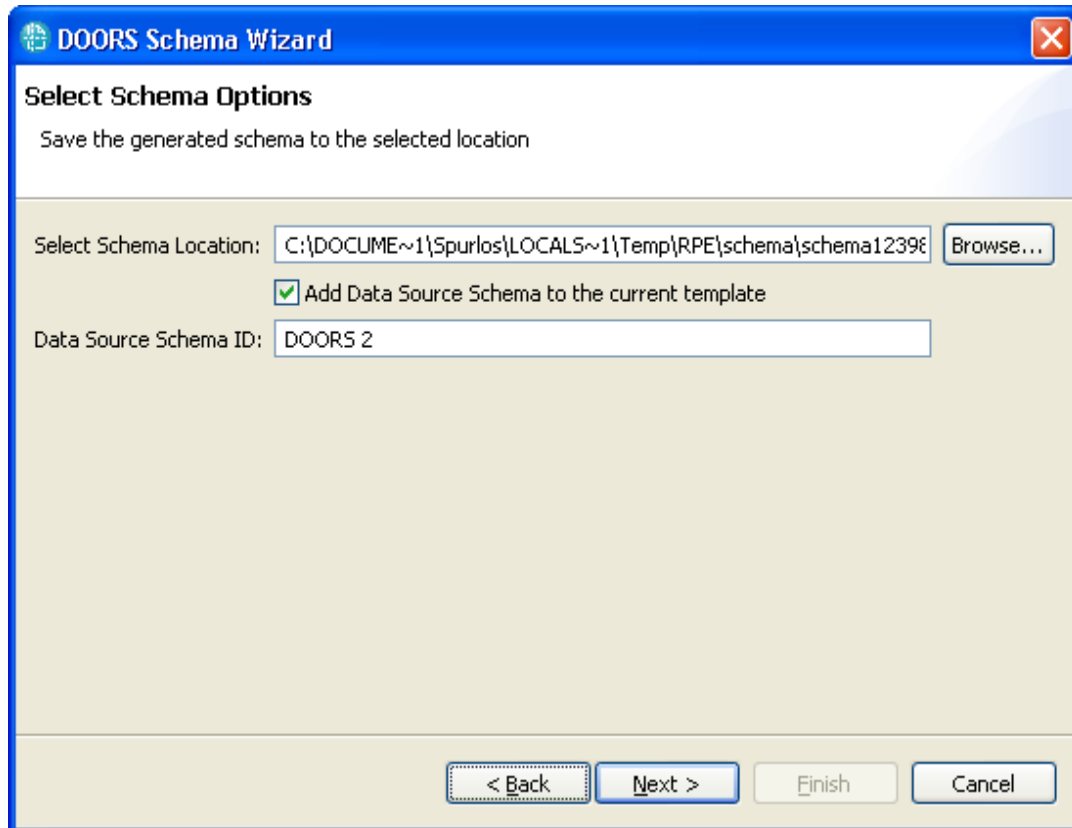


Figure 170

NOTE You should save discovered schemas so you can reuse them in other templates without having to run the schema discovery wizard again.

Review changes and finalize the wizard

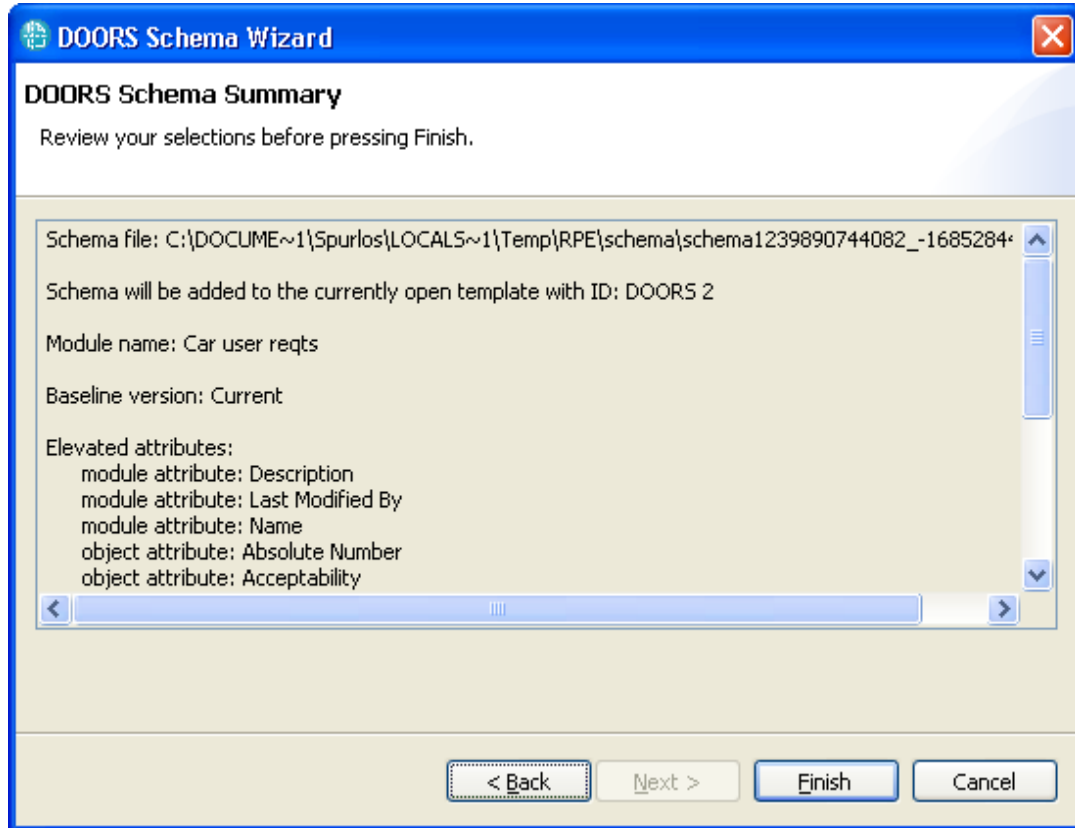
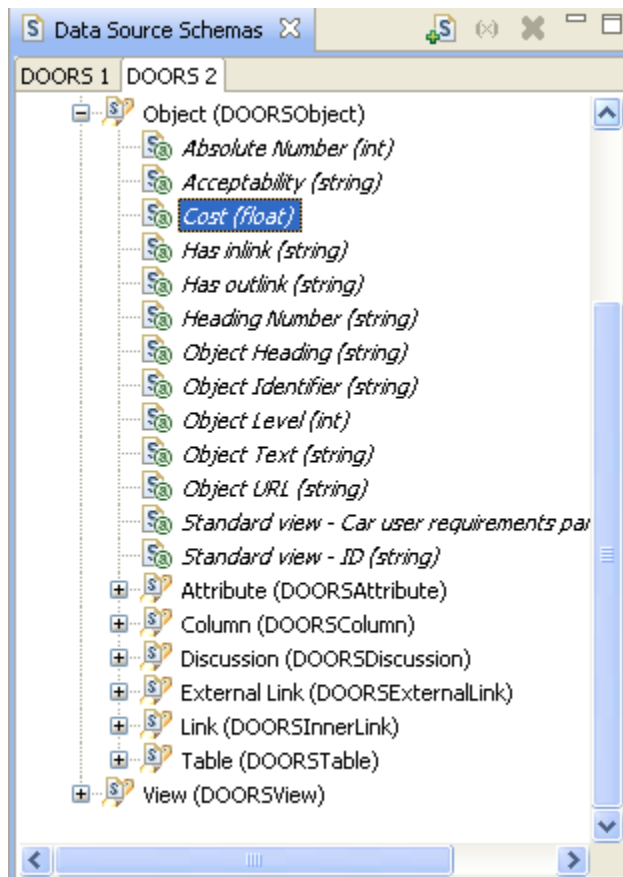


Figure 171 Summary screen

Once the wizard is finalized the schema is loaded in the schema view. You can see the attributes and columns added to the “Module.Object” element.



Linked object attributes discovery

Adding attributes for linked objects works in the same way as attributes for the “main” objects with the exception that you cannot add columns.

This wizard can be run as many times as needed to add attributes from more than 1 module.

NOTE You can have links going to more than one module and you can elevate attributes from all the linked modules. This will result in the linked object containing a reunion of all the attributes from all linked modules.

First you need to select a module.

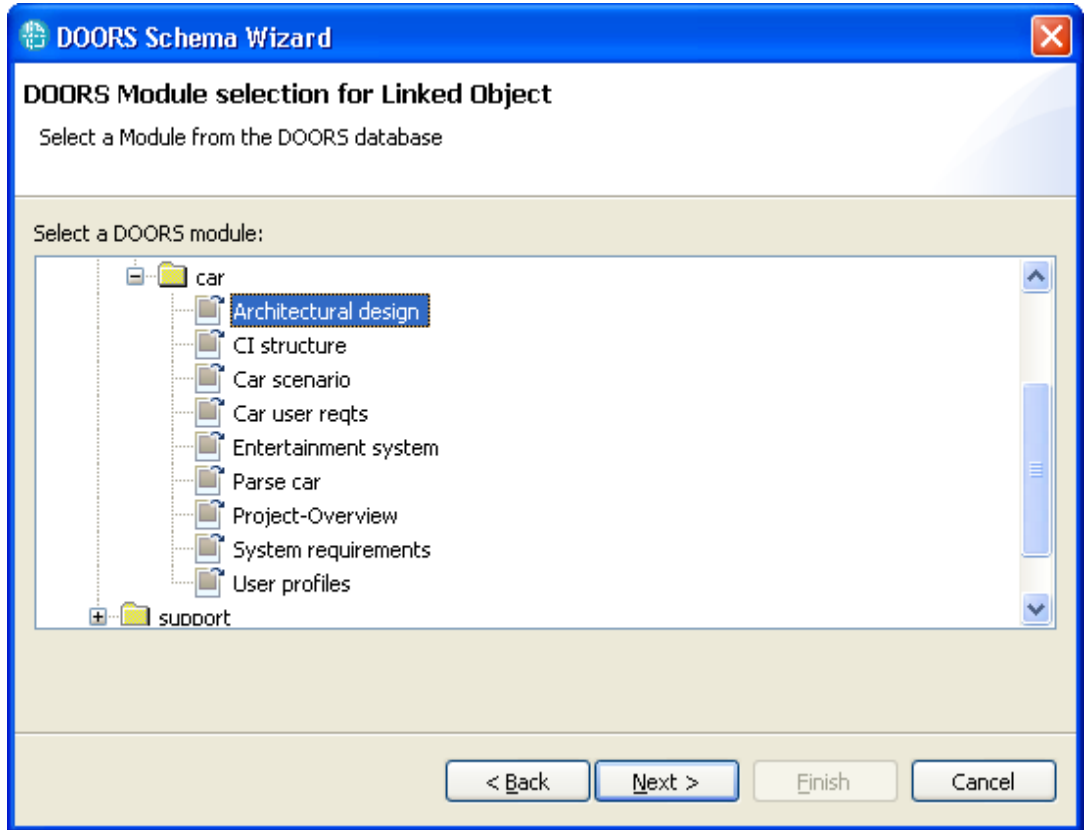


Figure 172

Then you select the module version to use.

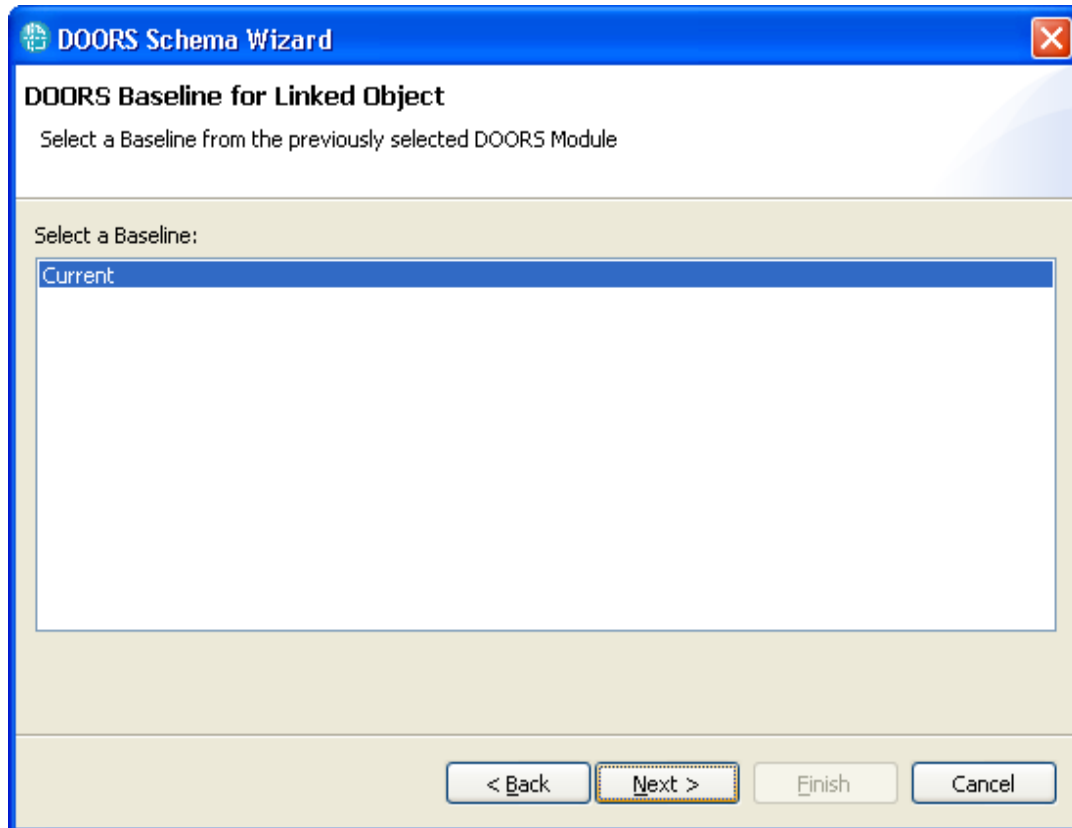


Figure 173

And last you select the attributes you want to add in the schema for the linked objects.

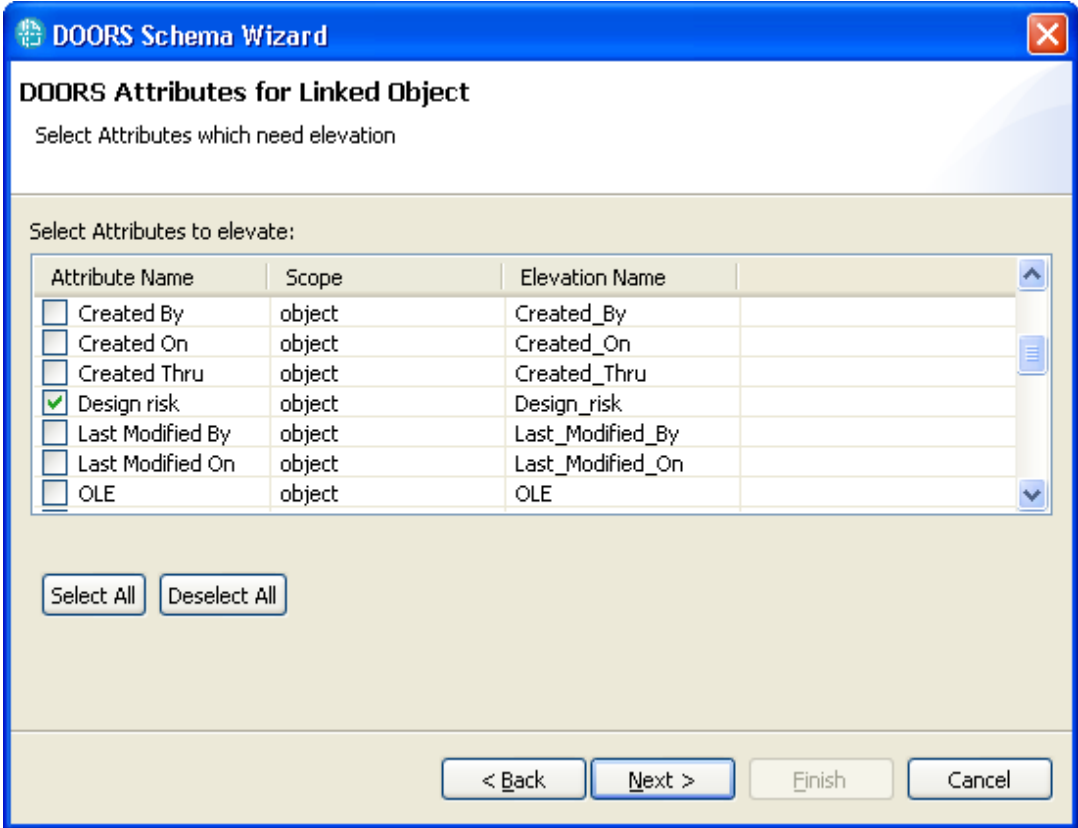


Figure 174

And finally the summary screen.

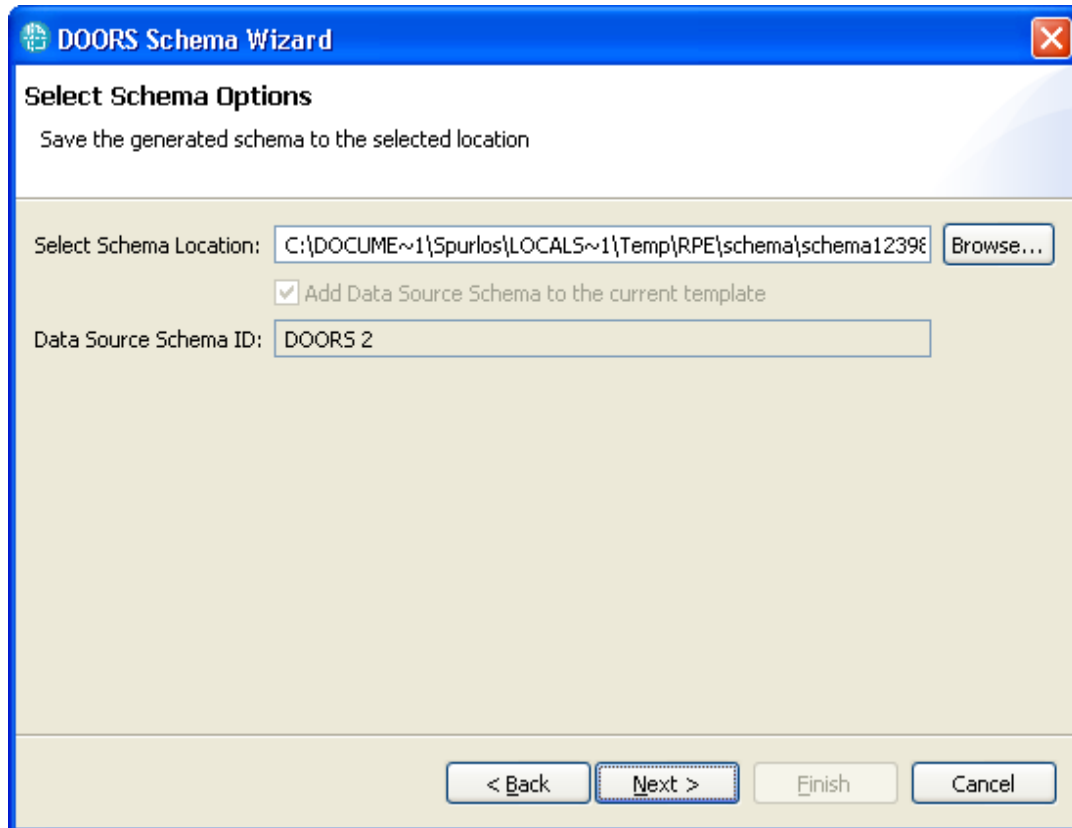


Figure 175

Once the wizard is finalized the schema is loaded in the schema view. You can see the attributes and columns added to the “Module.Object.Link.Linked Object” element.

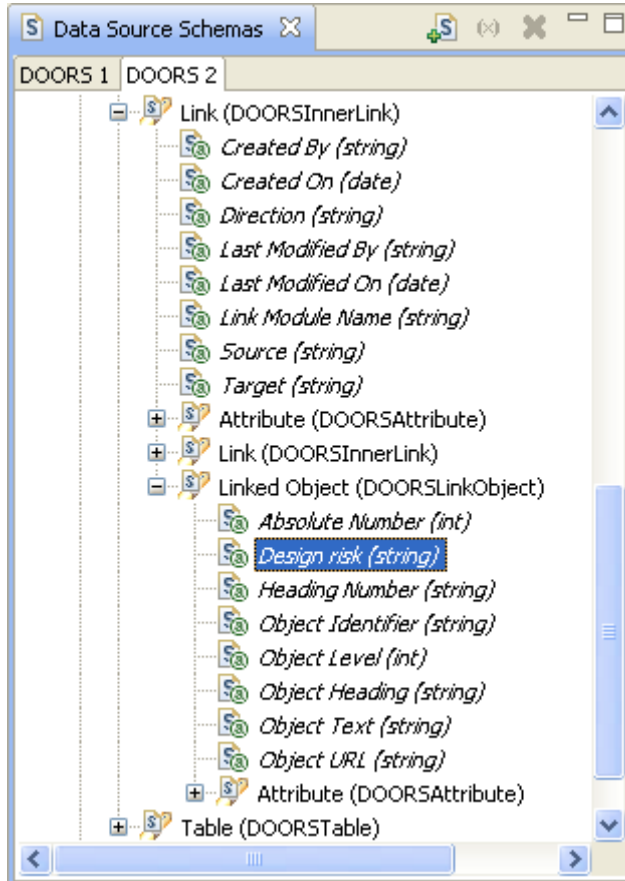


Figure 176 DOORS Schema with elevated attributes on linked objects

Tau Schema Discovery

RPE does not offer an integrated schema discovery wizard for Tau. To generate a Tau schema for your model you need to do the following actions:

Start Tau and open your model

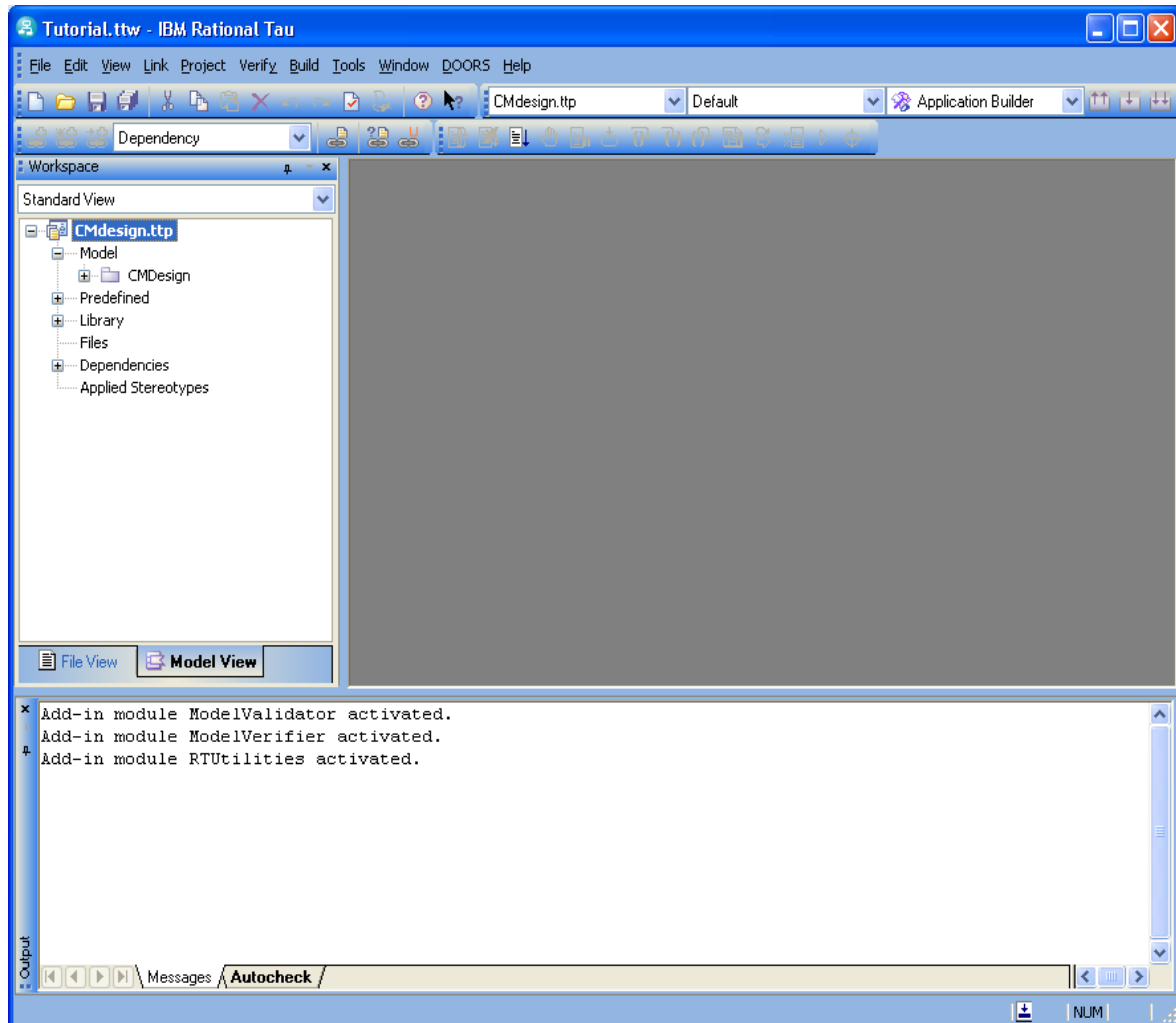


Figure 177

In the explorer select the model element that serves as your model root. For the TTMetamodel this is “Model”.

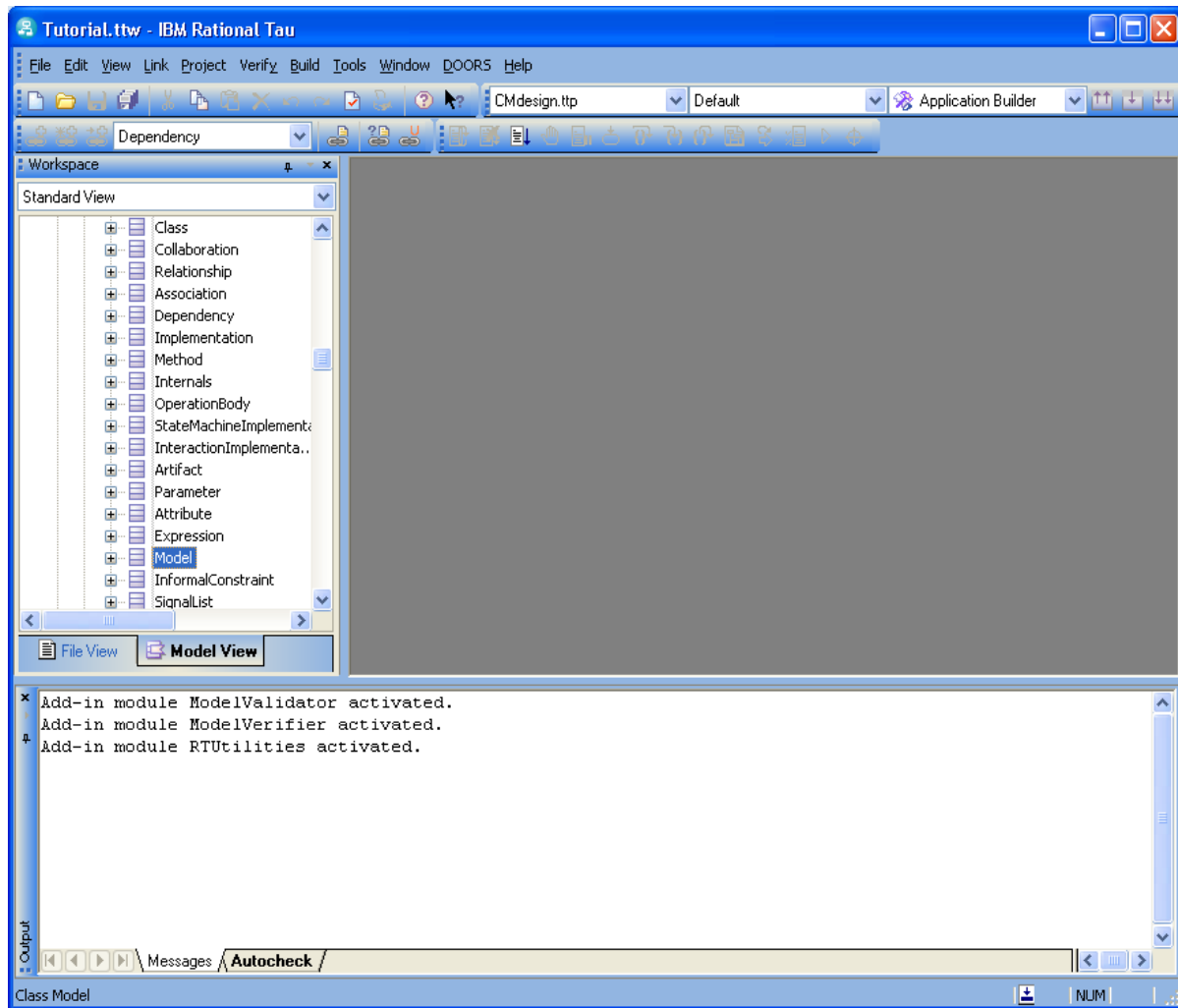


Figure 178

Load the TCL script file provided in the RPE installation.

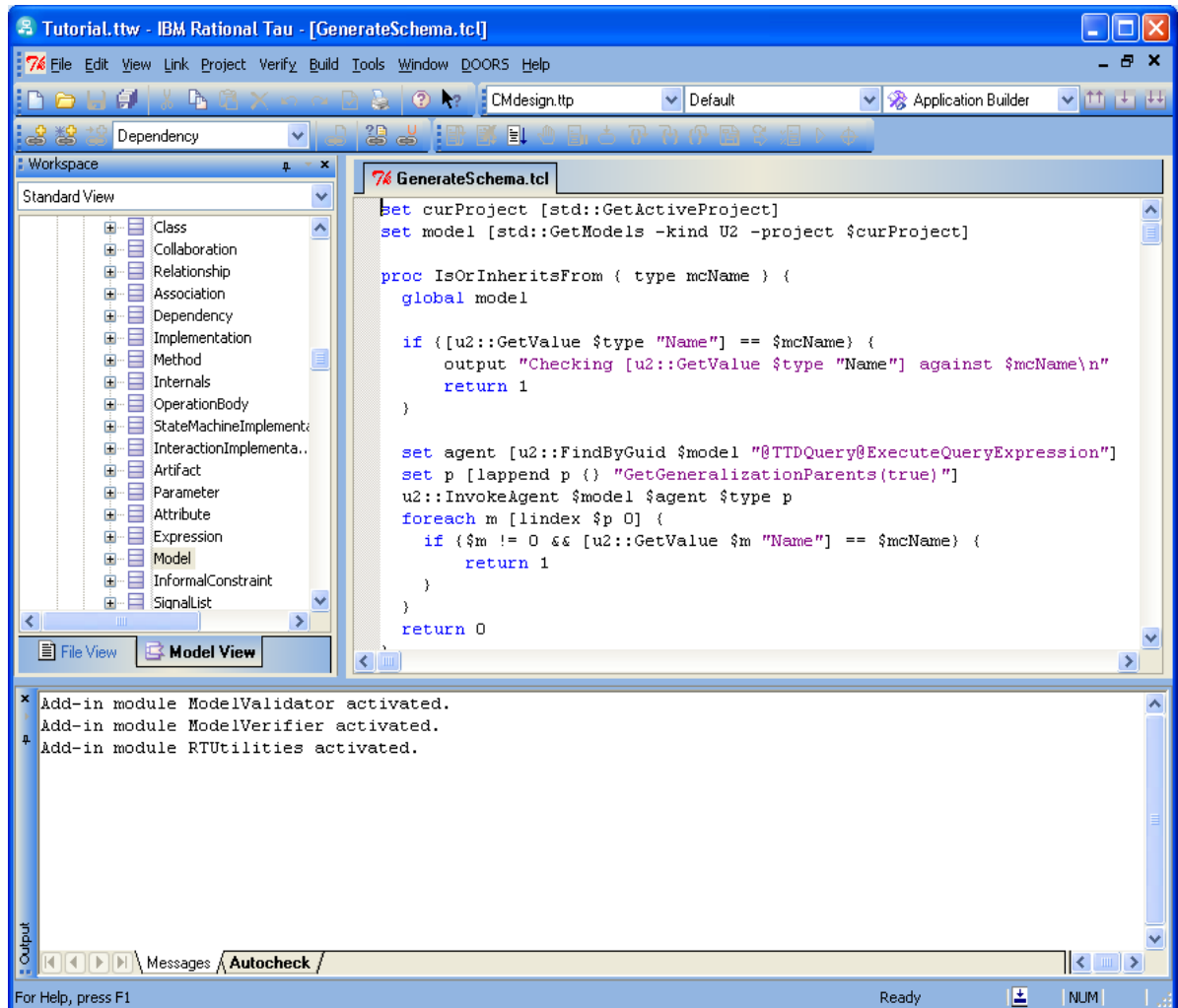


Figure 179

Run the script

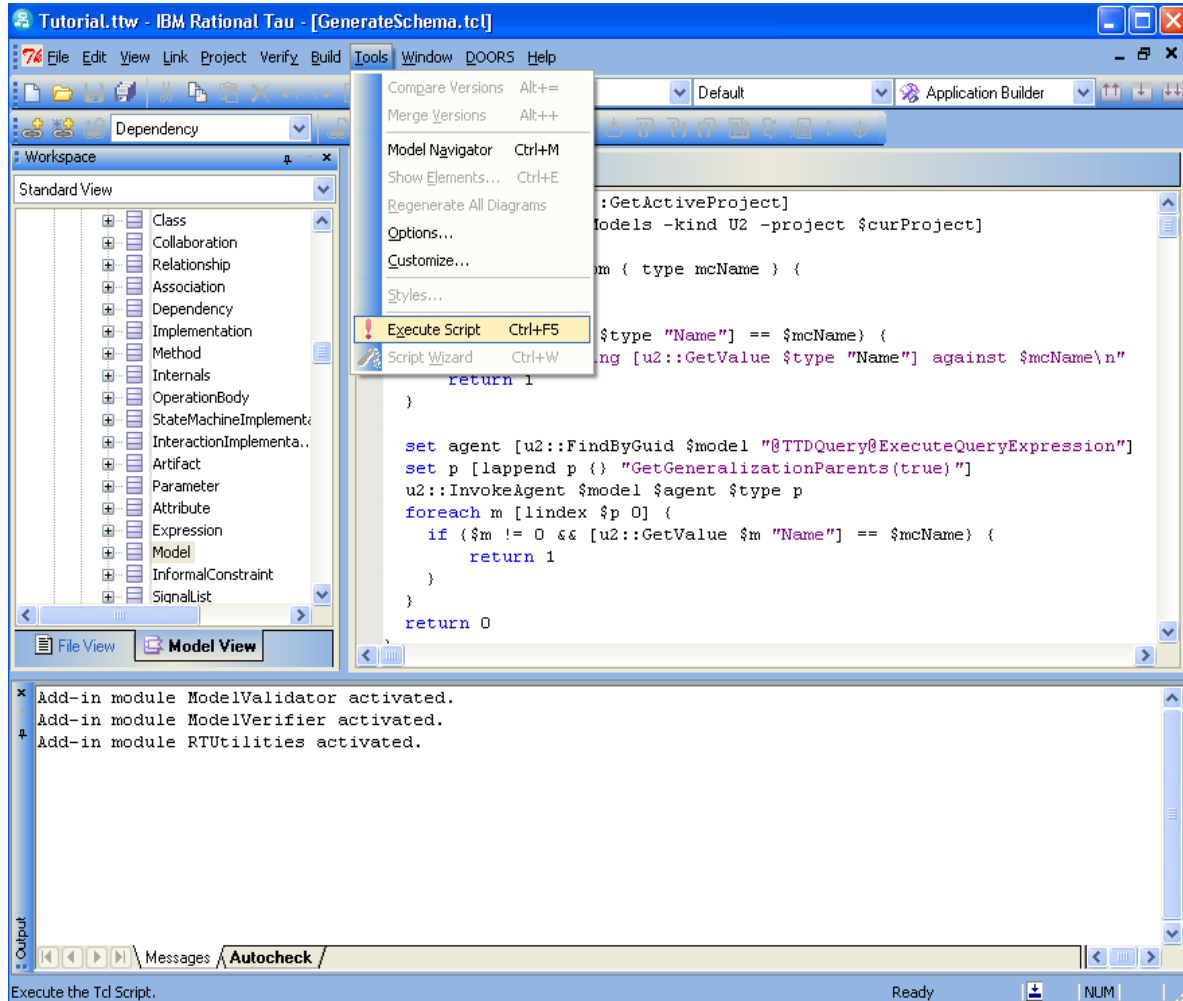
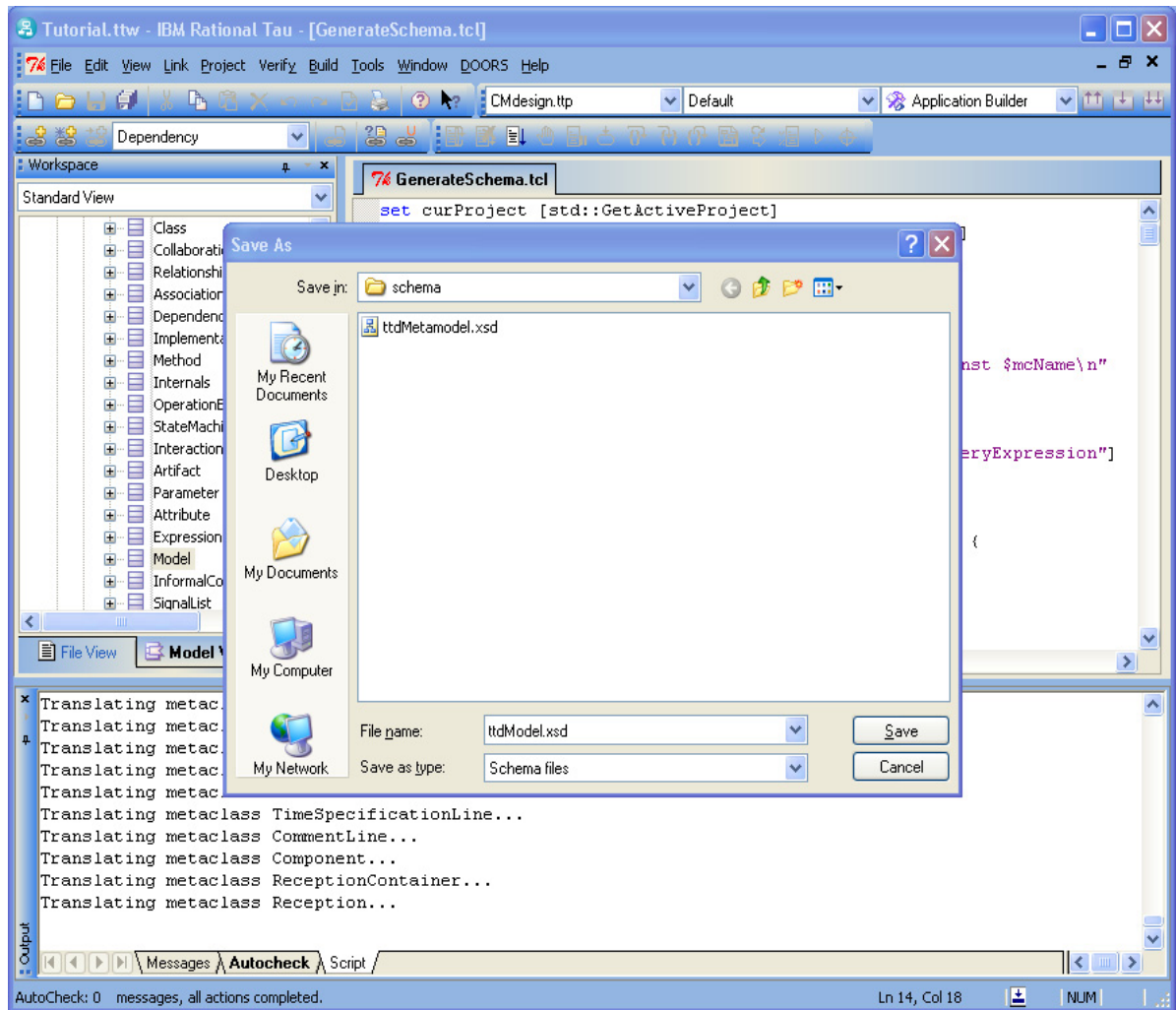


Figure 180

Once the script execution is complete you will be prompted with the location for saving the schema.



DOORS Addin

RPE integrates with DOORS to allow DOORS user to generate documents from within their familiar environment. RPE allows a DOORS user to generate documents using predefined document templates or document specifications.

The integration is available both within a module as well as from the database explorer

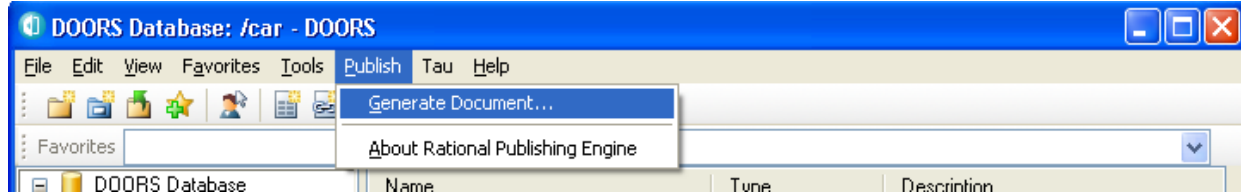


Figure 181 RPE Addin in database menu

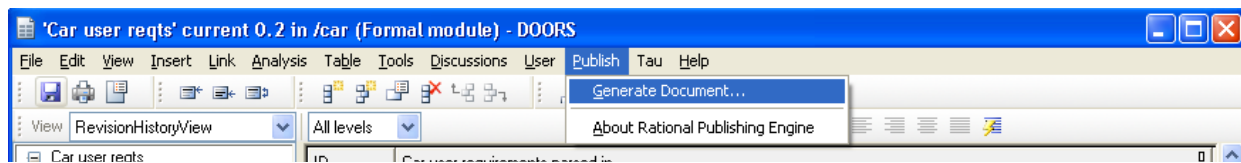


Figure 182 RPE Addin in module menu

Installation

Please consult the “Installation Guide” for details on how to install the DOORS Addin.

Usage

The Publish menu from the Addin invokes the Publish Wizard from Launcher. Please consult the “Publish Wizard” section for details.

Running from Database Menu

When the wizard is invoked from the Database menu the publish wizard will behave in the way described in the “Publish Wizard” section.

Running from Module menu

When the wizard is invoked from the Module menu the publish wizard will automatically configure all DOORS data sources from the selected template using the current module path, version and view.

Tau Addin

RPE integrates with Tau to allow Tau users to generate documents from within their familiar environment. RPE allows generating documents from predefined document templates or specifications.

Installation

Please consult the “Installation Guide” for details on how to install the Tau Addin.

Usage

Start the RPE Publish Wizard from the Publish->Generate Document menu. The Publish Wizard from RPE Launcher is started. Please consult the “Publish Wizard” section for details.

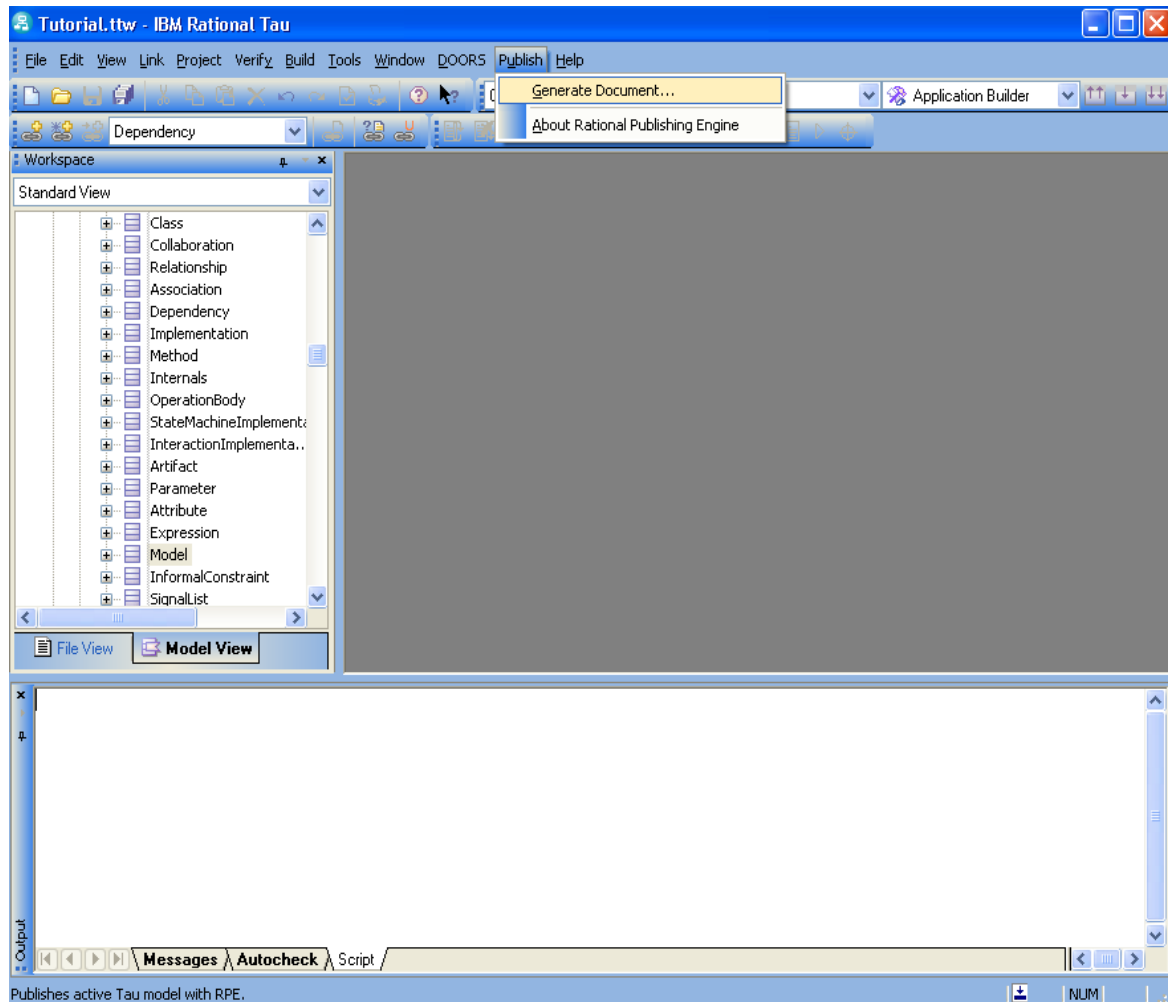


Figure 183

Deployment scenarios

Please refer the “Installation Guide” for details on deploying RPE on various platforms and modes.

Local engine

This is the most common usage scenario for RPE. In this deployment scenario RPE is fully installed on the local machine and all document generation tasks occur and are performed on the local machine as well.

Remote engine

This usage scenario is meant to support document generation on a server machine in order to minimize the resource consumption on the client machine.

How to

Tips & tricks

Editable elements

You can edit the content for: text, styled text, image, include file, bookmark and hyperlink. Double clicking any of the mentioned elements will open the content editor dialog.

Table of Contents in Word documents

If you want your output to contain a table of contents you can define a Table of Contents in your template or you can define the Table of Contents in the stylesheet.

If you define the Table of Contents (or Table of Figures, Table of Tables) in the template, the table will not show in Microsoft Word until you update the document fields. This can be done using the “Update Fields”/“Update Table” feature in Microsoft Word or using the macros provided by RPE.

Figure captions in Word documents

Just as the table of contents, the figure and table captions are not automatically updated. The resolution applied for Table of Contents can be applied here.

Included files

RPE handles Include File elements depending on the output format. For PDF, HTML and XSL-FO the included file is presented as a hyperlink while for Microsoft Word output an “INCLUDE TEXT” field is generated. This means that RPE will delegate the task of importing the file to Microsoft Word.

An included file is not visible in the output document until all fields are updated. The resolution applied for Table of Contents can be applied here.

NOTE A Word document linking other files is not self contained. Moving the document on other machines will prevent you from visualizing the content of the linked documents. If this is needed you have to include the content of the linked files using the “Break links” feature (Alt+E+K in Office 2007) from Microsoft Word or the “includeLinkedFiles” macro provided with RPE.

Heading styles

If you want to use the predefined heading styles for Microsoft Word (Heading 1, Heading 2 etc) and HTML (H1, H2 etc) the style name to use in RPE is 1,2, ..., 9. While there is no native concept of heading style for PDF and XSL-FO, RPE will use internally defined heading styles.

Formatting properties vs. Styles

You should favor defining styles in RPE instead of changing individual formatting properties for template elements.

RPE Styles vs. External Styles

While RPE offers a great deal of flexibility in defining formatting, if your main output is Microsoft Word or HTML, you can and should use external styles (defined in stylesheet) as much as possible. This approach allows changing the appearance of the output document on the fly and enforces a uniform look company wide.

Numbering headings for Microsoft Word

For headings to be numbered as a hierarchical list the easiest way is to use a stylesheet that has the headings numbered.

Dynamic images

RPE can include images whose path is not known at design time. To achieve this you need to:

- Add an image element to the template
- Double click the image to edit its content
- Define the path as a static, data or scripted expression

Dynamic included files

RPE can include files whose path is not known at design time. To achieve this you need to:

- Add an include file element to the template
- Double click the element to edit its content
- Define the path as a static, data or scripted expression

Unicode data in output

All Unicode data will be rendered as long as the font used supports it.

For PDF additional configuration is necessary (see the PDF output section for details).

For Word output, if you used a non-Unicode font you can change that font in the output document after the document generation.

For HTML output, if you used a non-Unicode font you can change the font family in the stylesheet after the document generation.

For PDF output, if you used a non-Unicode font you need to generate the document again using true type Unicode fonts and setting the appropriate output properties (see the PDF output section for details)

Moving an element outside the visible editor region

RPE Document Studio is an Eclipse RCP application. One trait of Eclipse applications is how auto scrolling is handled. If you are trying to move an element into a template location not visible without scrolling, you need to do the following:

- Select the element
- Drag it near the top/lower margin of the editor so that a part of the figure goes beyond the margin, but not the whole figure
- Pause for a second or two. Scrolling should begin at this time

Table fit to window

When auto fit to window is specified for a table, RPE will resize the table's columns using the following algorithm:

- Columns (cells) with a specified width are not changed
- The highest index column with no width specified is resized to occupy all the remaining space between the table's margins and the document's margins

If all the columns have their width specified, no changes are made to the table's width.

NOTE This behavior applies only to Word output.

PDF Tables

Tables in PDF output will always occupy the full available width unless the cell's width is set. There is no equivalent for Word's "auto fit to contents".

NOTE This is a technical limitation that we look to address in future versions.

Table merge

On all output formats consecutive tables will have the aspect of a single table. However unless cell widths are explicitly set to the same size the columns of the resulting table will not have the same width for all rows.

Once per table

When tables are merged it is desirable for some rows to appear only once, a good example being the header rows. This behavior can be specified through the "once per table" property of the row element.

Formatting	
+	common
-	specific
	row index
	row break across pages
	row repeat at page beginning false
	once per table false

Figure 184

NOTE The property does not function for rows in tables coming from different template elements. See below:

Table	
Row	
Cell	Cell
1	
Row	
Cell	Cell
2	
Table	
Row	
Cell	Cell
2	
Row	
Cell	Cell

Figure 185

Frequently asked questions

Q: *When Document Studio/Launcher started for the first time the menu actions do not work.*

A: All the menu commands work but they are not visible until the welcome screen is closed.

Q: *When inserting text with more than one line, only the first line is displayed*

A: This behavior is by design. RPE Document Studio is meant to allow for the template designer to focus on the structure of the document.

Q: *The forbidden mouse icon is displayed while expanding the branches in the Outline panel*

A: The outline pane can be used to add/delete/copy/paste elements just as the main editor window. Hence, if a palette element is selected the mouse cursor will show the locations where that element can be dropped in the outline pane.

Q: DOORS headings are not numbered

A: The easiest way to achieve heading numbering is to have the Heading styles numbered in the stylesheet that you use.

Q: Opening a Document Specification does not load the template in the studio. You have to open the template first, then the Document Specification. If you open the Document specification first and then the template a new document specification is created.

A: This behavior is by design. This allows testing different document specifications without changing the current document template.

Q: I would expect that the tree in the right pane is already expanded

A: We are evaluating the possibility of this being an option

Q: Why does RPE allow providing incorrect values to formatting properties?

A: RPE validates all properties values and will warn the user if the value is not correct. However RPE will not discard the value. All incorrect values are handled at generation time either by being ignored or by being replaced with valid values.

Q: If multiple DOORS Instances are available and "new_instance" is set to false, which DOORS Instance is used?

A: There is no definitive answer on this one. However, we've tested the behavior across different versions and different platforms, with up to 3 clients simultaneously. Consistently, when executing e.g. "GetObject ("", "DOORS.Application")", it was the client which had been open the longest which was returned.

Troubleshooting RPE

Please consult the "Installation Guide" for details on installing and configuring RPE on various platforms.

Common problems

Problem	Resolution
RPE won't start	<ul style="list-style-type: none"> • Java 1.6 or later needs to be installed on the machine • Check arguments in rpe-studio.ini and rpe-launcher.ini files from the studio and launcher folders and edit the following values <p>-vmargs -Xms56m -Xmx1024m</p> <p>Lowering these values should help start RPE on low resource machines or on Virtual Machines.</p> <p>-vmargs -Xms56m -Xmx512m</p>
Cannot generate documents from DOORS Data Sources	<ul style="list-style-type: none"> • A DOORS 9.1 or later client must be installed • Check that the data source is properly configured (the module path, view name and baseline are case sensitive)
Cannot generate documents from Tau Data Sources	<ul style="list-style-type: none"> • A Tau 4.2.0.1 or later installation must be available • The path to Tau bin folder must exist in the PATH system variable • Check that the data source is properly configured
Cannot run Word macros	<ul style="list-style-type: none"> • Microsoft Word needs to be installed • The macro must be defined in the stylesheet
No OLEs in the output document.	<ul style="list-style-type: none"> • OLEs are support for Word output only. In order to see get the OLEs in your Word document please follow the indications from the Output chapter.

NOTE For any problems you encounter during RPE usage we recommend sending the log files mentioned bellow to the support team.

RPE Core Logging

RPE logs all the activity to the console and to a file on the file system. The console will display human readable information while the file contains detailed information usable by RPE support. The logging behavior is controlled through the *log4j.properties* file located in %RPE_HOME%.

NOTE The system variable RPE_HOME must be defined for RPE to be able to read the logging configuration.

NOTE The default output of the log file is RPE\RPE.log in the user home folder, usually:
C:\Documents and Settings\\temp\RPE\RPE.log

RPE UI Logging

RPE Document Studio and RPE Launcher, like any other Eclipse RCP applications, log all user interface related problems in their own log files. These log files can be found in the workspace for Document Studio and Launcher. The location of the workspaces, unless changed by the user are:

C:\Documents and Settings\\Application Data\IBM\Rational\RPE_20090601\Studio\workspace

C:\Documents and Settings\\Application Data\IBM\Rational\RPE_200910601\Launcher\workspace

The log files are located inside the workspace in the .metadata folder.

NOTE These log files do not exist unless errors occurred.

RPE Core Debug Mode

RPE provides an extended logging mode for document generation. That mode can be activated from the Launcher's preference pages.

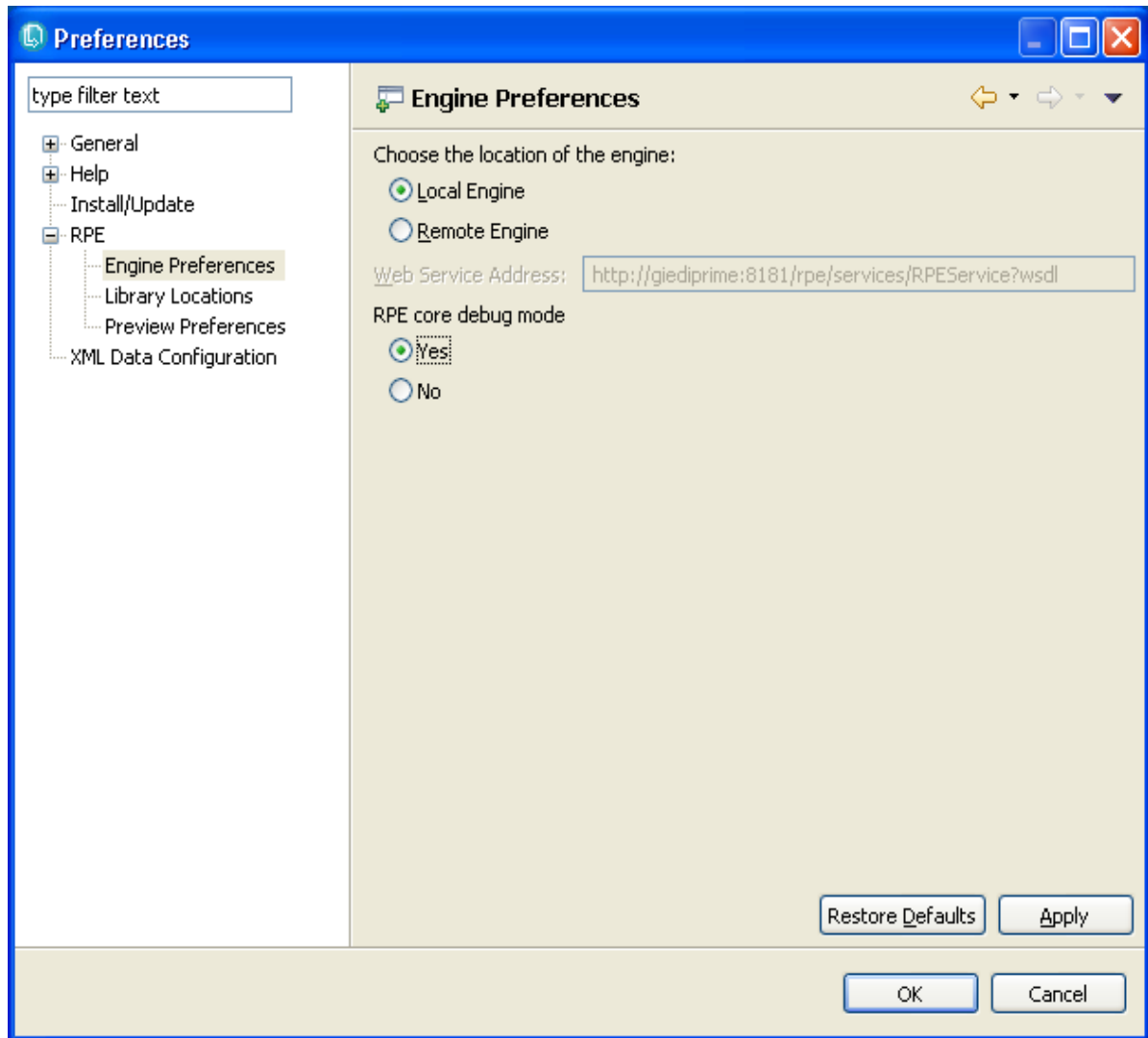


Figure 186 Engine preferences

If running in Core Debug mode, RPE will dump detailed information in the log files. Information about the current element, conditions checked and current data source context will be logged. You can use that file yourself or send it to support.

Customer feedback module

Customer Feedback Module is a tool that helps customer to report defects to IBM Rational Support. In order to minimize the errors in reporting product specific and system data, the CFM

tool gathers product specific and system data automatically. It can send text, log files, system information, screenshots and video capture.

It can be started by invoking "Generate Support Request..." under the "Help" menu.

By clicking the "Add Product Files" button, RPE-specific log files (if available) are appended, for sending them to Support.

For more information regarding the CFM usage, refer to it's specific documentation available by clicking the "Help" button on the CFM main dialog.

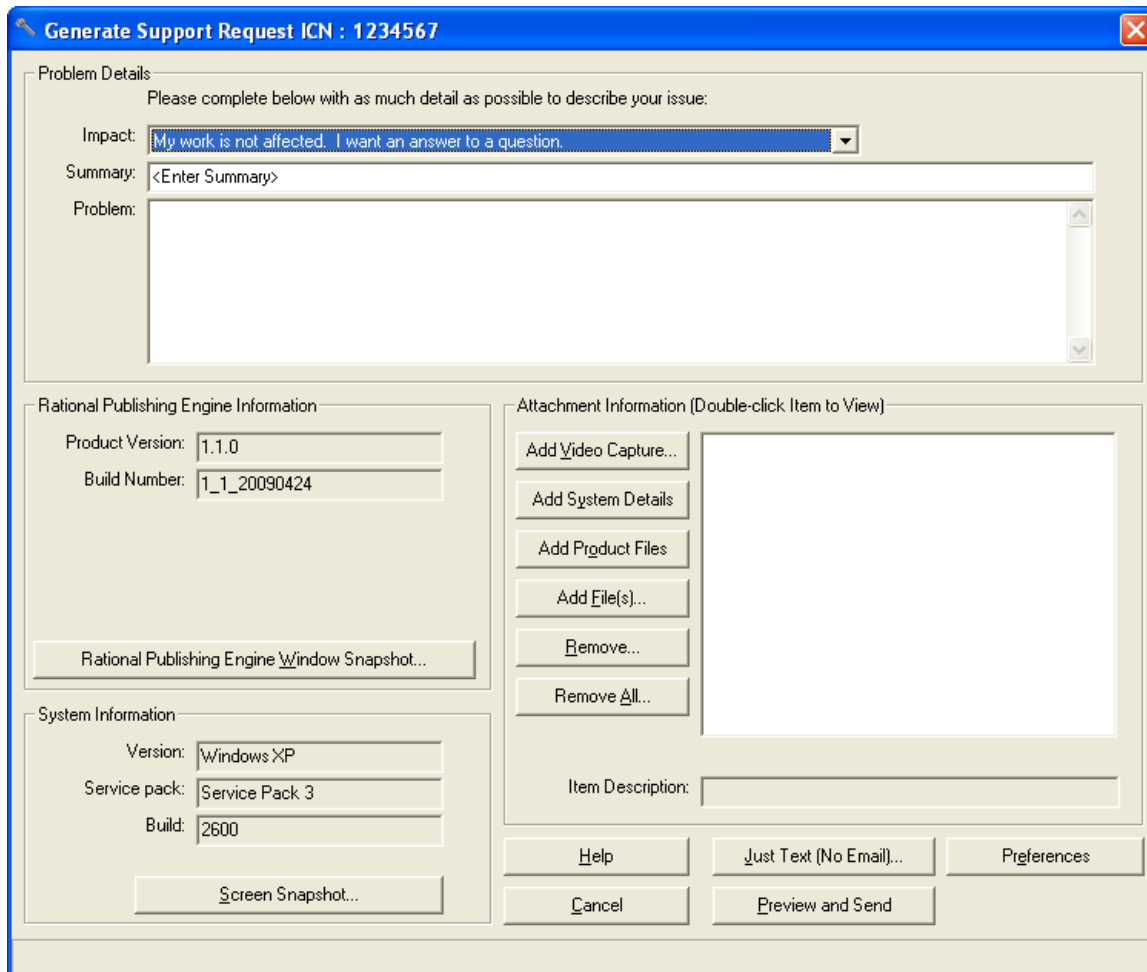


Figure 187 CFM

Examples

A template for DOORS data

The example is available in the RPE installation in the DOORS Examples folder as "Tutorial_DOORS.dta".

Basic setup

Create a new Document Template

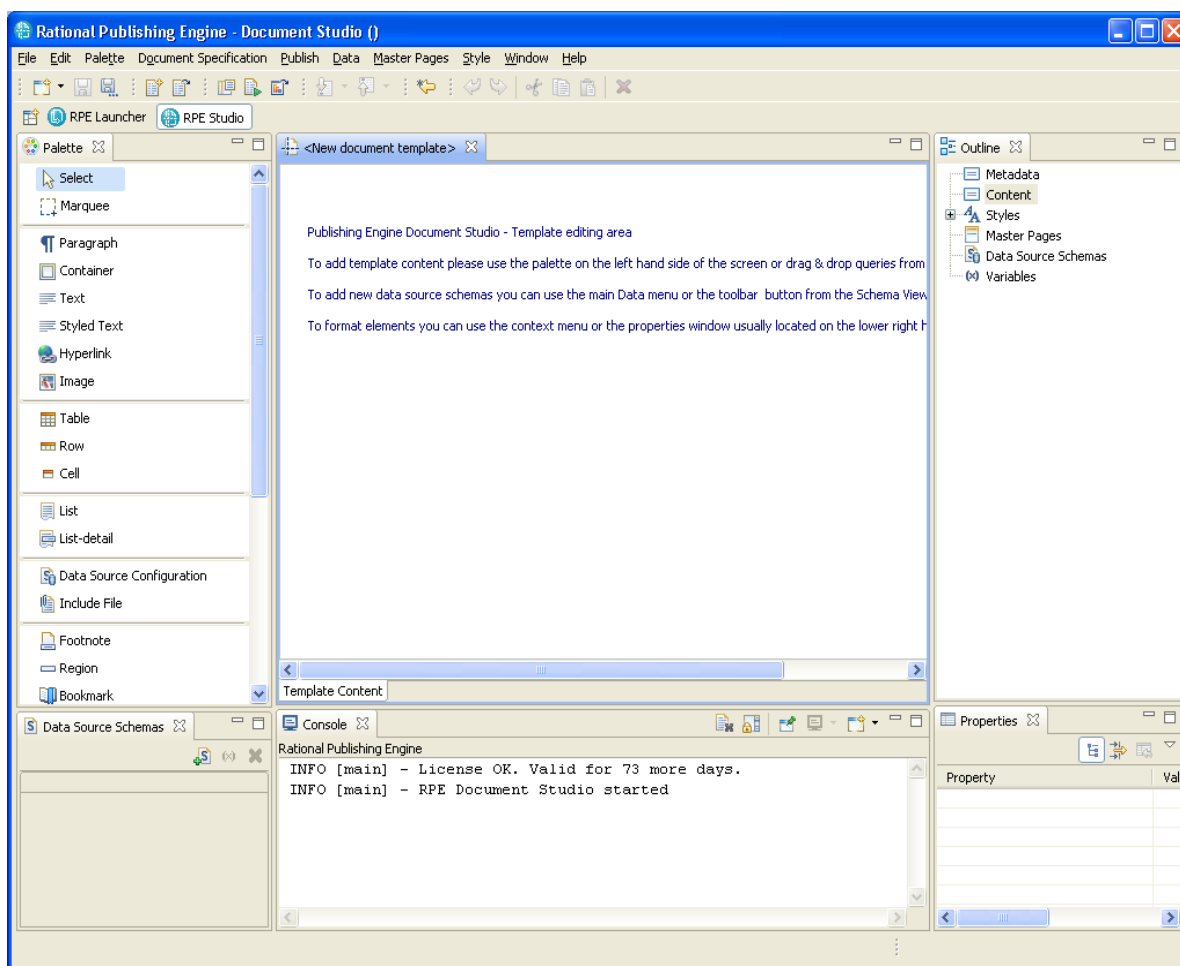


Figure 188

Add a data source schema using the Add Data Source Wizard. Change the data source type to DOORS and accept the default values provided by RPE.

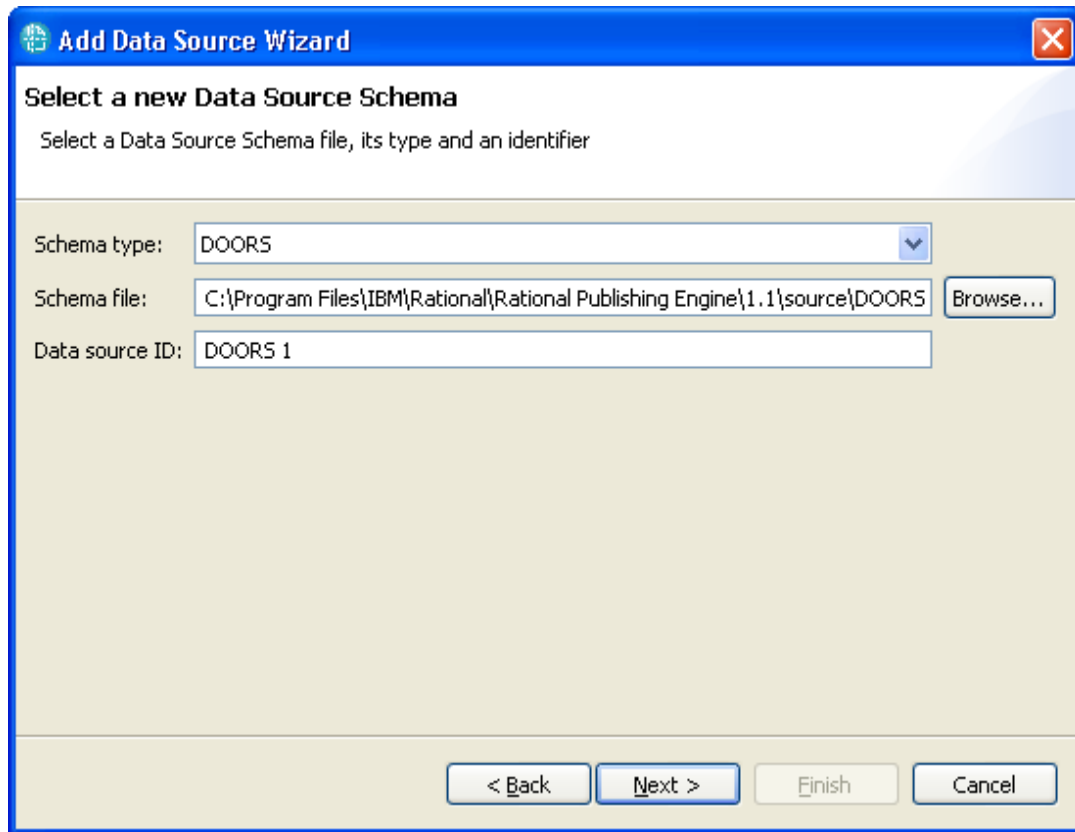


Figure 189

Review selection

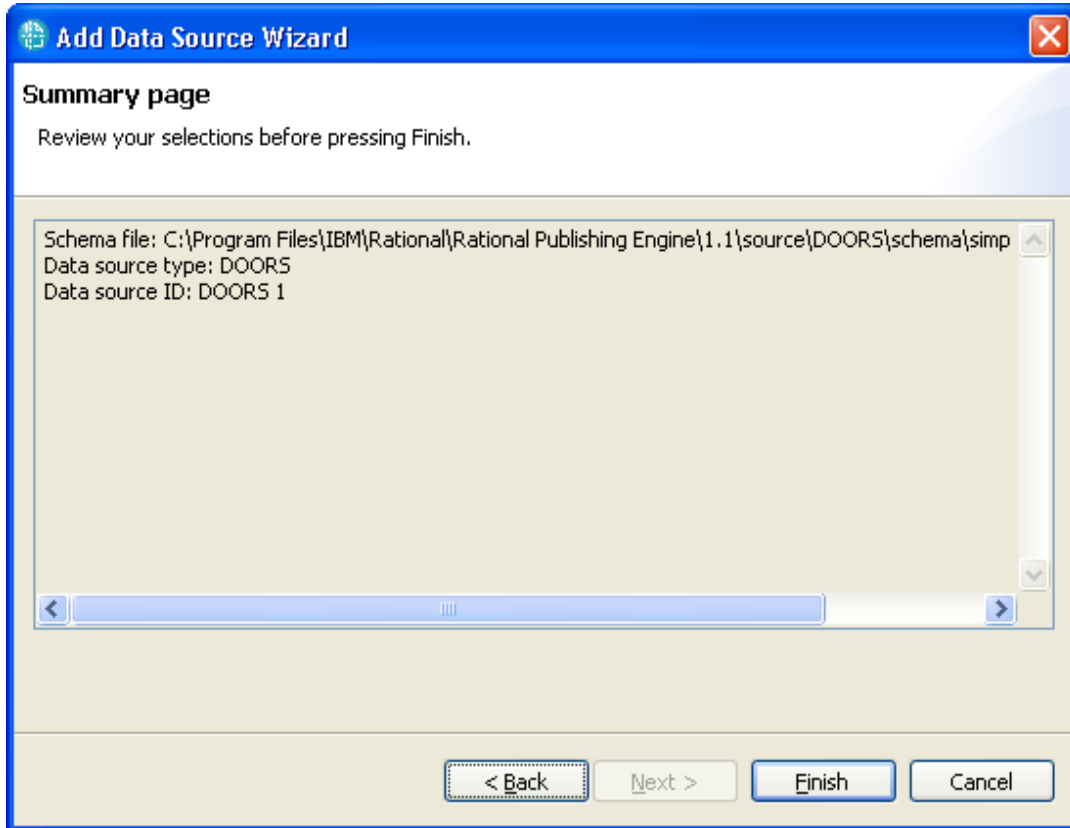


Figure 190

Add a **container** element in the template

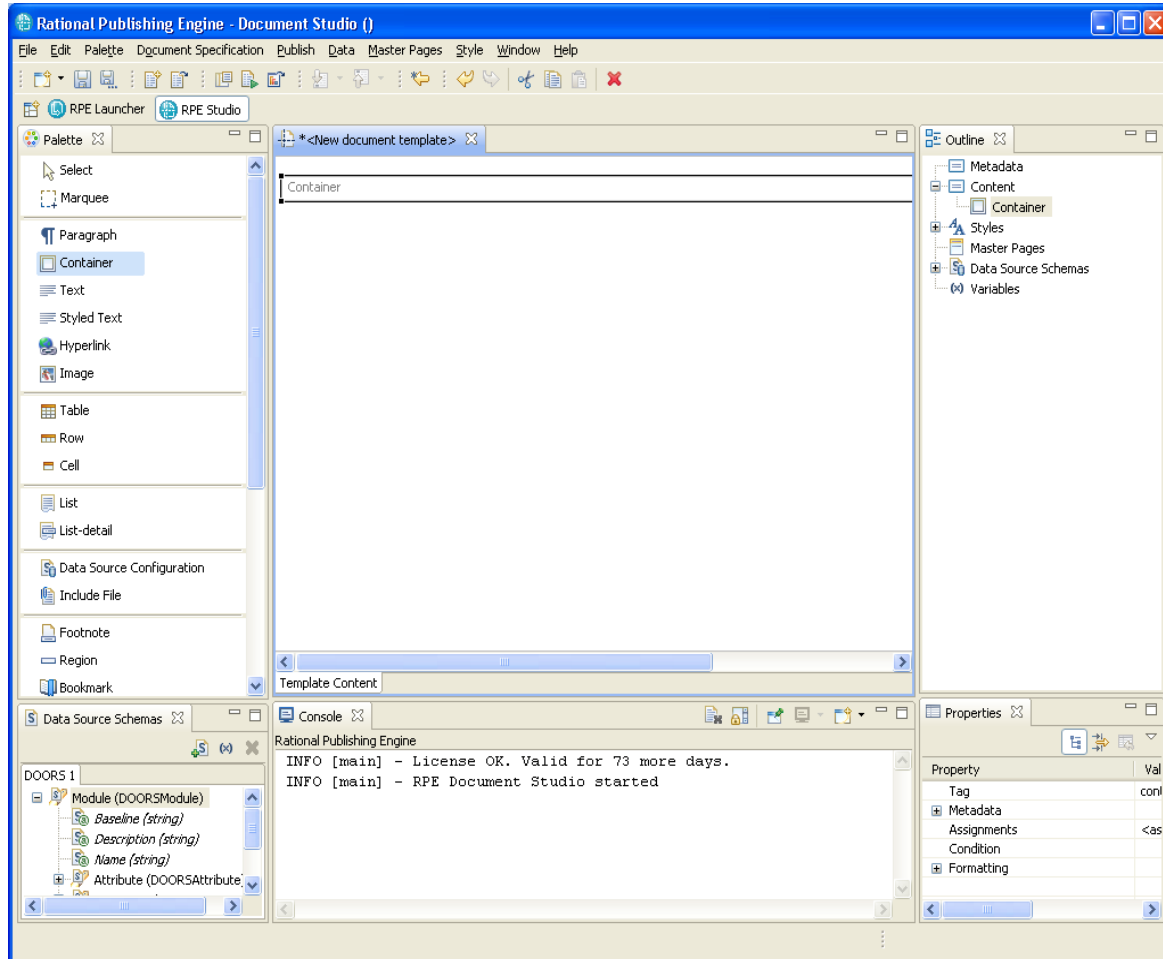


Figure 191

Assign the Module.Object query to the element.

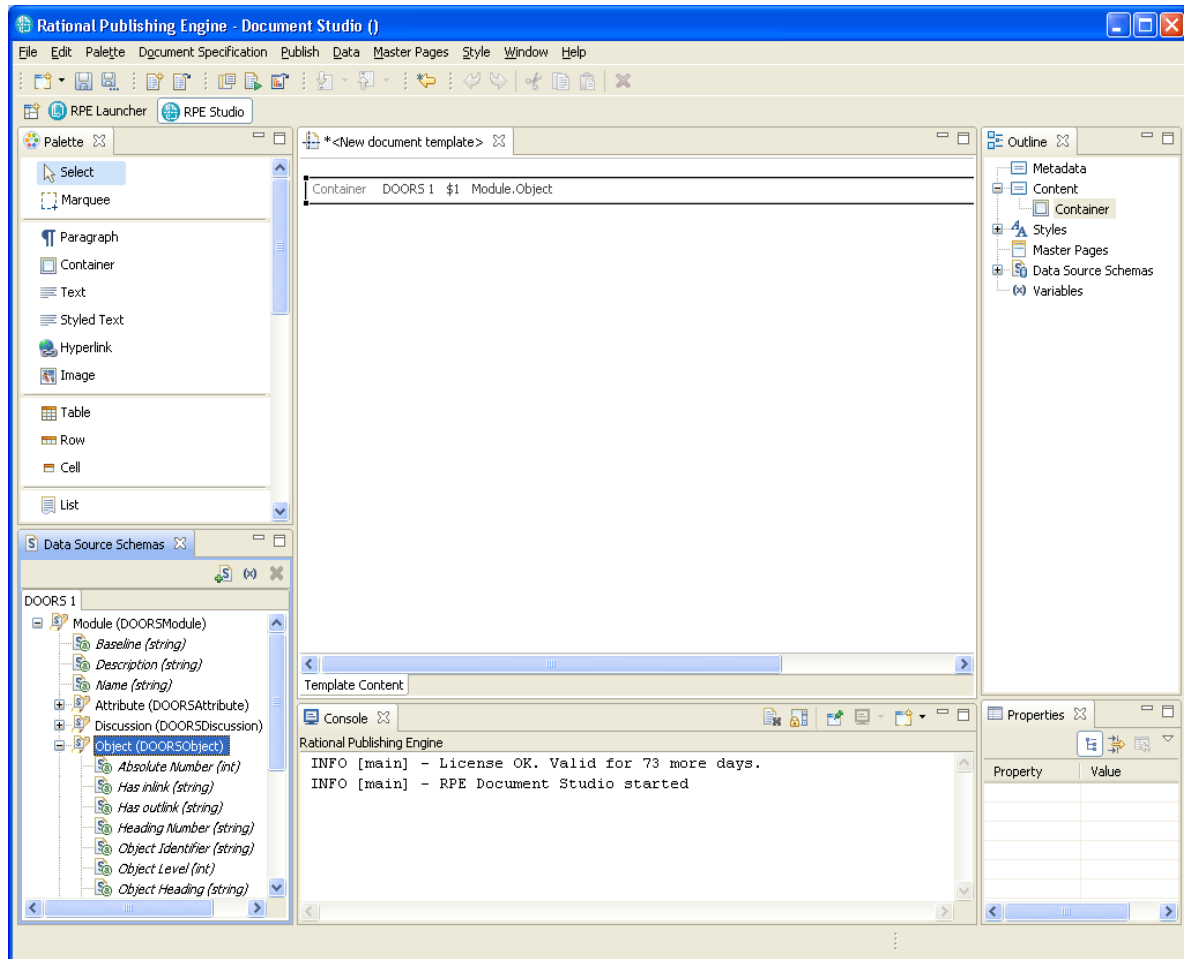


Figure 192

Add the “Object Heading” attribute in a text element in the container

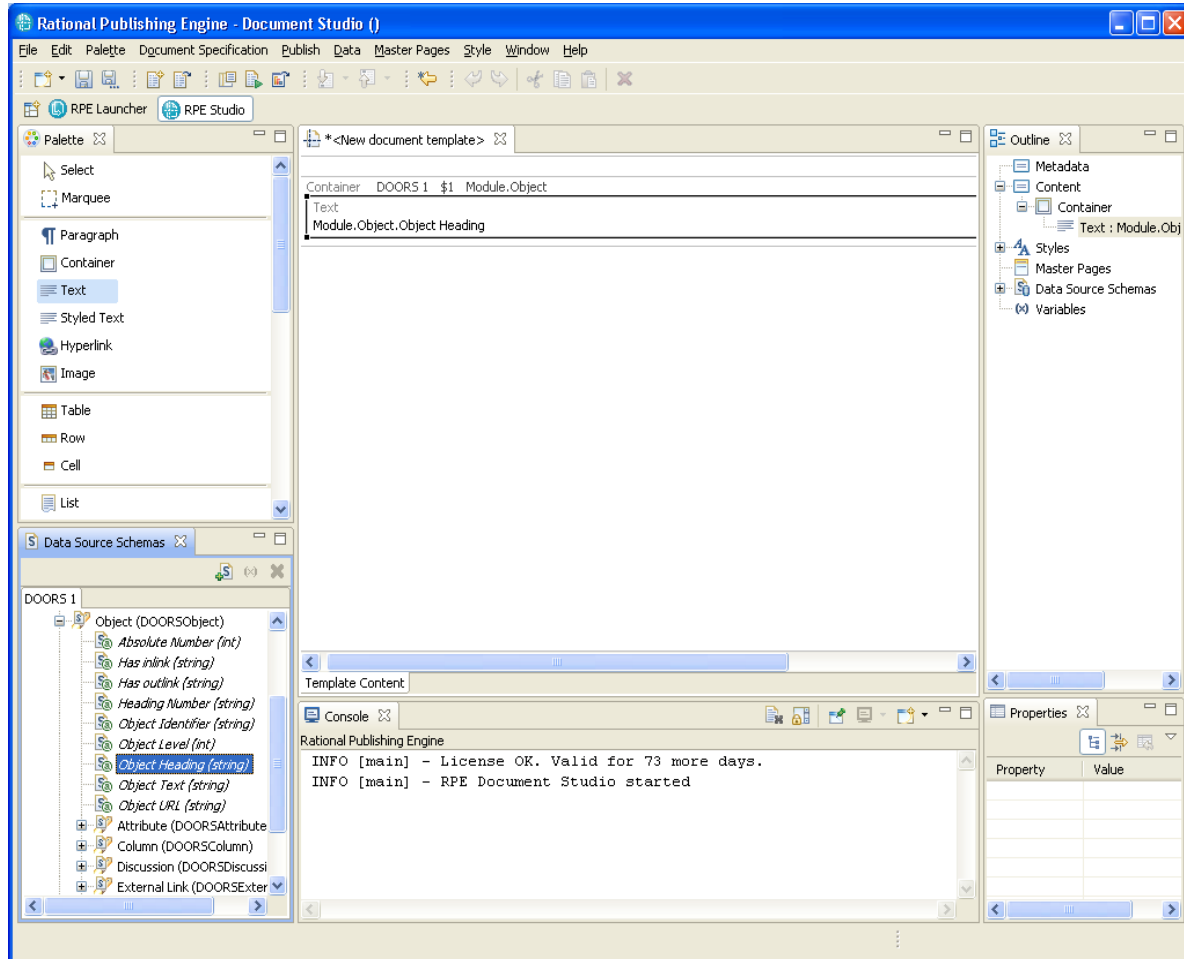


Figure 193

Apply a Condition to the display of the text element only if the Object Heading is not empty.

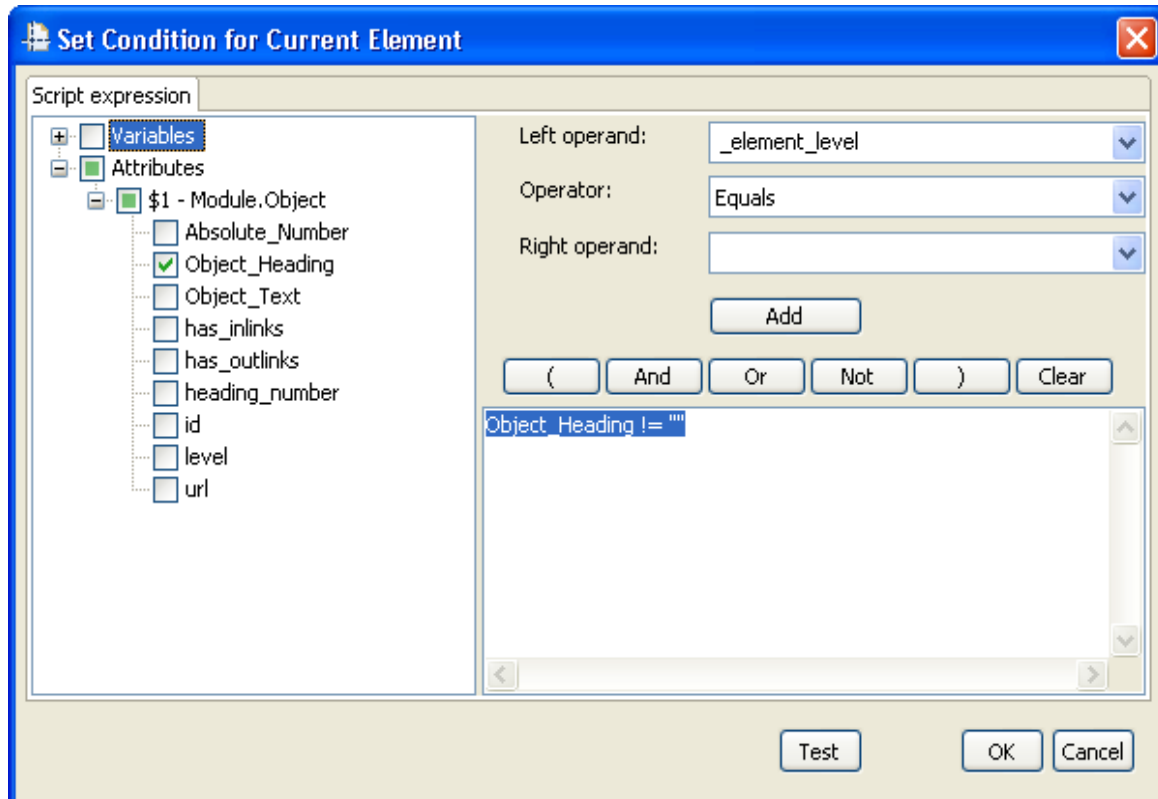


Figure 194

Add the “Object Text” attribute in its own paragraph to avoid collating multiple object texts on the same line.

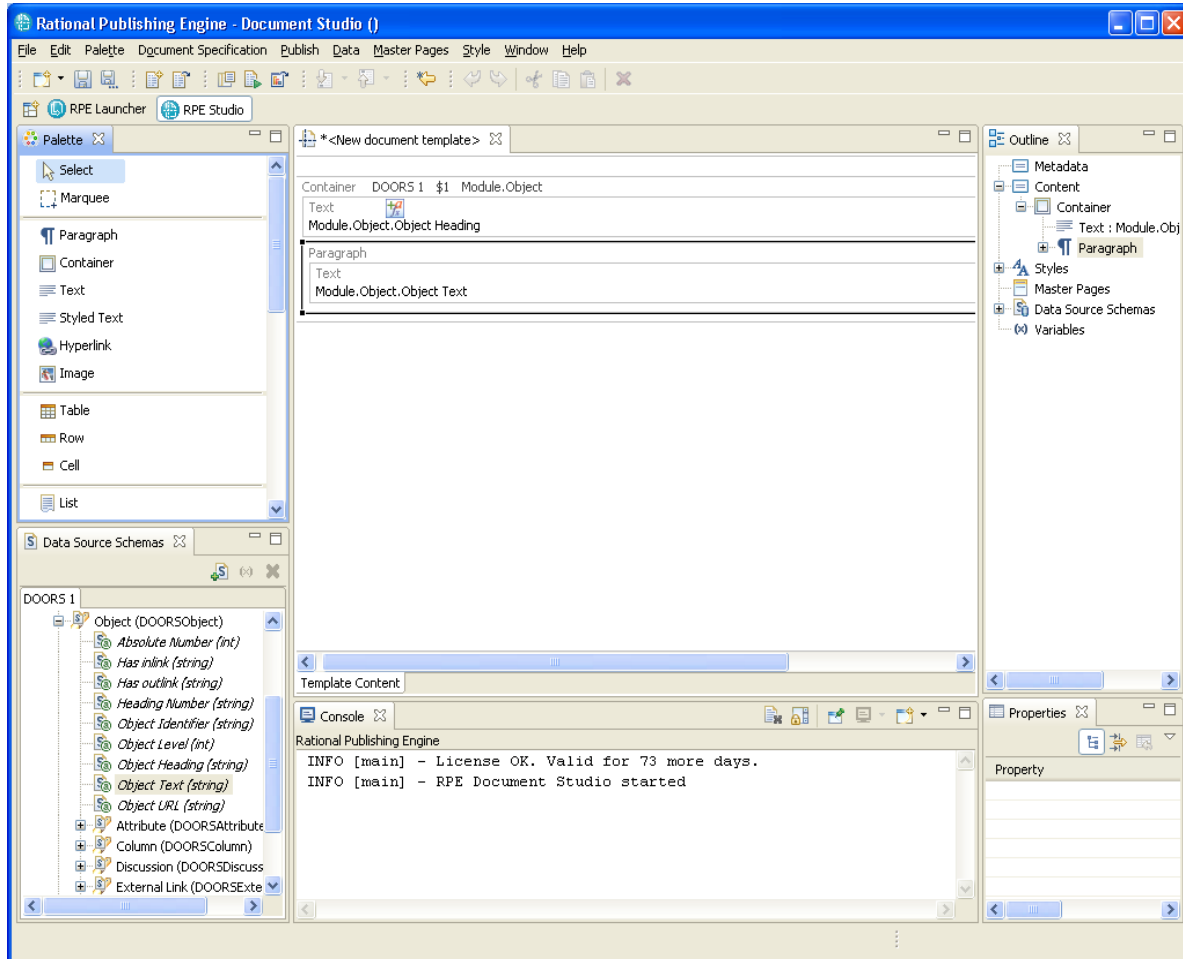


Figure 195

Condition the display of the paragraph containing the Object Text

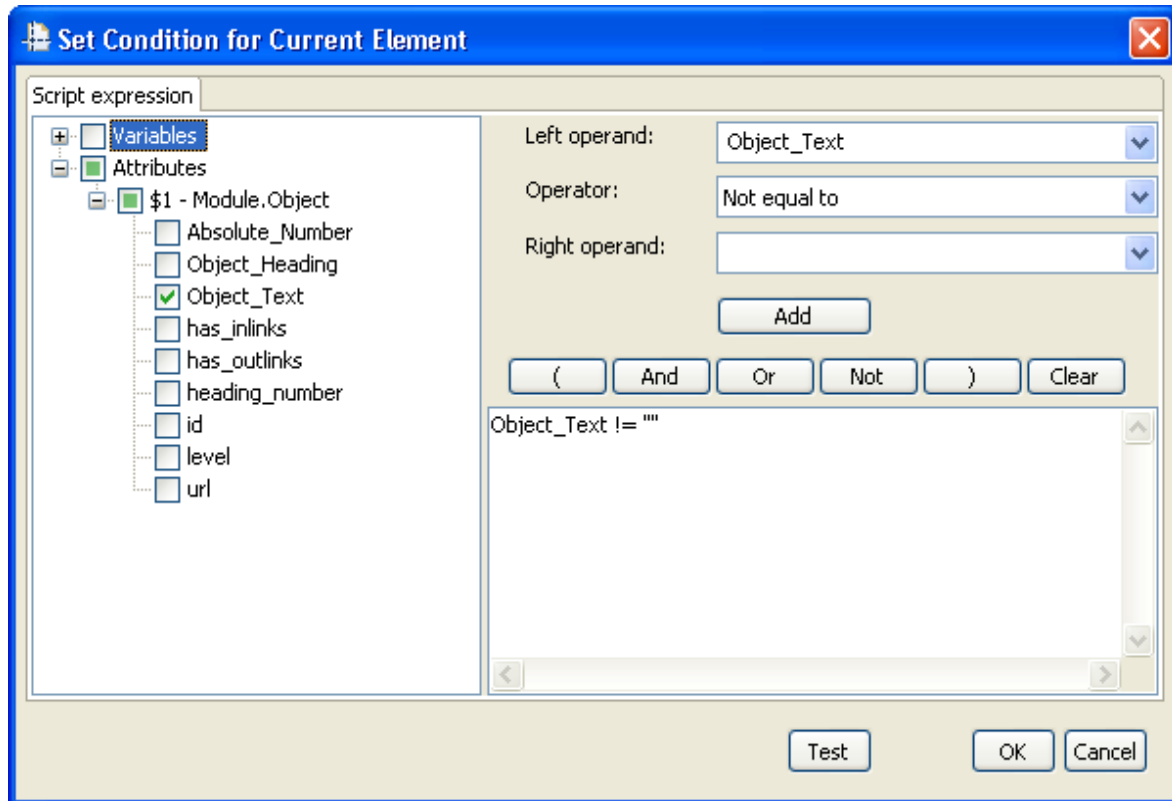


Figure 196

Save the template

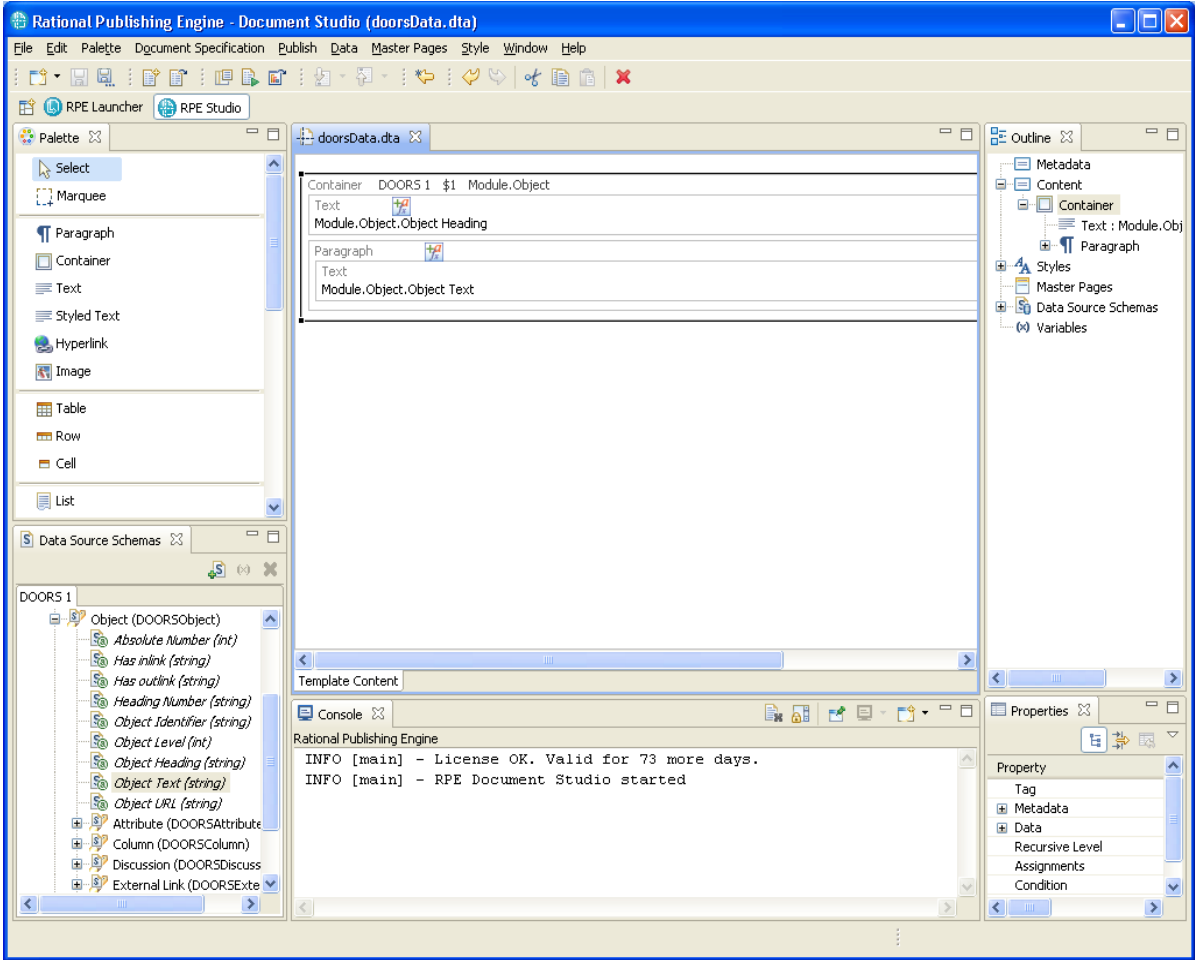


Figure 197

Advanced template options

Add a Table of contents and an image at the top of the document

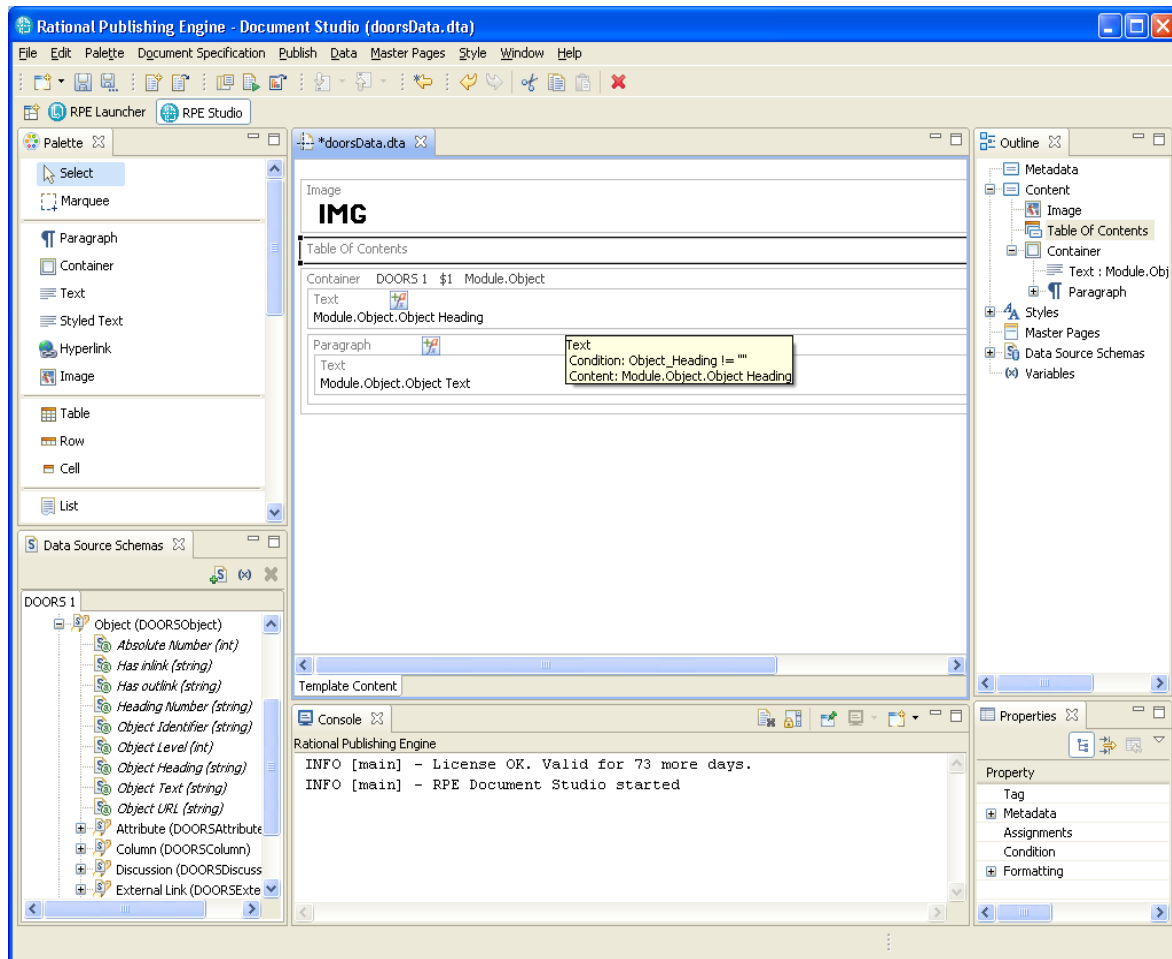


Figure 198

Load an image file from the file system

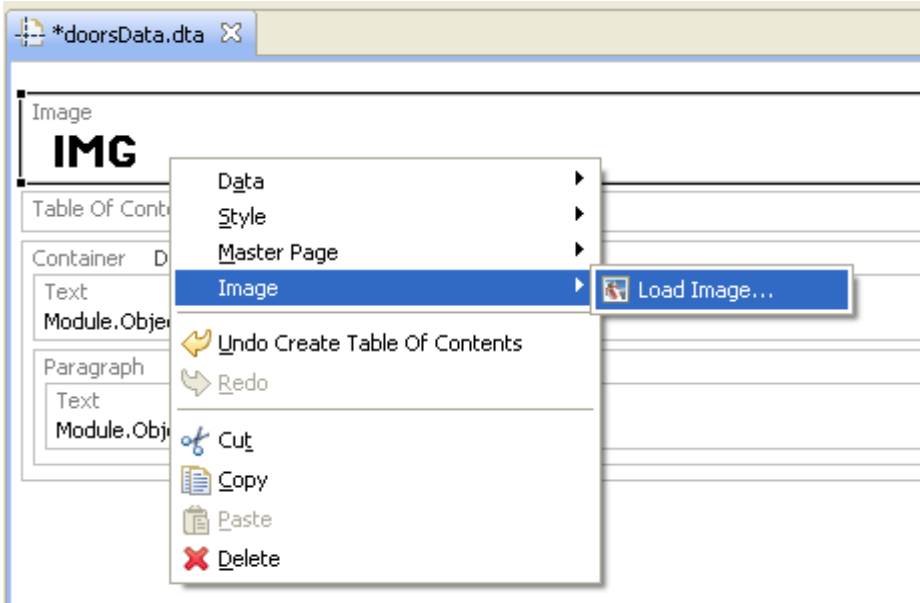


Figure 199

NOTE The selected image is copied in the template.

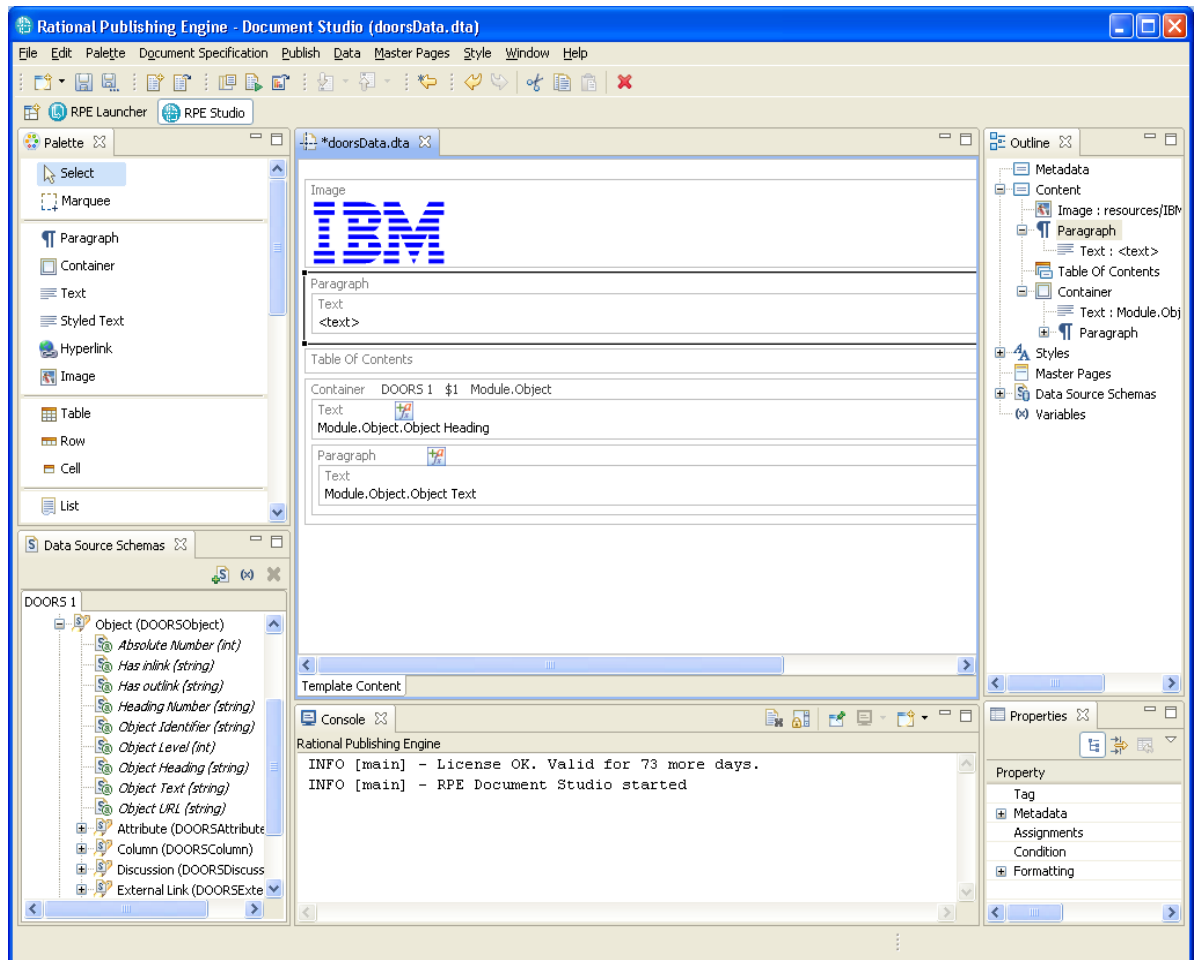


Figure 200

NOTE Save your template

Add a paragraph with a text element after the image to hold document title.

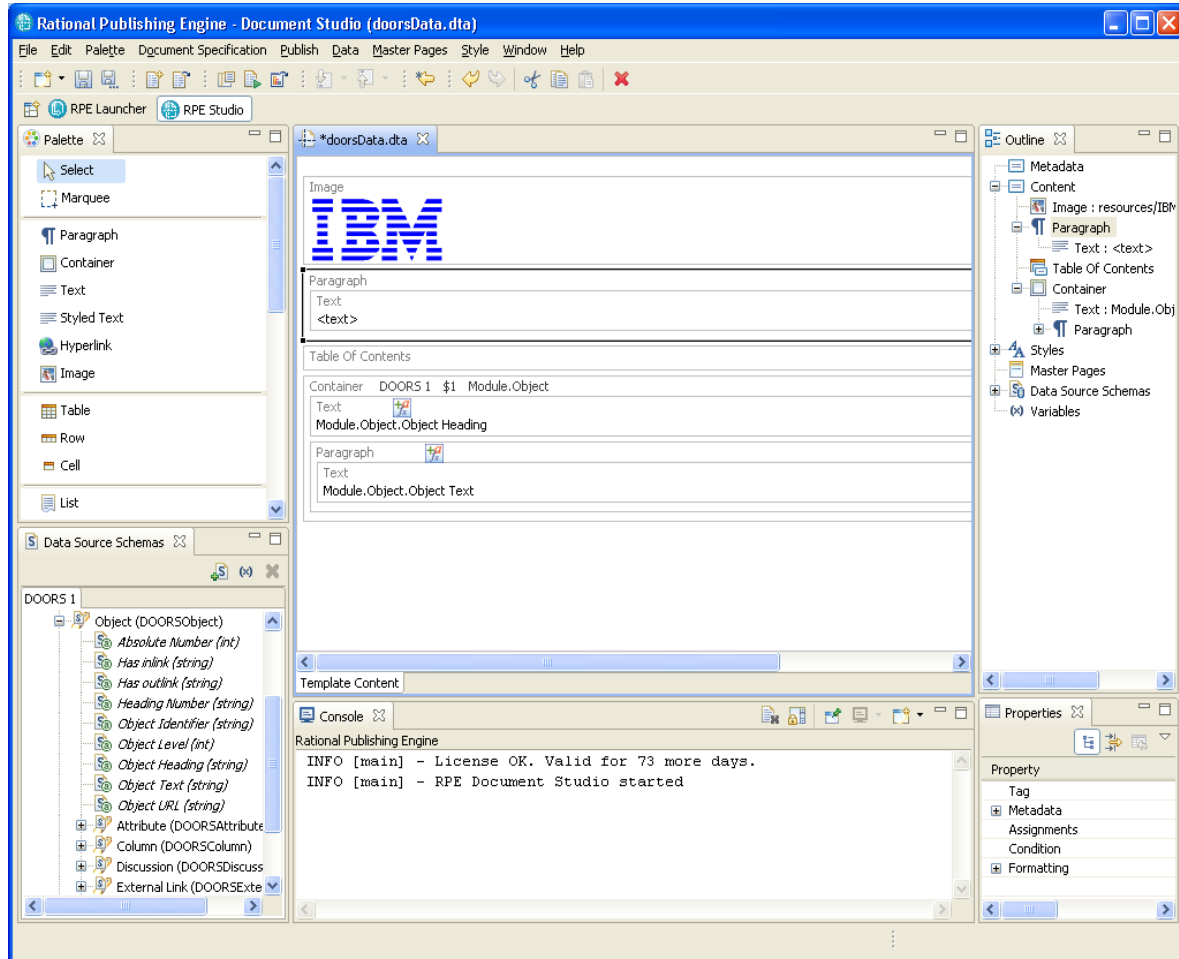


Figure 201

NOTE Save your template

You can use a symbolic name for the paragraph, such as “Document Title Placeholder”

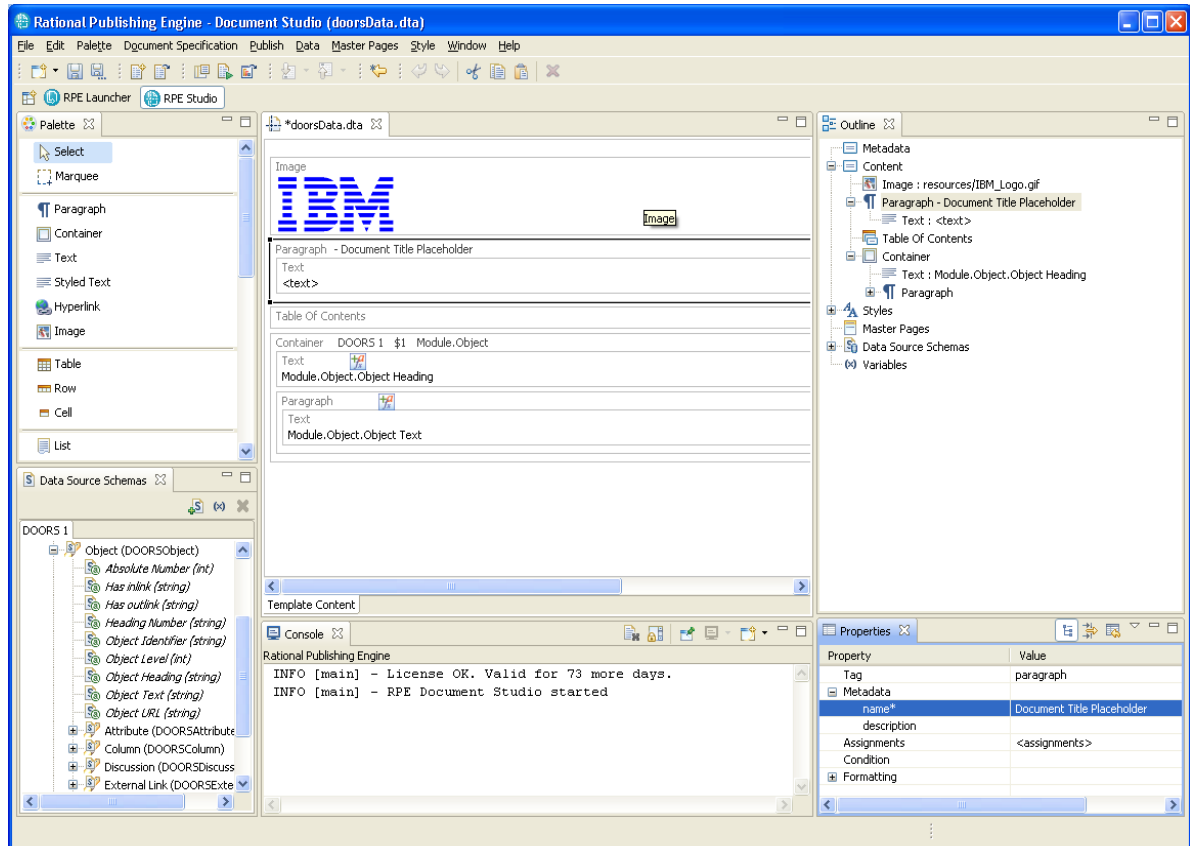


Figure 202

NOTE Save your template

Add a variable to the document and name it “DocumentTitle”. Set its default value to “DOORS Data”. Make sure you define the variable as external.

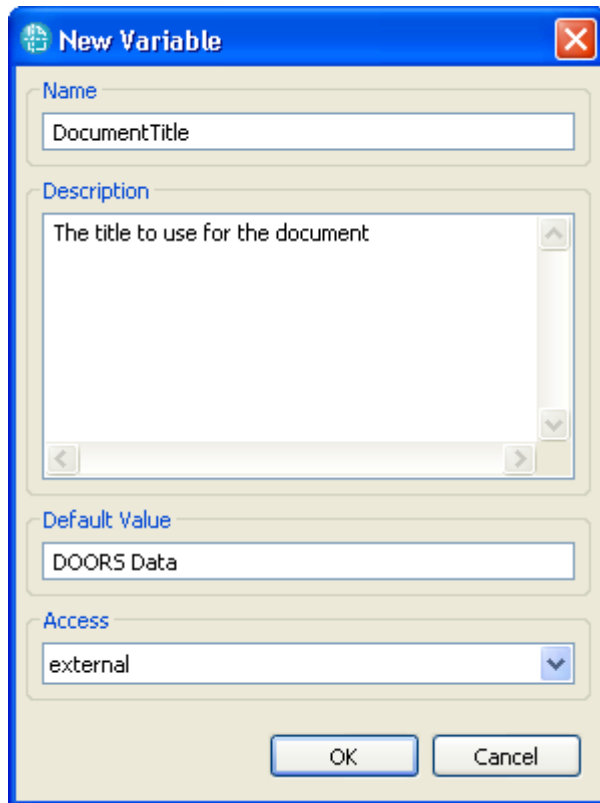


Figure 203

NOTE save your template

Drag the variable from the outline over the text in the “Document Title Placeholder” paragraph.

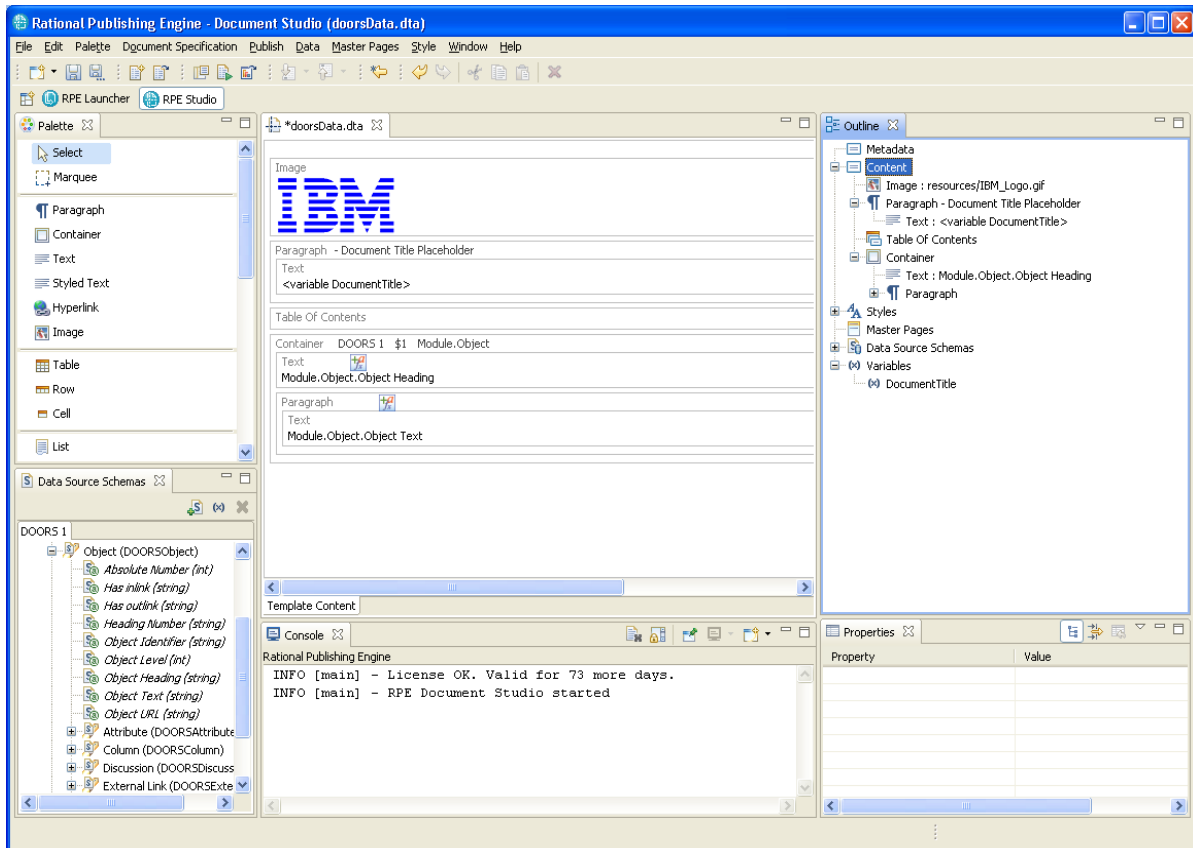


Figure 204

NOTE Save your template

Define a title style with the name "RPE_Title".

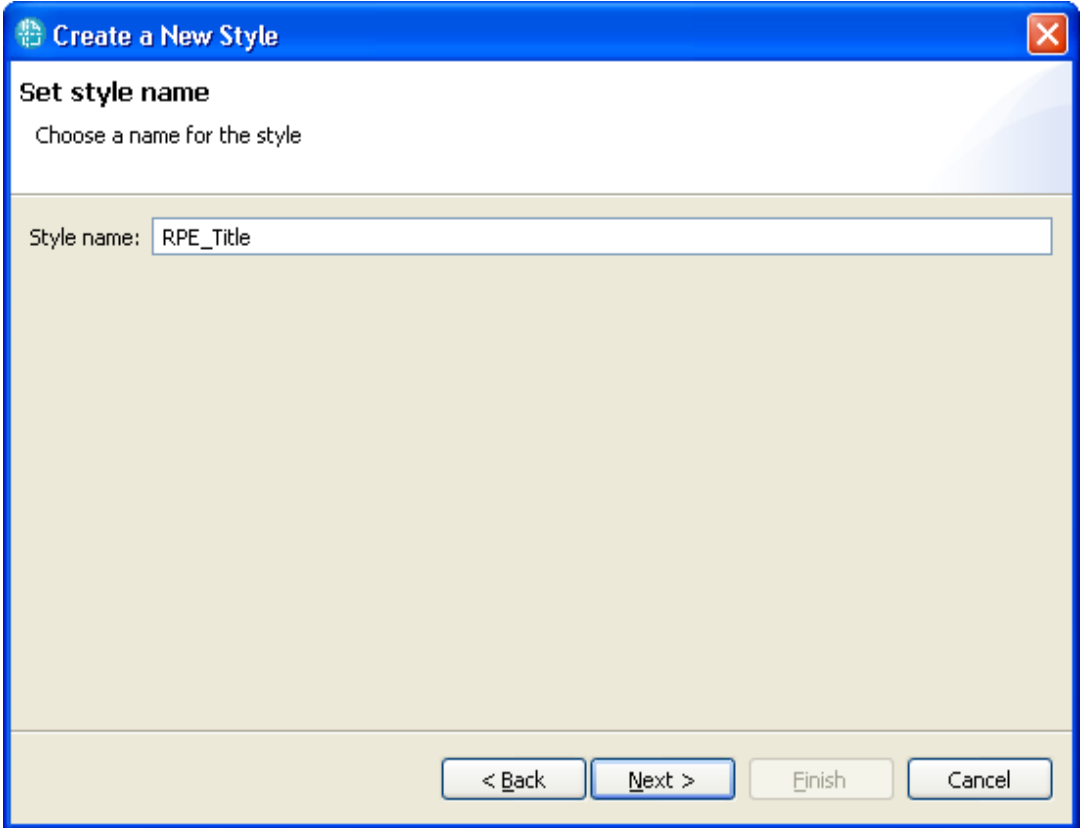


Figure 205

Add the text font, color and size properties to the new style.

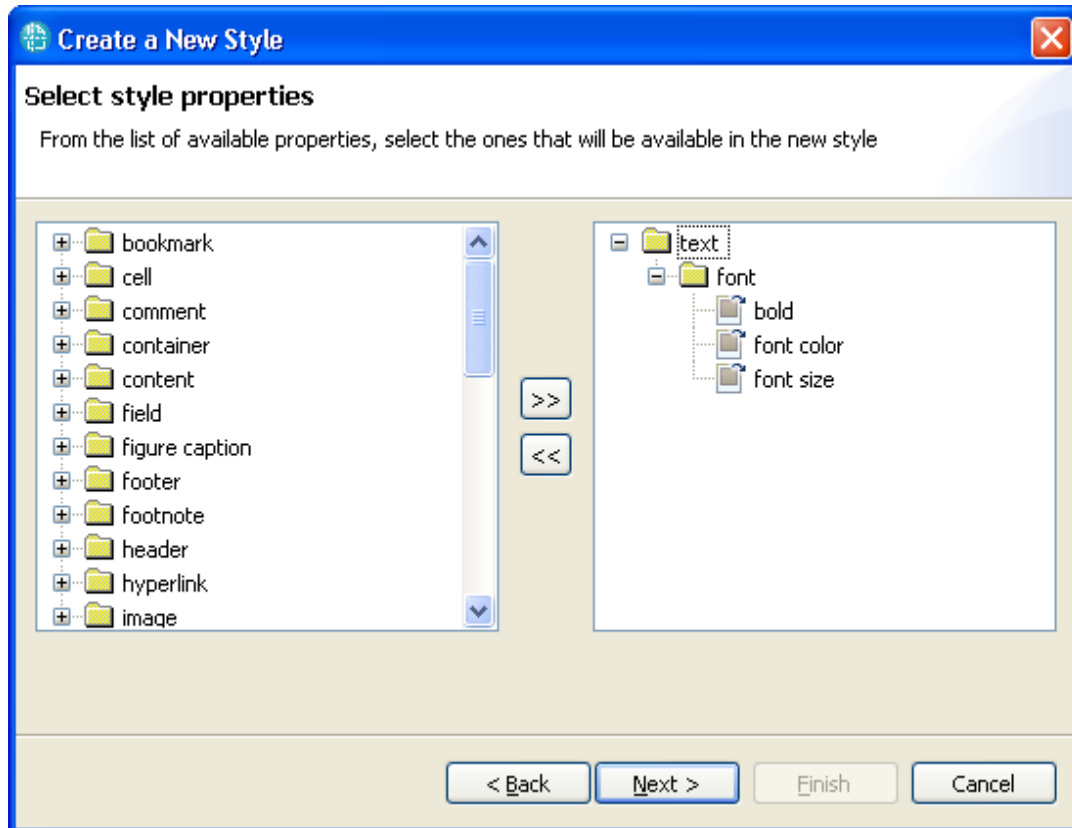


Figure 206

Define the values for the style.

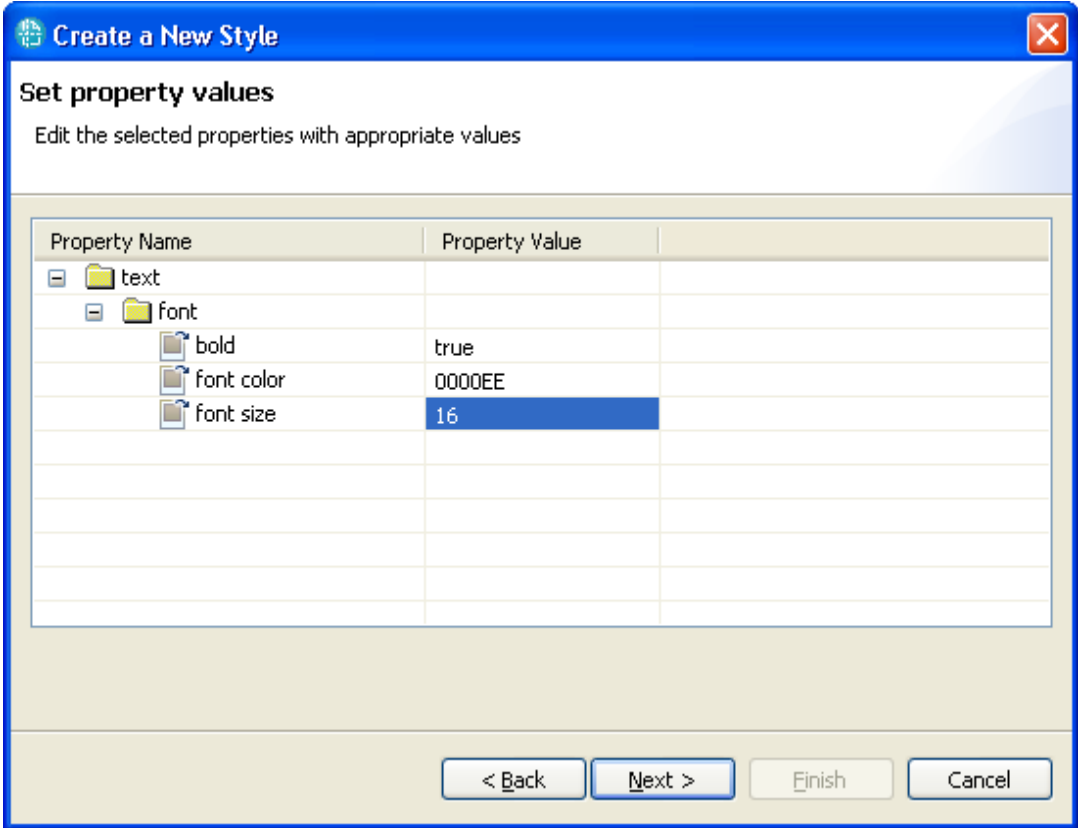


Figure 207

Review the selections and click Finish.

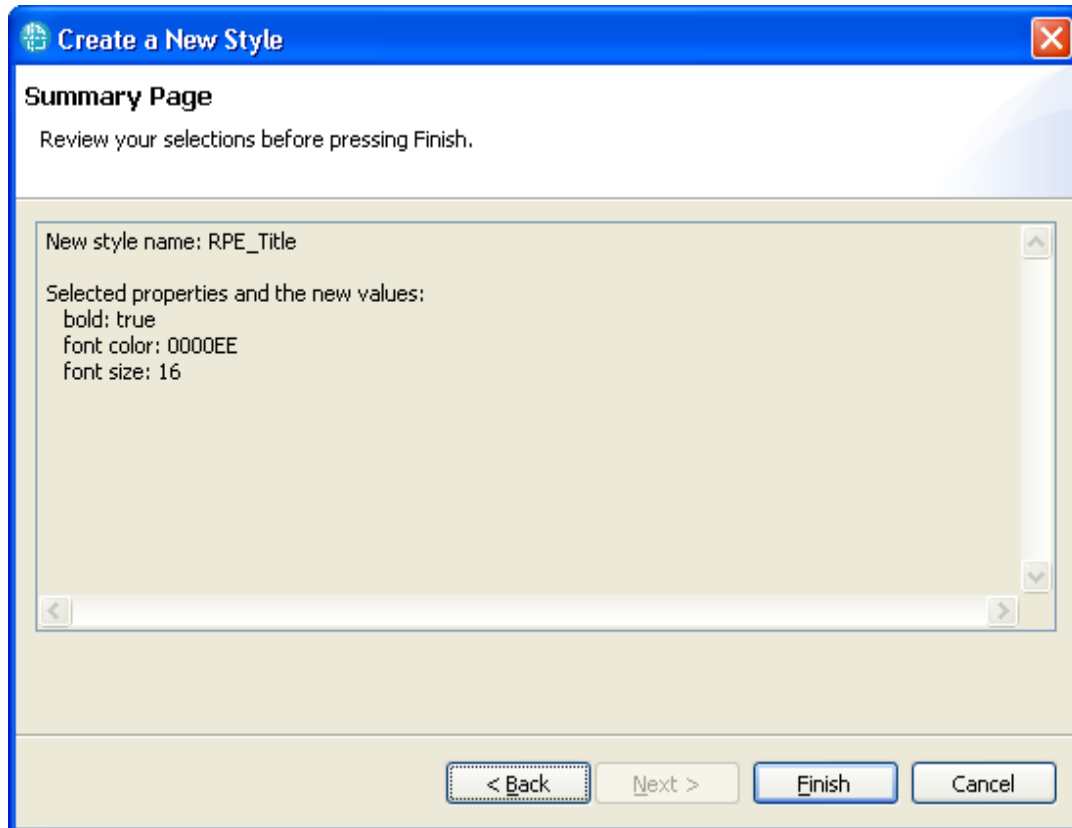


Figure 208

NOTE Save your template.

Assign the “RPE_Title” style to the “Document Title Placeholder” paragraph by dragging the style form outline onto the paragraph.

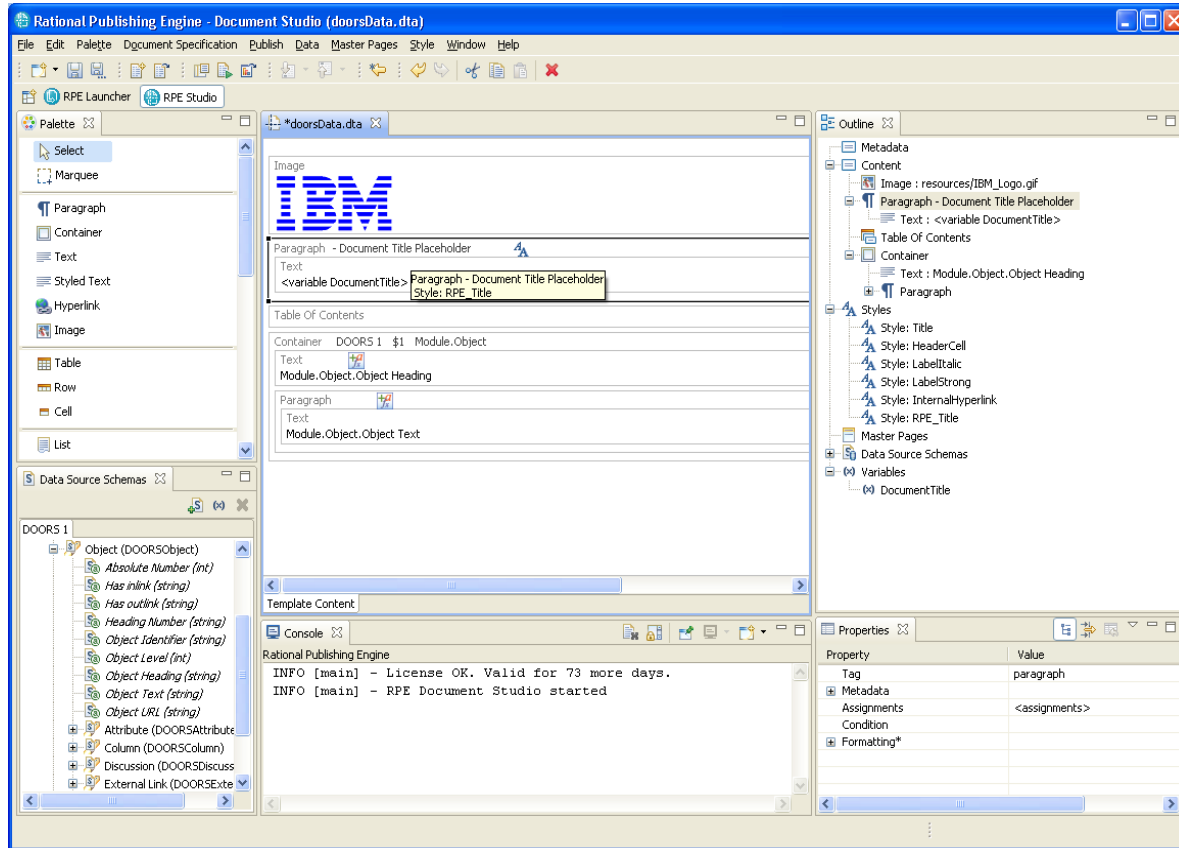


Figure 209

Create a master page named “MP Data”

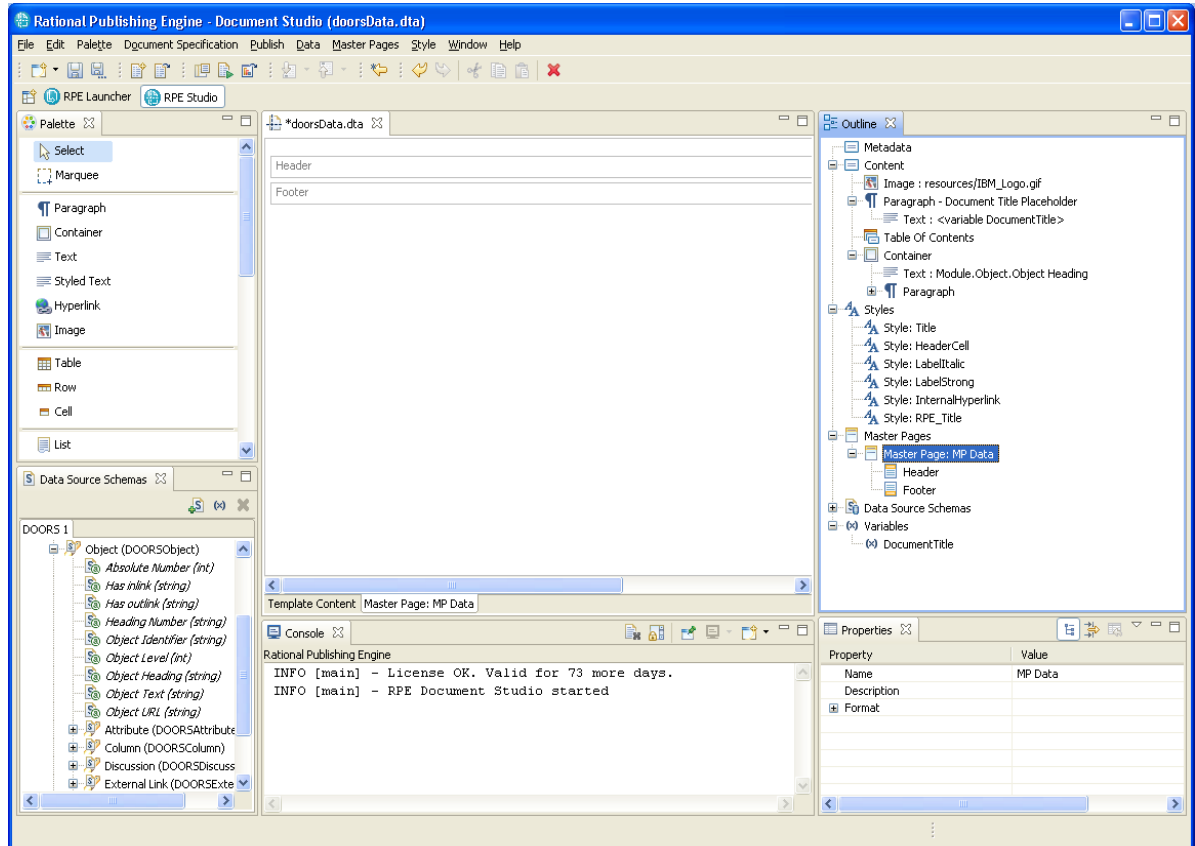


Figure 210

Add the page number field in the footer and a paragraph with a text in the header.

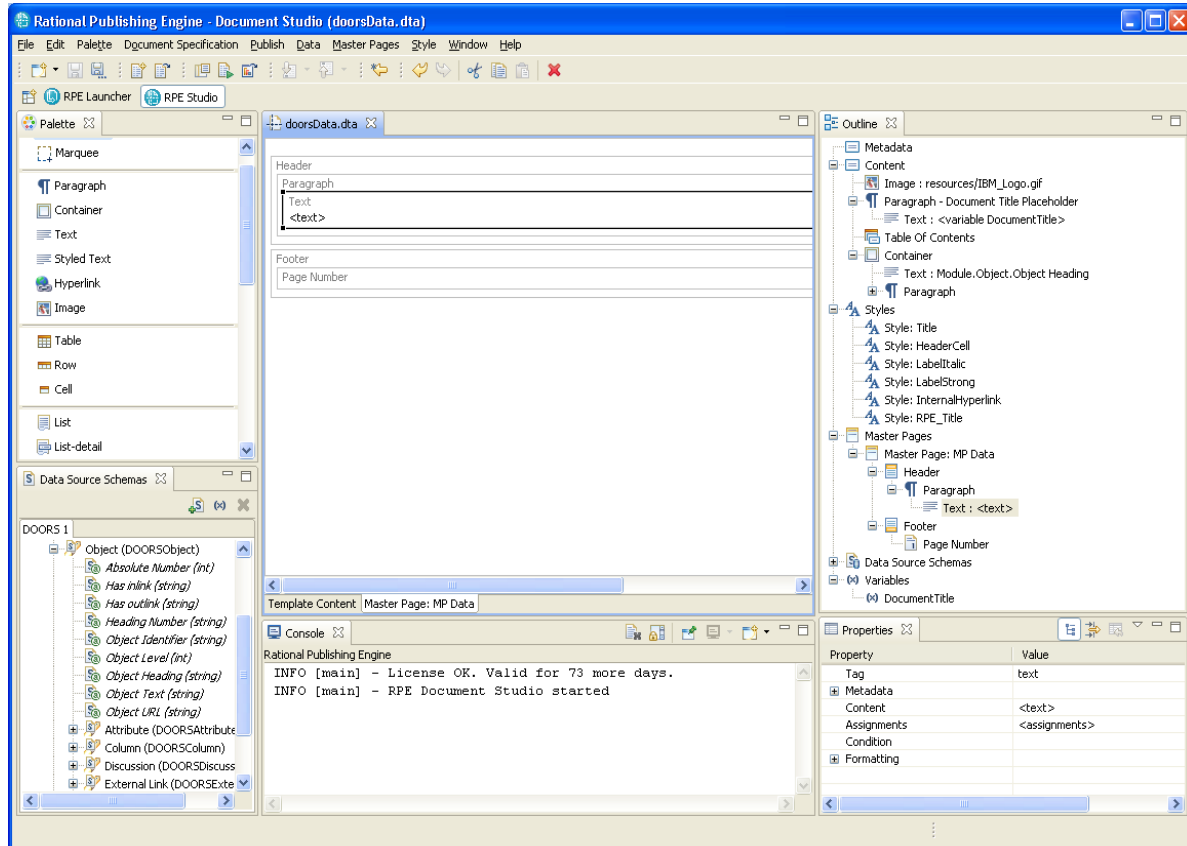


Figure 211

Define a new external variable named “HeaderData” and drag it in the text element from the header.

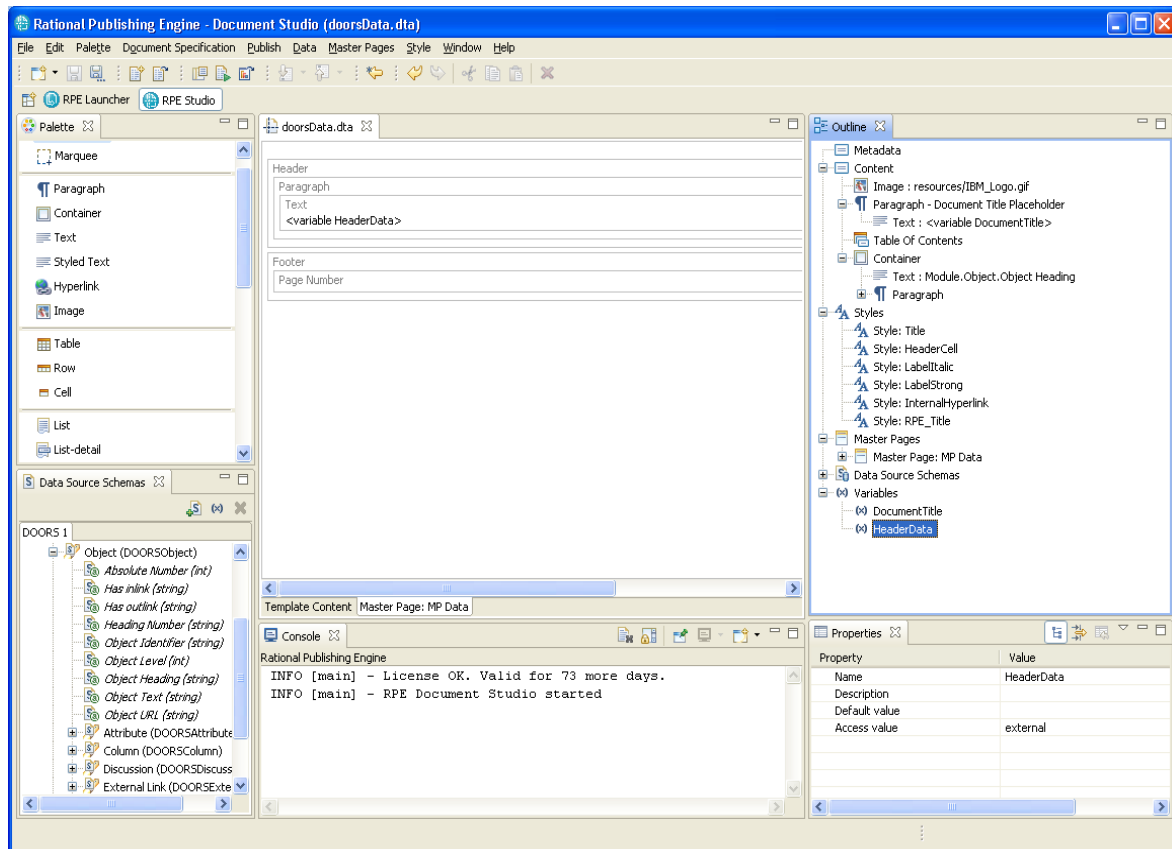


Figure 212

NOTE Save your template.

Assign the page to the container element.

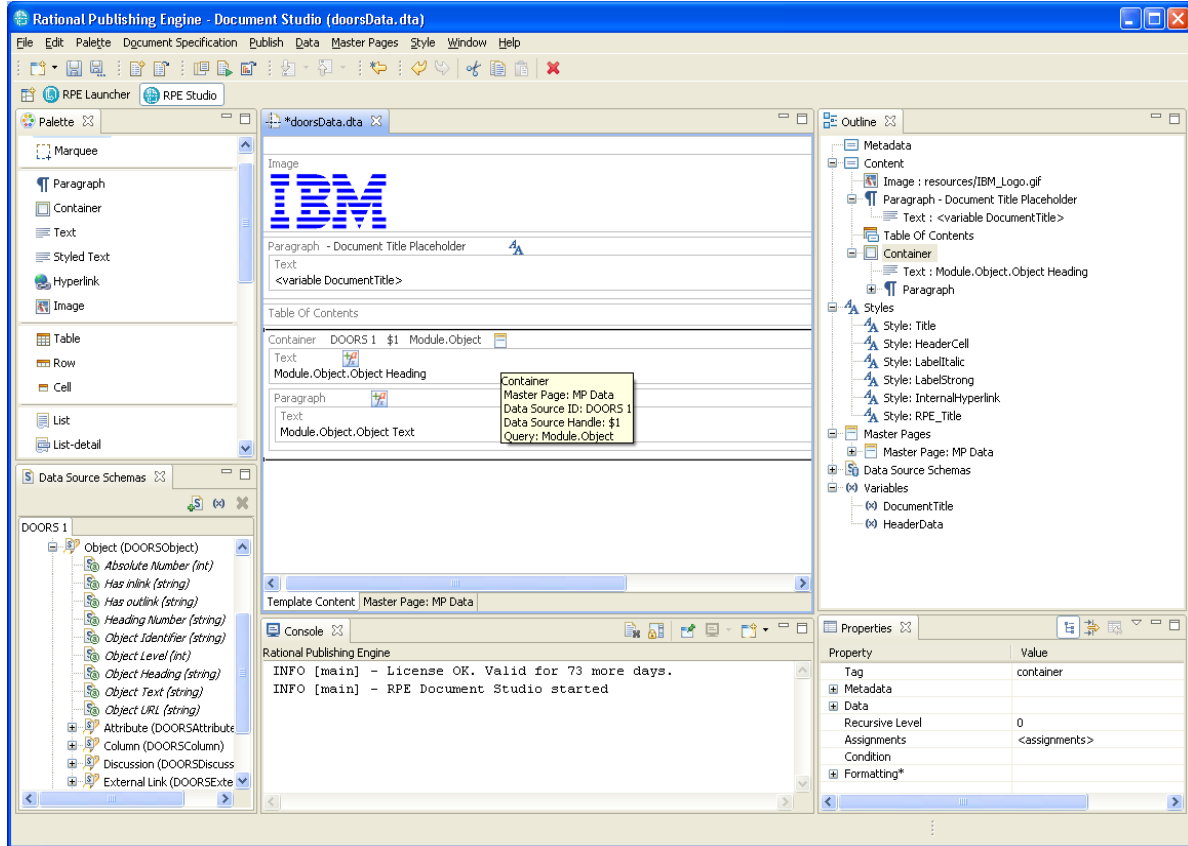


Figure 213

NOTE Save your template.

Using DOORS Link

Create a new container in the “Module.Object” container and assign to it the query “Module.Object.Link”. Name this container “Link Container” for easier reference.

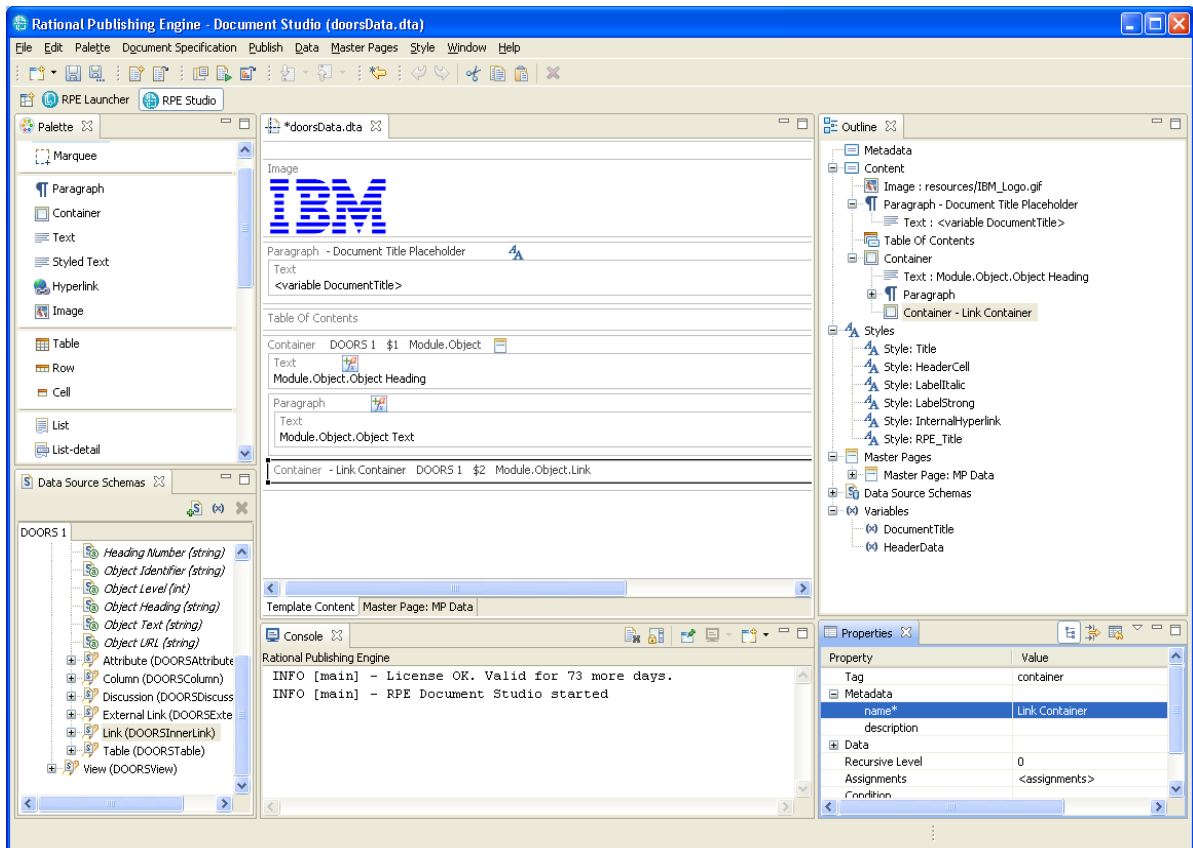


Figure 214

Edit the filter for the “Module.Object.Link” query and use the native filter to define it as: “Link direction in”

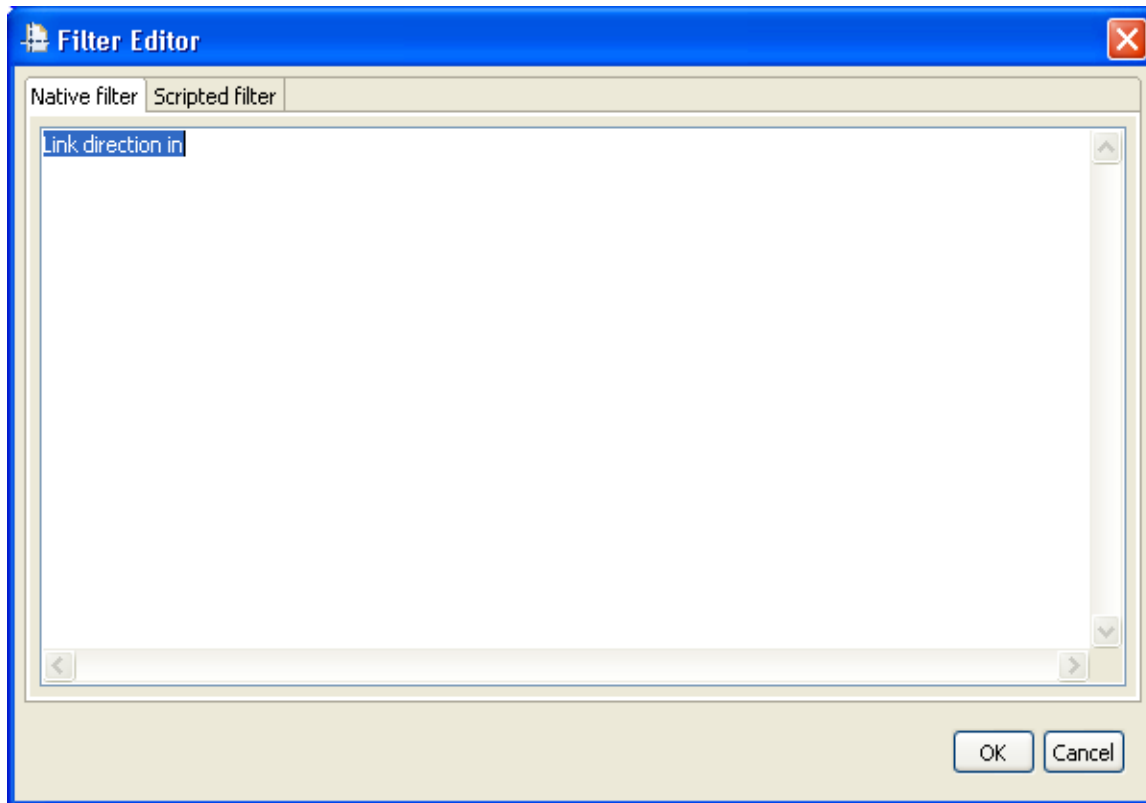


Figure 215

Put a paragraph inside the “Link Container” and assign the “Module.Object.Link.Linked Object” query.

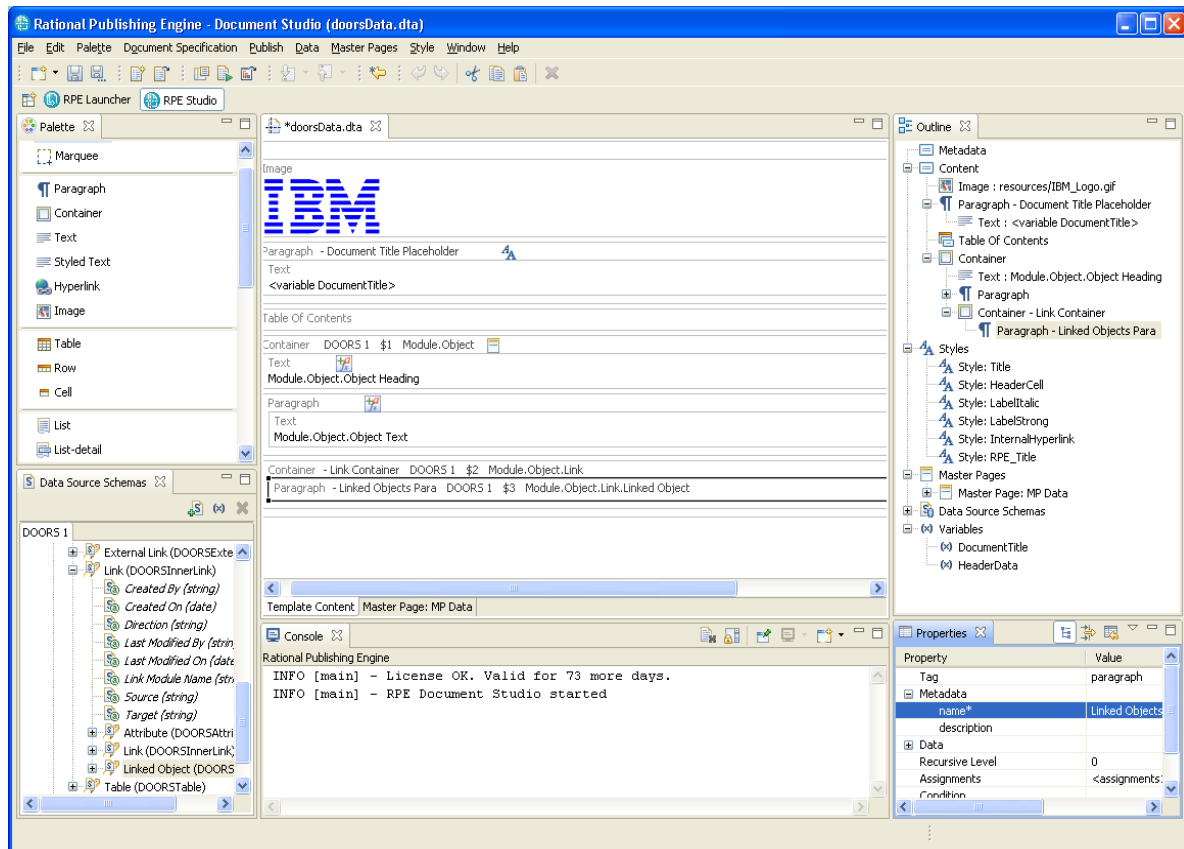


Figure 216

NOTE Save your template

Add the Object Text attribute of the 'Linked Object' in the newly created element.

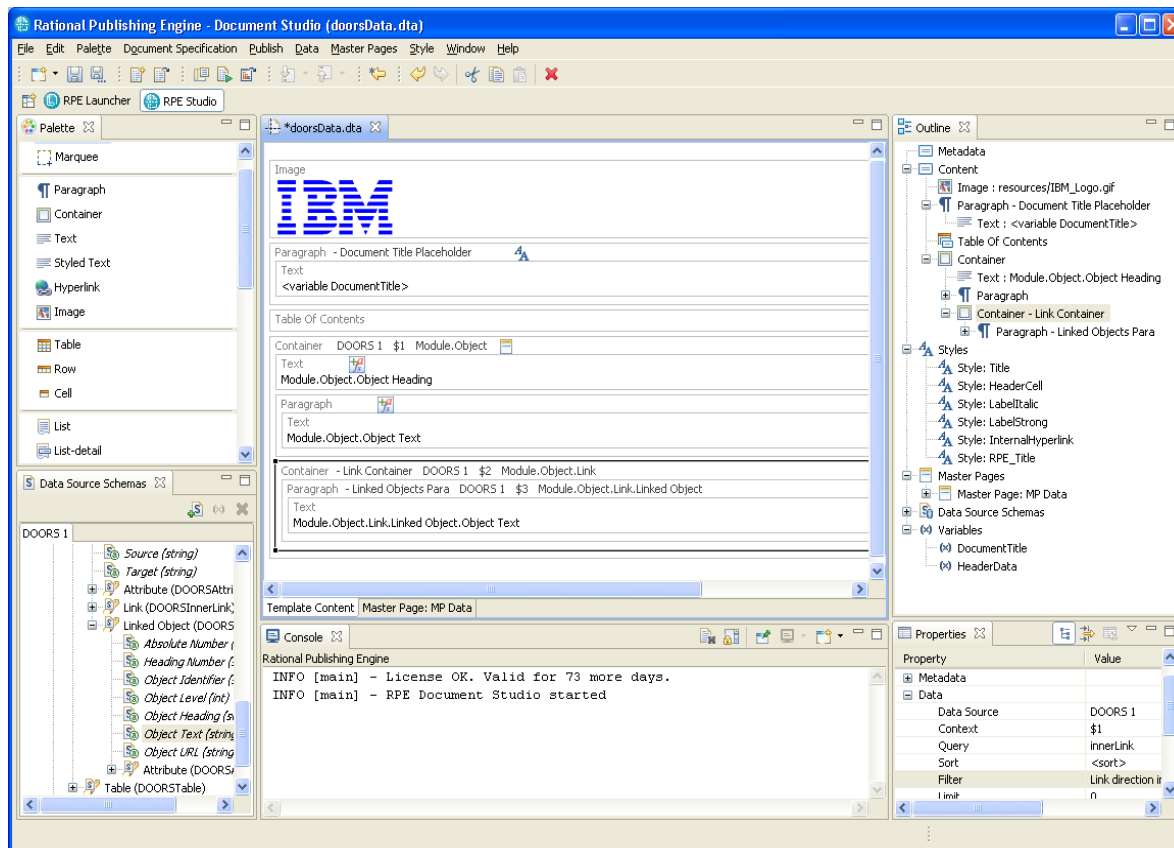


Figure 217

NOTE Save your template.

Generate the document

Switch to the RPE Launcher Perspective. If the perspective icon is not visible under the toolbar use the button located in the top left part of the screen, right below the toolbar.

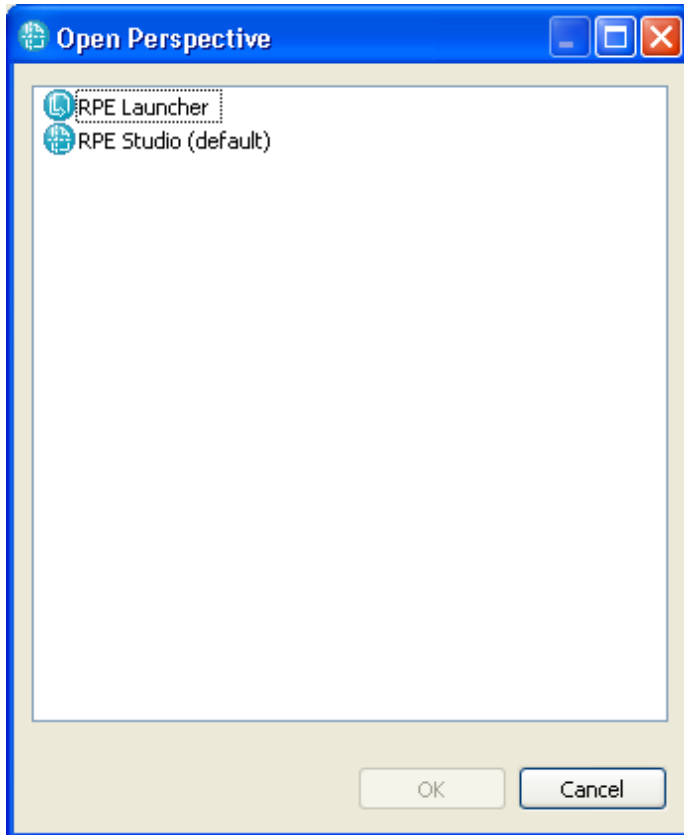


Figure 218

The launcher perspective is displayed.

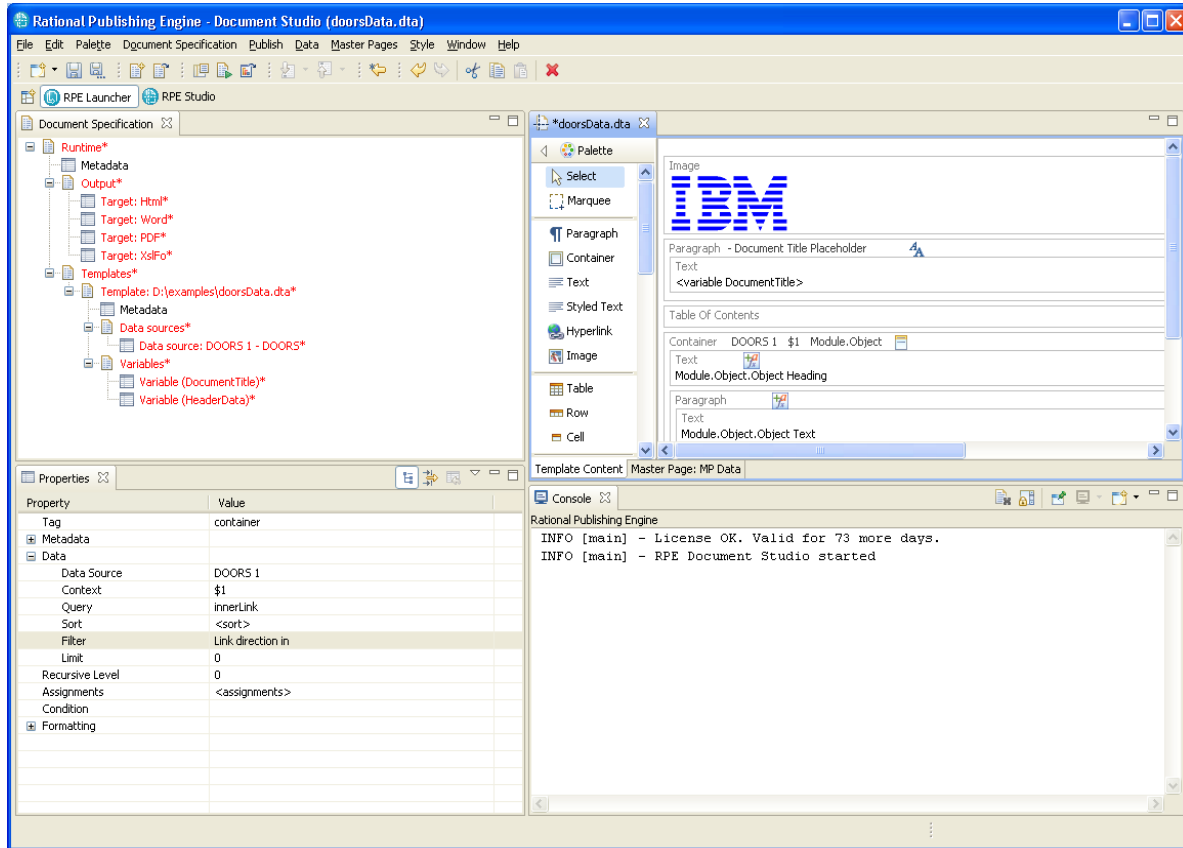


Figure 219

NOTE When the launcher is embedded in RPE Document Studio and the “keep document specification synchronized” option is checked in the preference page, the document specification is synchronized with the edited template at every load/close/save operation. Otherwise you need to manually synchronize the document specification to ensure all the data source schemas from the template have a data source equivalent in the document specification.

Configure the Data source

Using the “Configure Data Source” mechanism described in the Input section you can configure the DOORS Data Source.

The screenshot displays the Rational Publishing Engine interface. The top pane, titled '*Document Specification', shows a hierarchical tree structure:

- Runtime*
 - Metadata
 - Output*
 - Target: Html*
 - Target: Word*
 - Target: PDF*
 - Target: XslFo*
 - Templates*
 - Template: D:\examples\doorsData.dta*
 - Metadata
 - Data sources
 - Data source: DOORS 1 - DOORS
 - Variables*
 - Variable (DocumentTitle)*
 - Variable (HeaderData)*

The bottom pane, titled 'Properties', shows a table of properties for the selected 'Data source: DOORS 1 - DOORS'.

Property	Value
name	DOORS 1
description	
type	DOORS
driver	eval
URI	/demo/car/Car user reqts
module_id	
doors_home	C:\Program Files\IBM\Rational\DOORS\9.2\bin\doors.exe
doors_param	-data 36677@localhost
username	
password	
baseline	Current
view	Standard view
new_instance	false
command	
ignored	false

Figure 220

Provide values to the two variables.

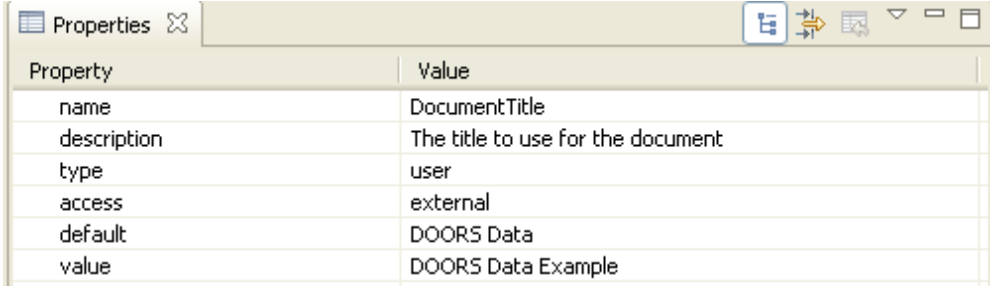


Figure 221

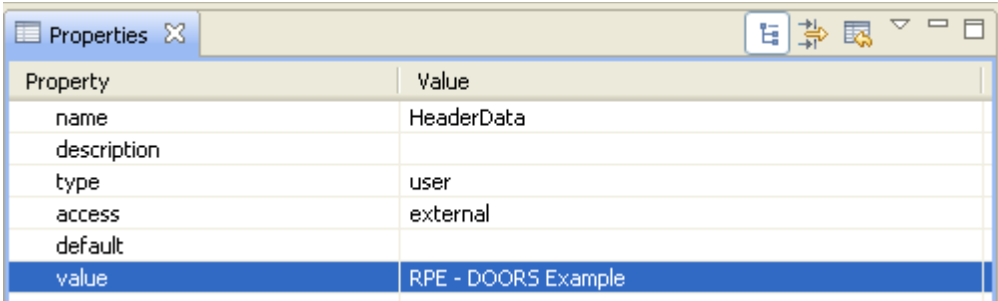


Figure 222

Generate the document

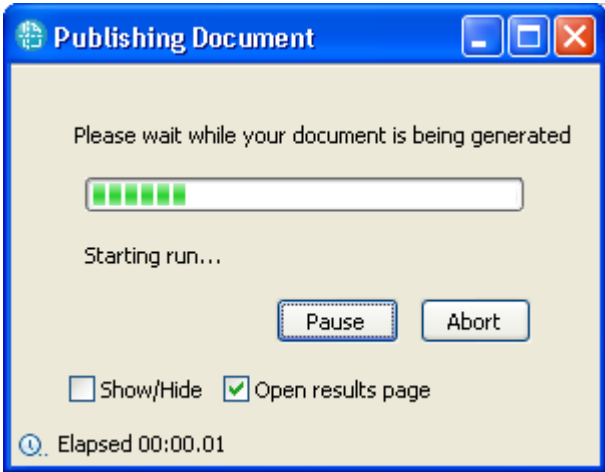


Figure 223

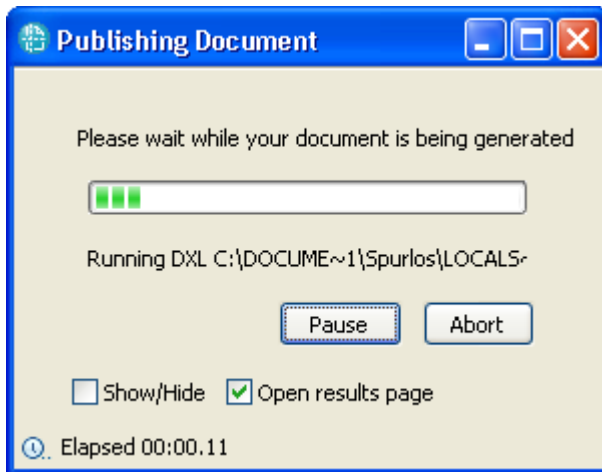


Figure 224

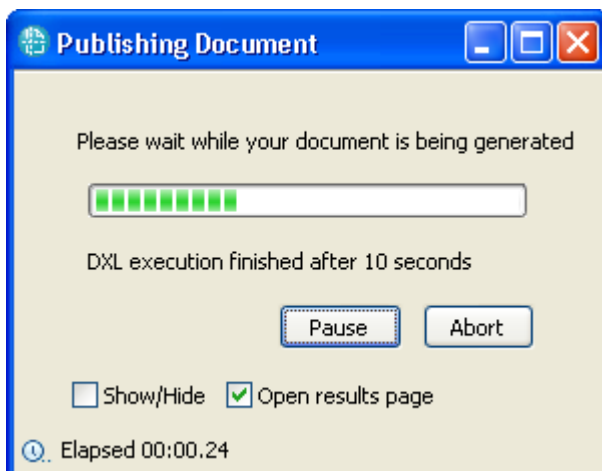


Figure 225

View the results

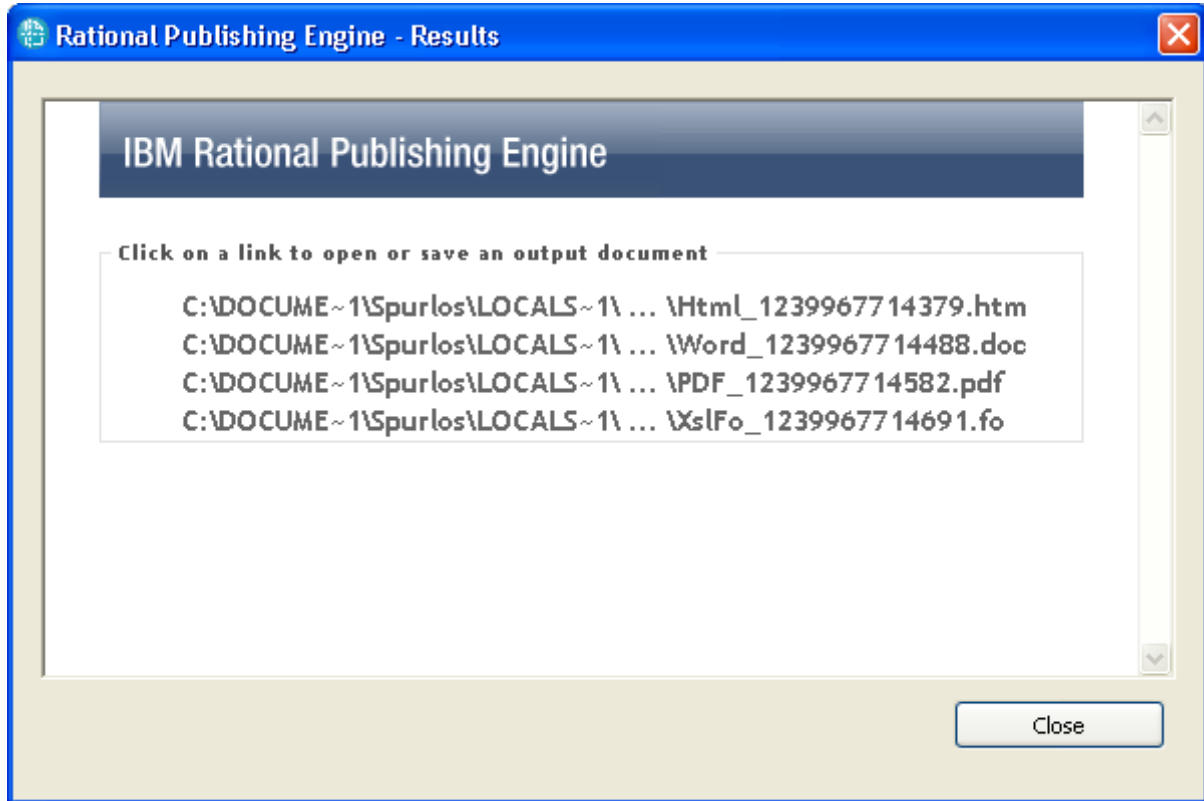


Figure 226

NOTE When you open the generated Word document for the first time you need to manually update the table of contents. You can use the provided “updateTOCs” macro to automate this task.

A template for Tau data

This chapter shows how to build a simple RPE Document Template. The output document will contain the list of top level packages, and for each package the list of classes and the diagrams contained in that package. For each class a table will be generated listing all the class’s attribute names and types.

- Package
 - Diagrams
 - Diagram
 - Diagram
 - ...
 - Classes

- Class
 - Attribute table
- Class
 - Attribute table
- ...

Start Document Studio

Add a Tau Data source Schema using the “Add Data Source Wizard”. Accept the values provided by RPE.

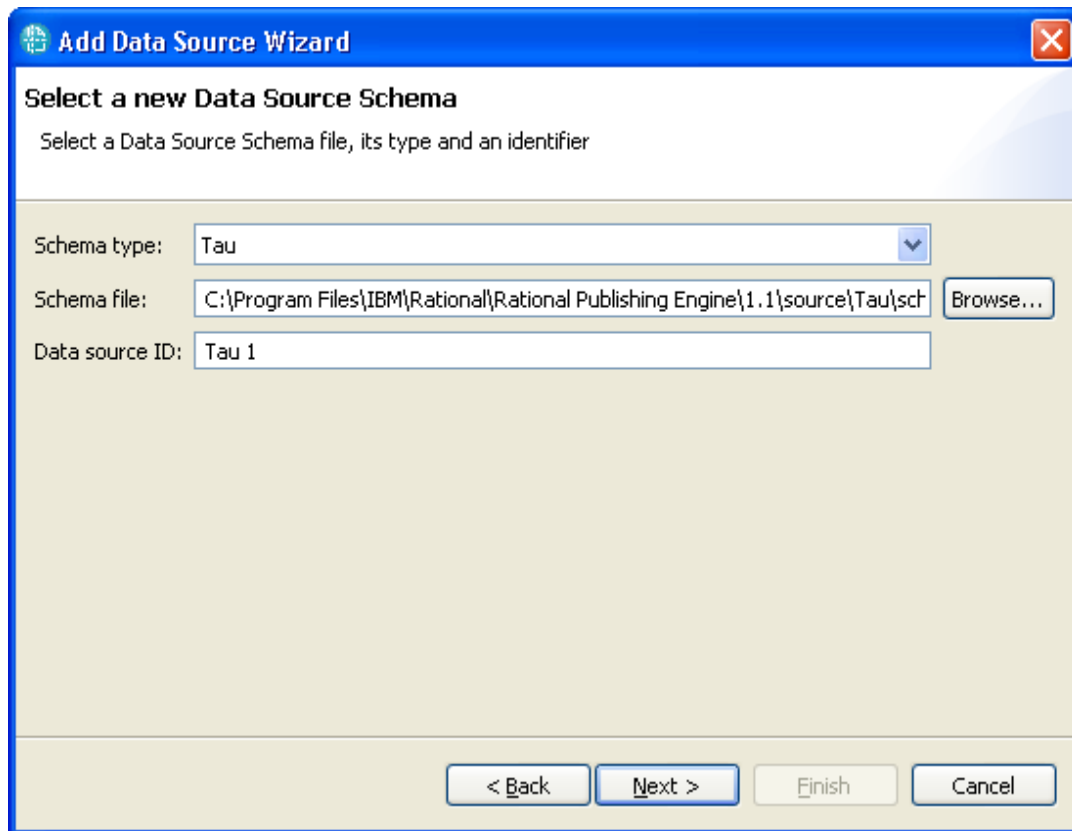


Figure 227

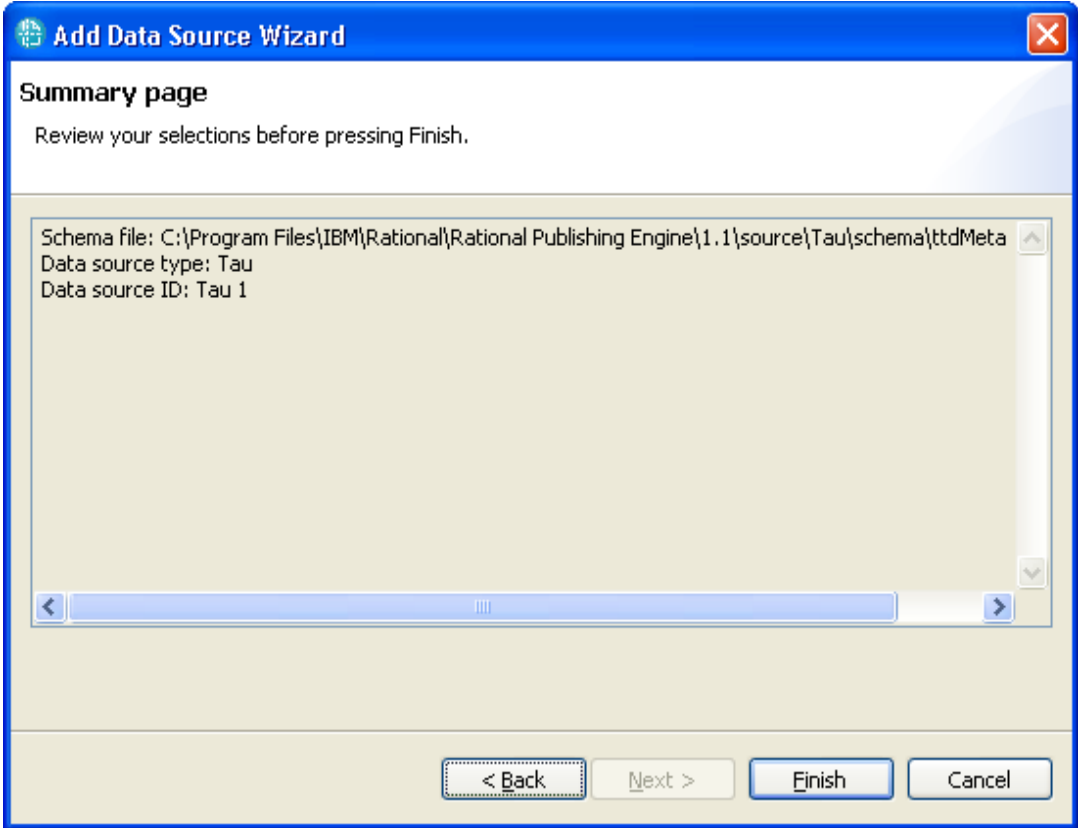


Figure 228

Save the document.

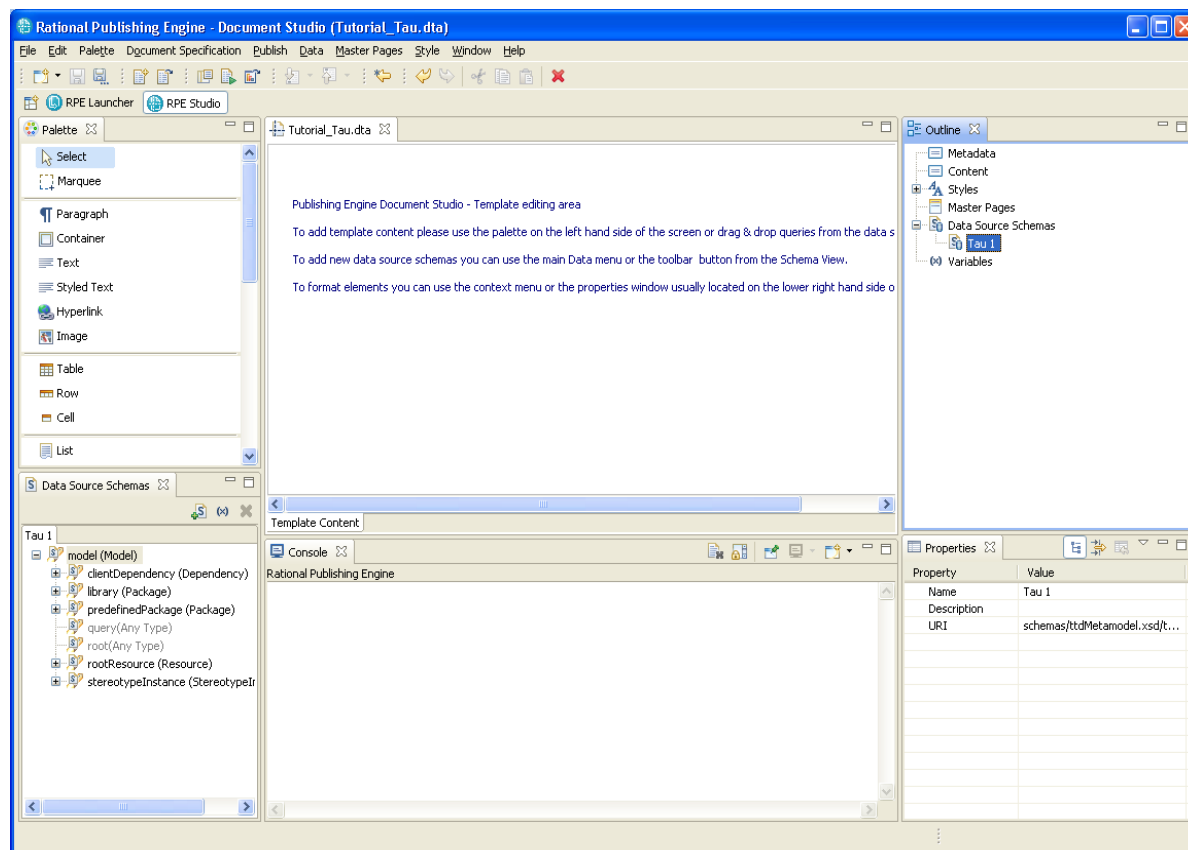


Figure 230

NOTE Save the document template.

Define the template content

Define a query for the top level packages. Add the "Package" cast under the *model.root* element by clicking the "Cast to type" button.

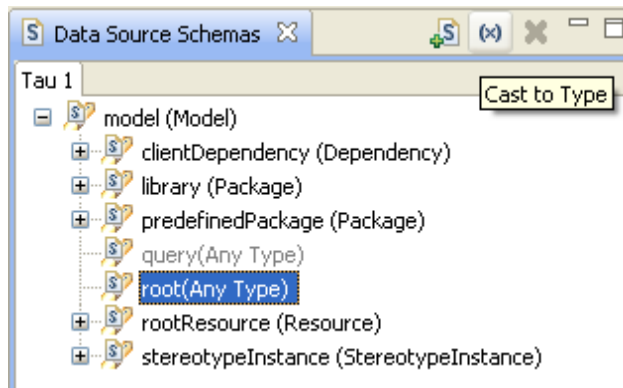


Figure 231

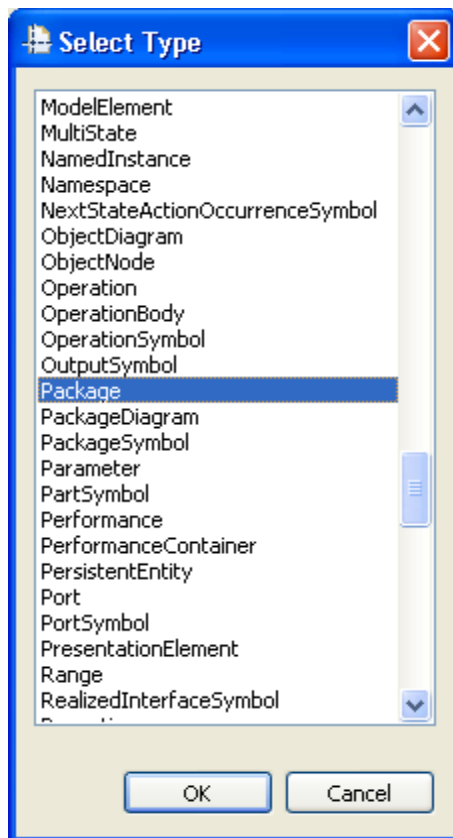


Figure 232

Create a container element in the template and drag the newly added Package element to it. Name the container “Root Packages” for easier referencing.

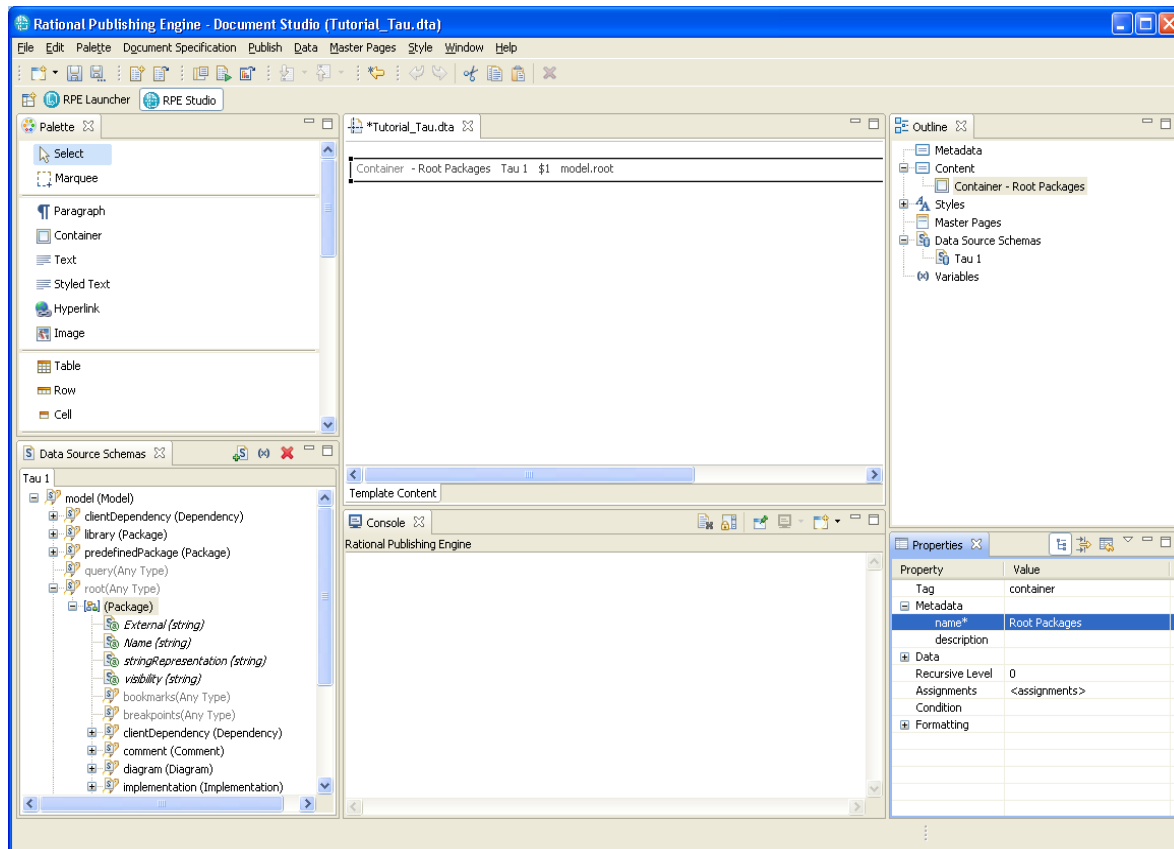


Figure 233

NOTE Save the document template.

Create a paragraph in the container element and add the “Name” attribute.

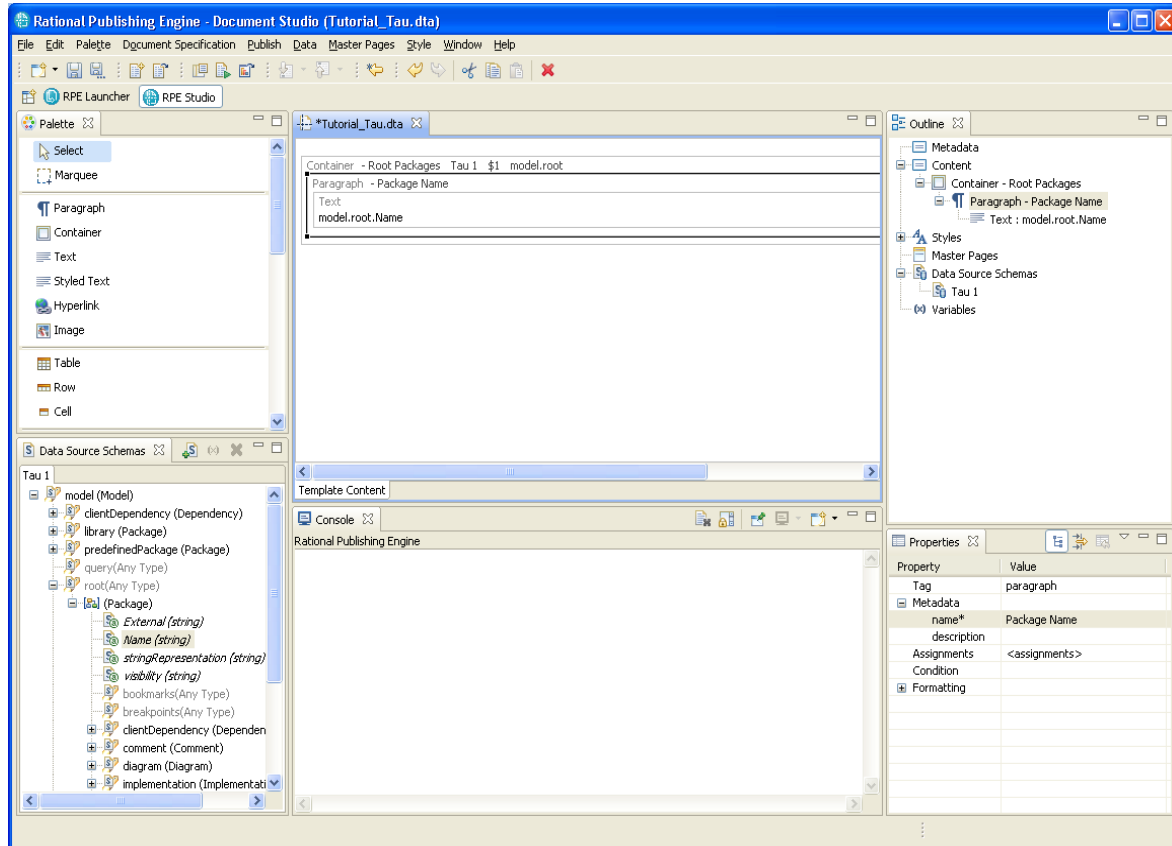


Figure 234

NOTE If you would generate the document using the template in its current form, the output will contain the name of all the top level packages in the model.

- Package
- Package
- Package

NOTE Save the document template.

To continue, add the static text heading **Diagrams** for the list of diagrams by adding a Paragraph into the Container, adding Text into the paragraph, then double-click the text and set the value to **Diagrams**.

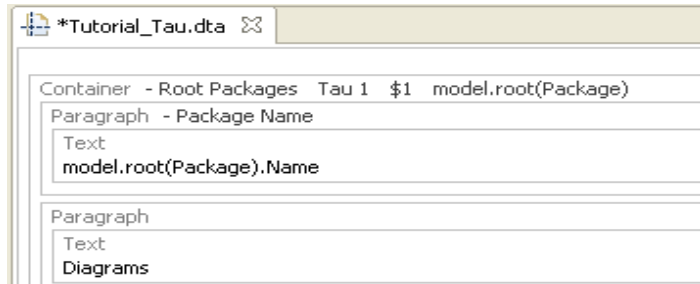


Figure 235

Add the package diagram element to the template in a new container created in the “Root Packages” container.

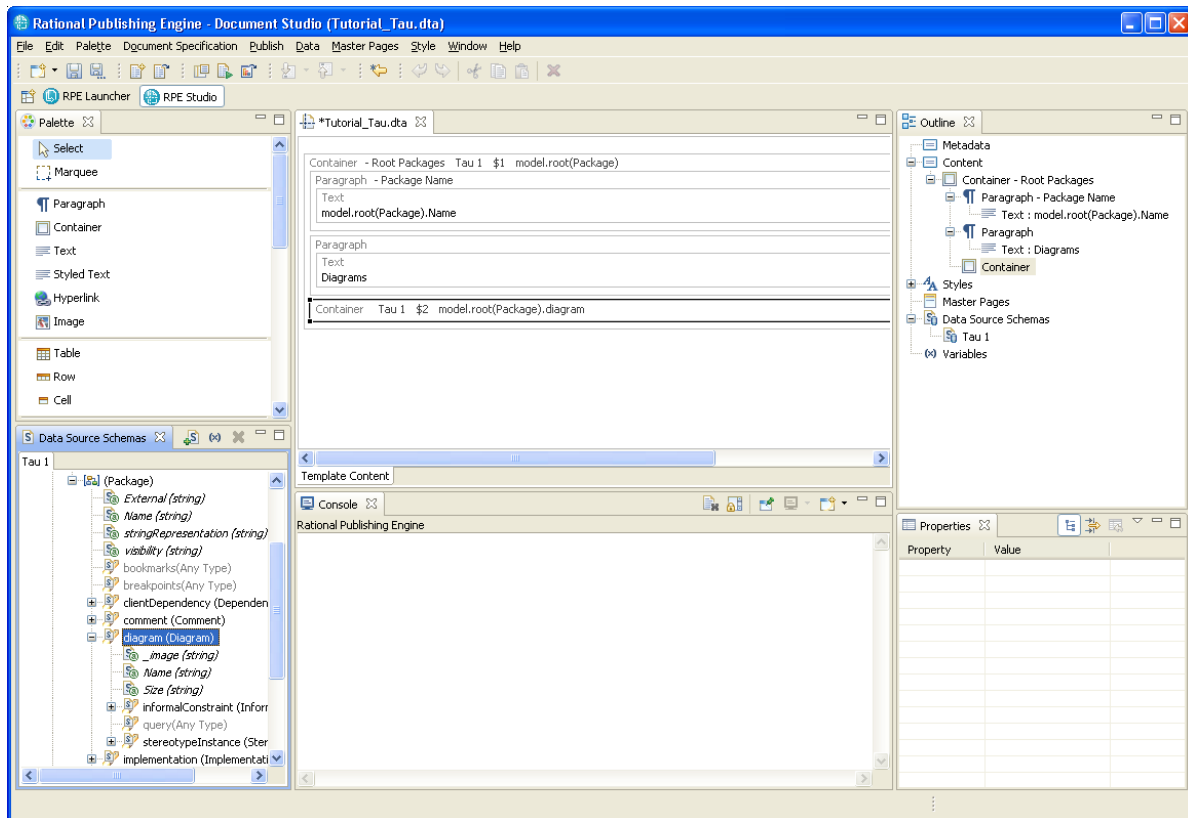


Figure 236

NOTE If you are interested in a specific diagram type you can add a cast to that diagram type below the diagram(Diagram) node and use it instead of using the generic “Diagram” type.
Add an image element in the container. Set its content to be the “_image” attribute of *model.root(Package).diagram(Diagram)*.

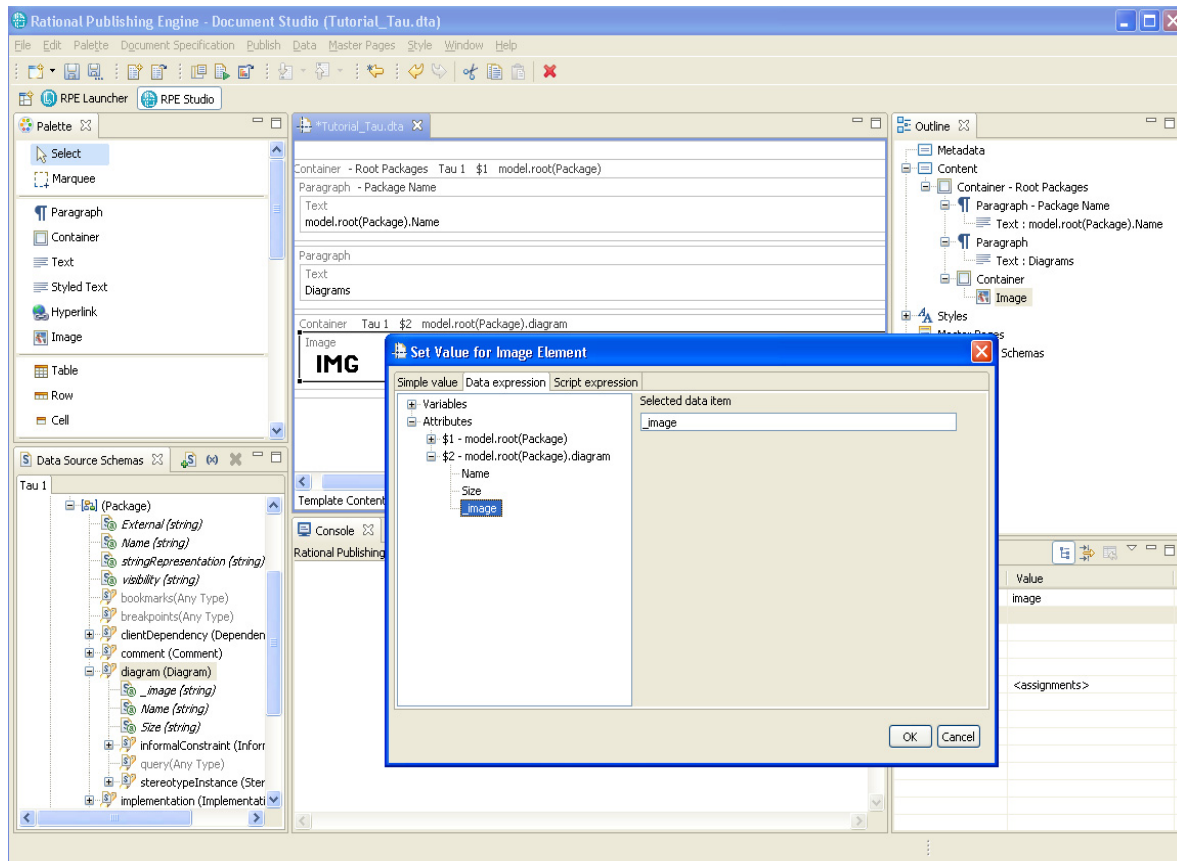


Figure 237

Add a paragraph with a figure the name of the diagram bellow its image. Additionally you can add a figure caption field so you can build a Table of Figures.

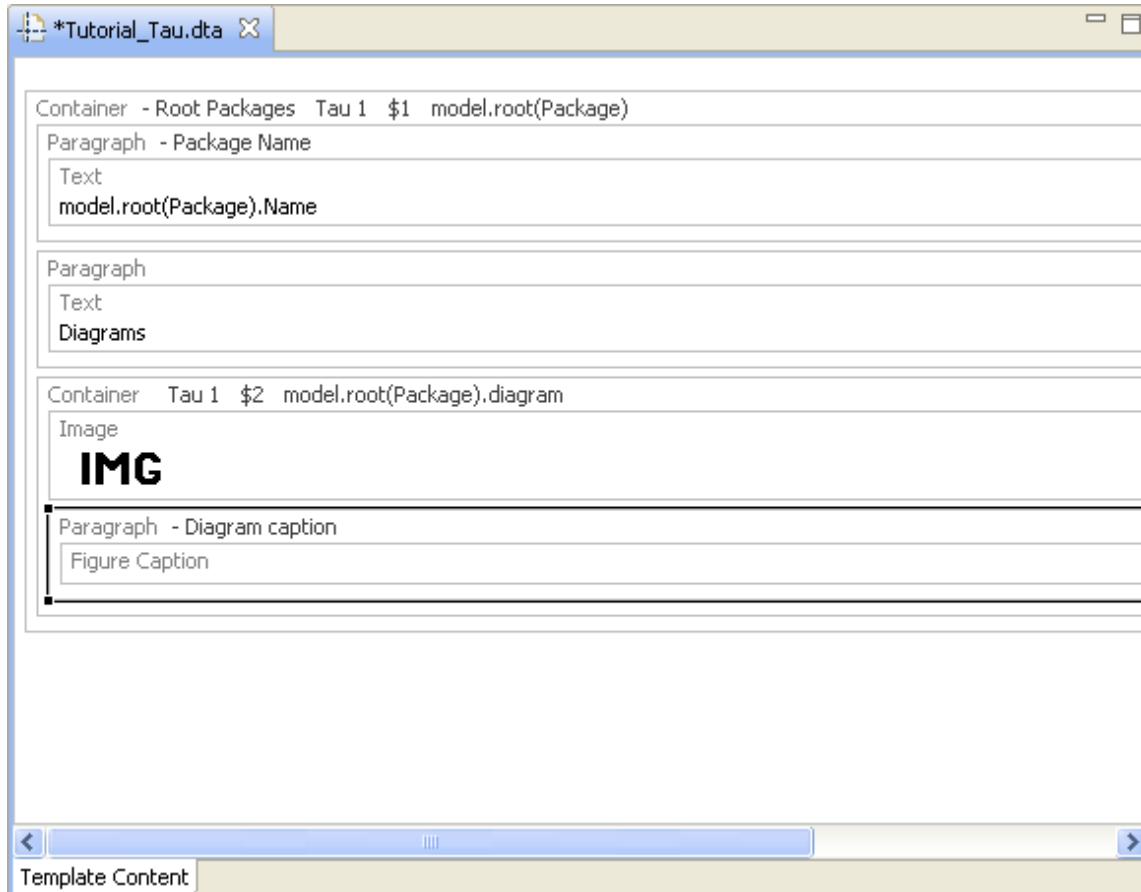


Figure 238

NOTE If you were to generate the document using the template in its current form, the output will contain the name of all the top level packages in the model, and all the diagrams in each package:

- Package
 - Diagrams
 - Diagram Image
 - Figure number
 - Diagram image
 - Figure number
- Package
 - Diagrams

- Diagram Image
 - Figure number
- Diagram image
 - Figure number

To display the figure caption with the diagram name, use the label property of the caption.

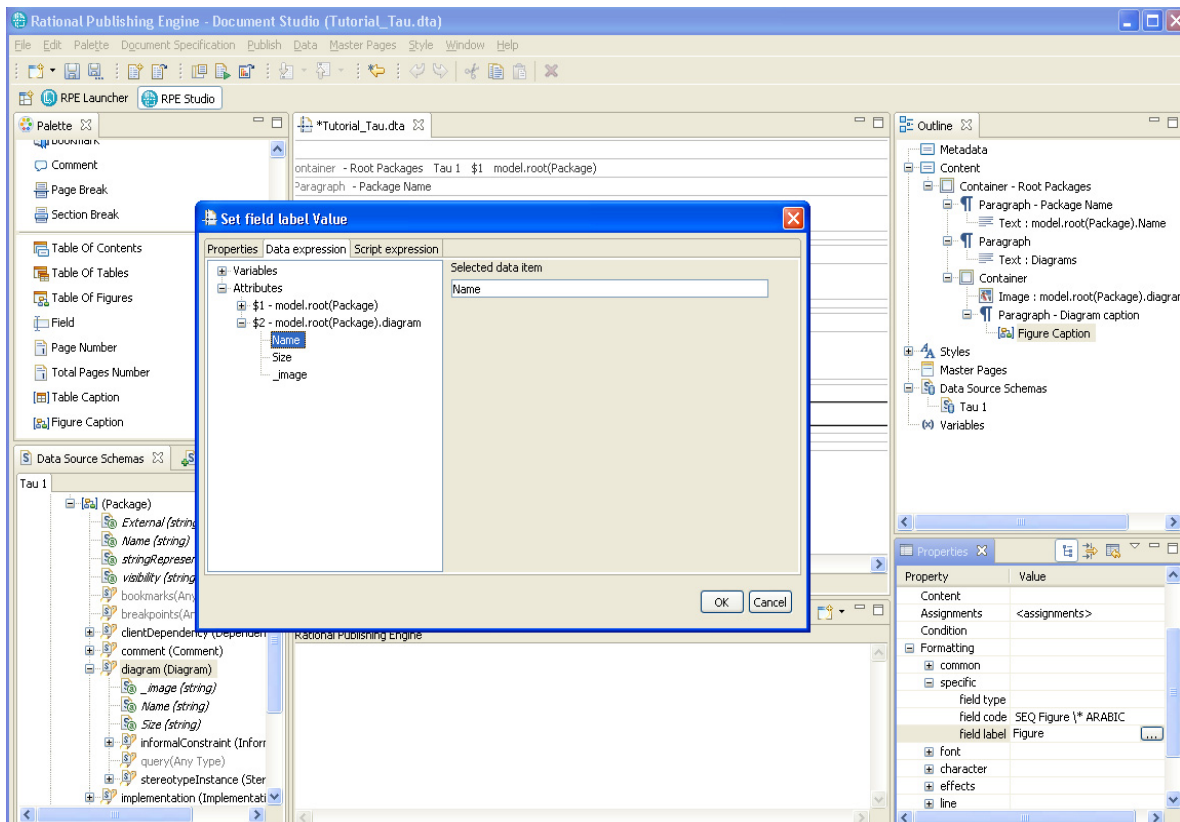


Figure 239

This way the figure caption will contain the diagram name as well.

The next step is to list the classes from each package. First add the static text for the class list.

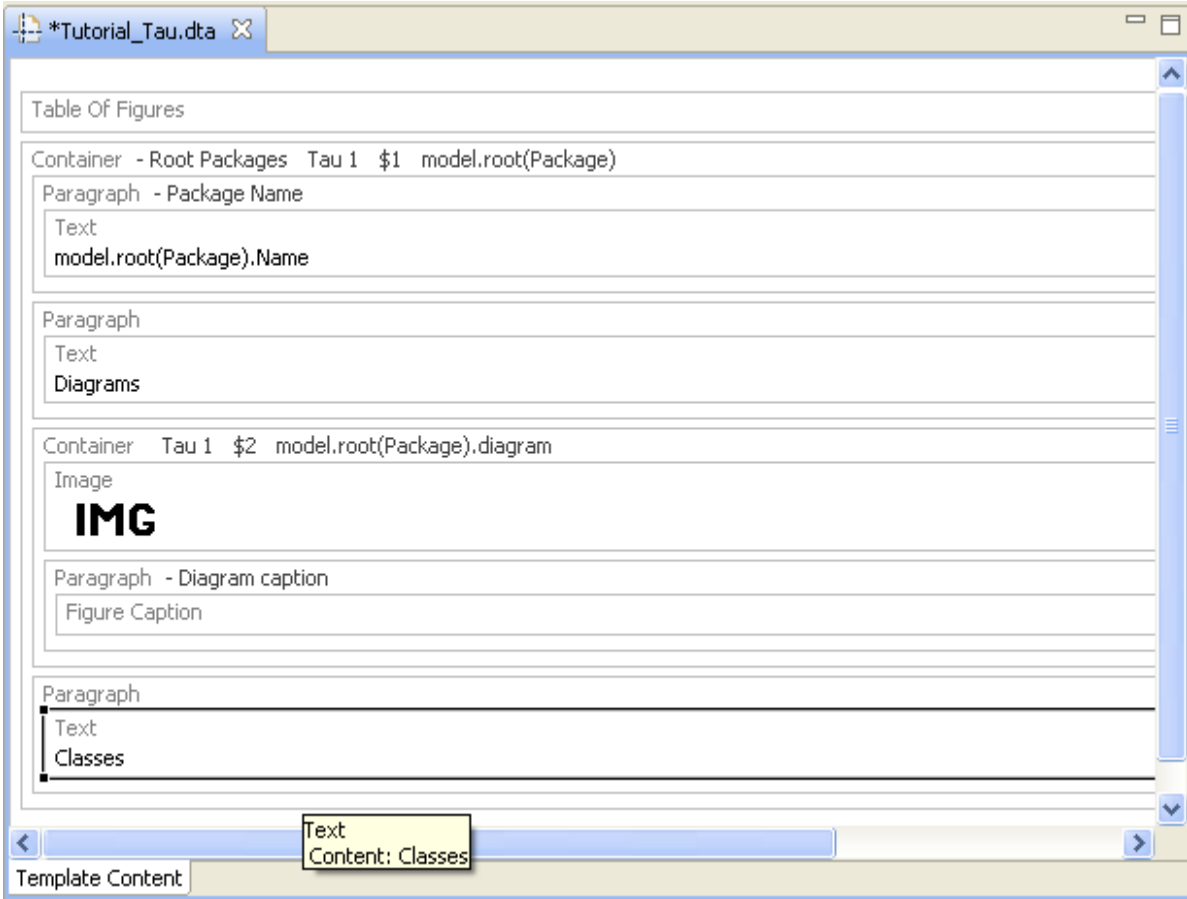


Figure 240

Add the “Class” type cast for the “ownedMember” element of the package and use it to create the query on a new container. Name the container “Class Container” for easier reference.

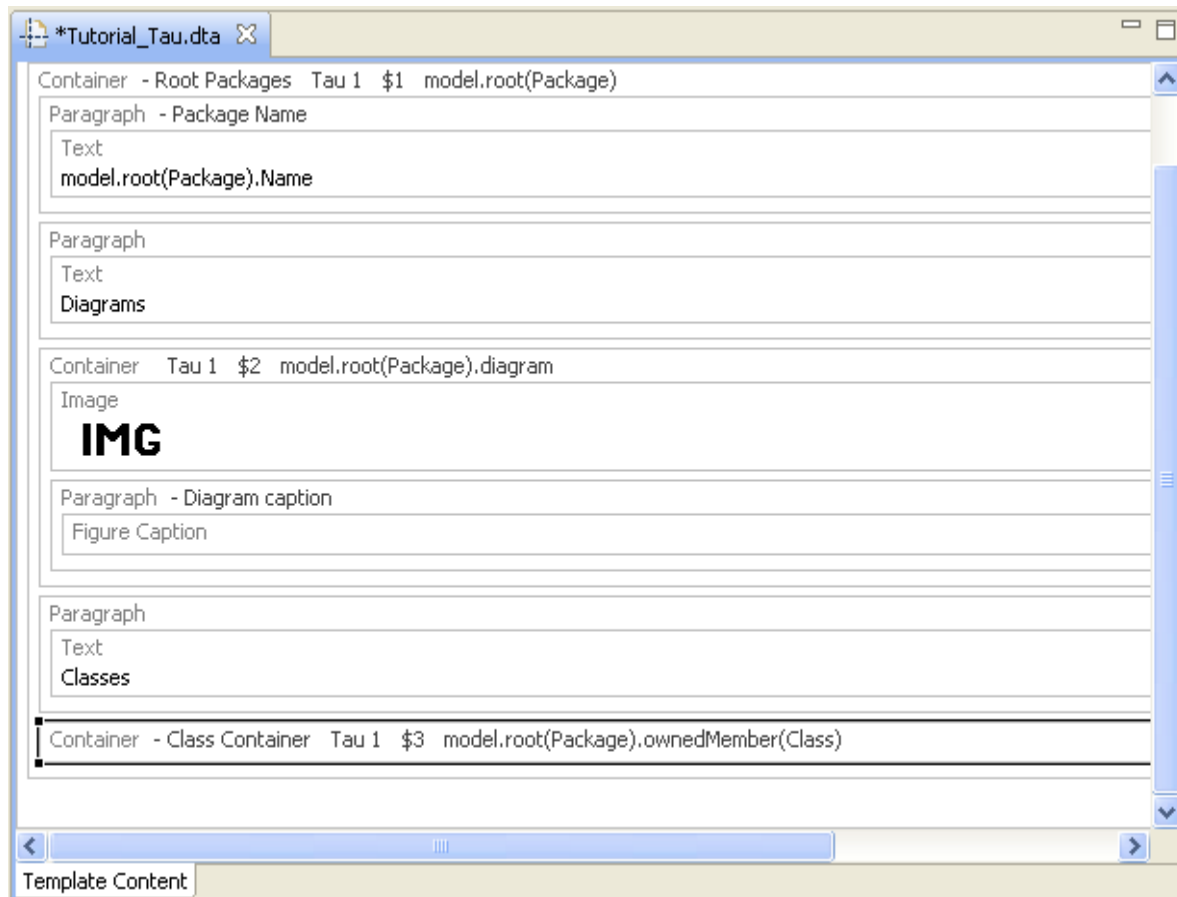


Figure 241

Add the name of the package in its own paragraph inside the “Class Container”.

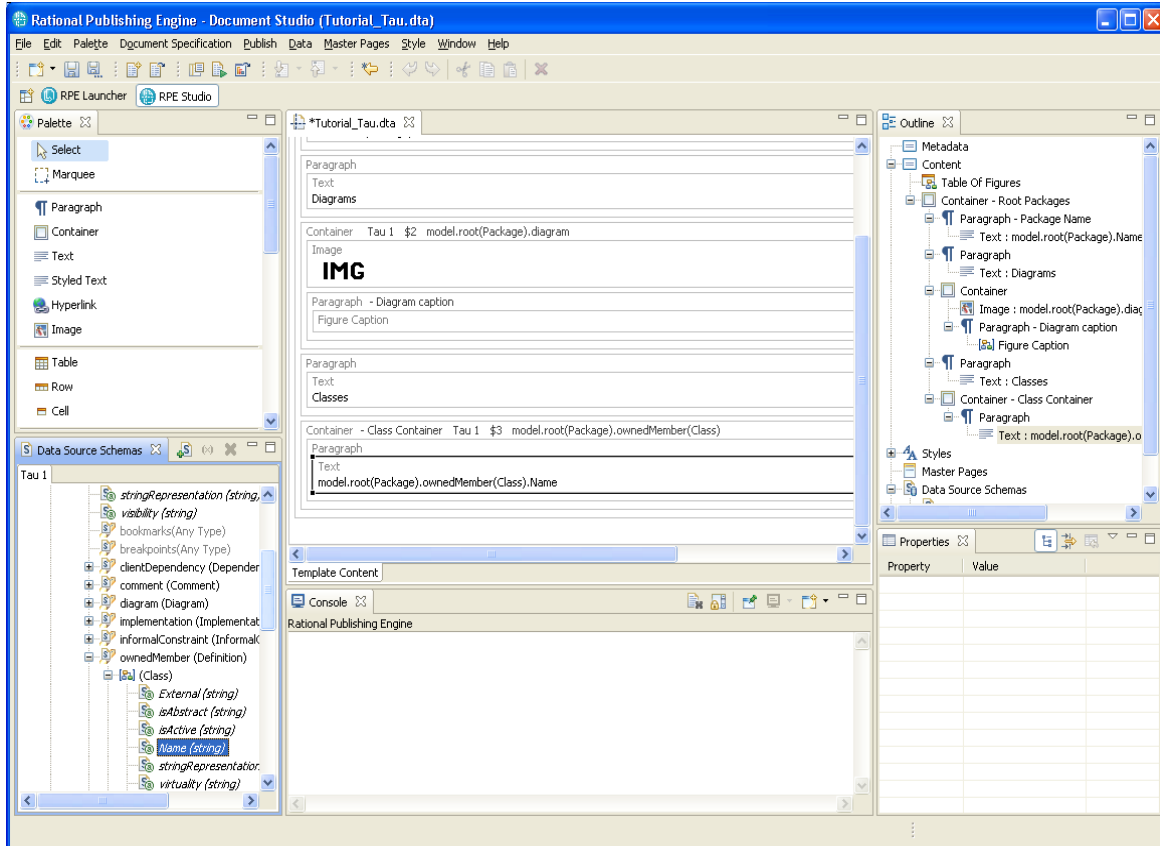


Figure 242

NOTE If you would generate the document using the template in its current form, the output will contain the name of all the top level packages in the model, and all the diagrams in each package:

- Package
 - Diagrams
 - Diagram (image and name)
 - Diagram (image and name)
 - ...
 - Classes
 - Class 1
 - Class 2
 - Class 3
 - ...
- Package

- Diagrams
 - Diagram (image and name)
 - Diagram (image and name)
 - ...
- Classes
 - Class 1
 - Class 2
 - Class 3
 - ...
- ...

NOTE Save the document template.

Add a new text element after the text element containing the class name and set its content to the static text “ (Active)”. Format the text as italic.

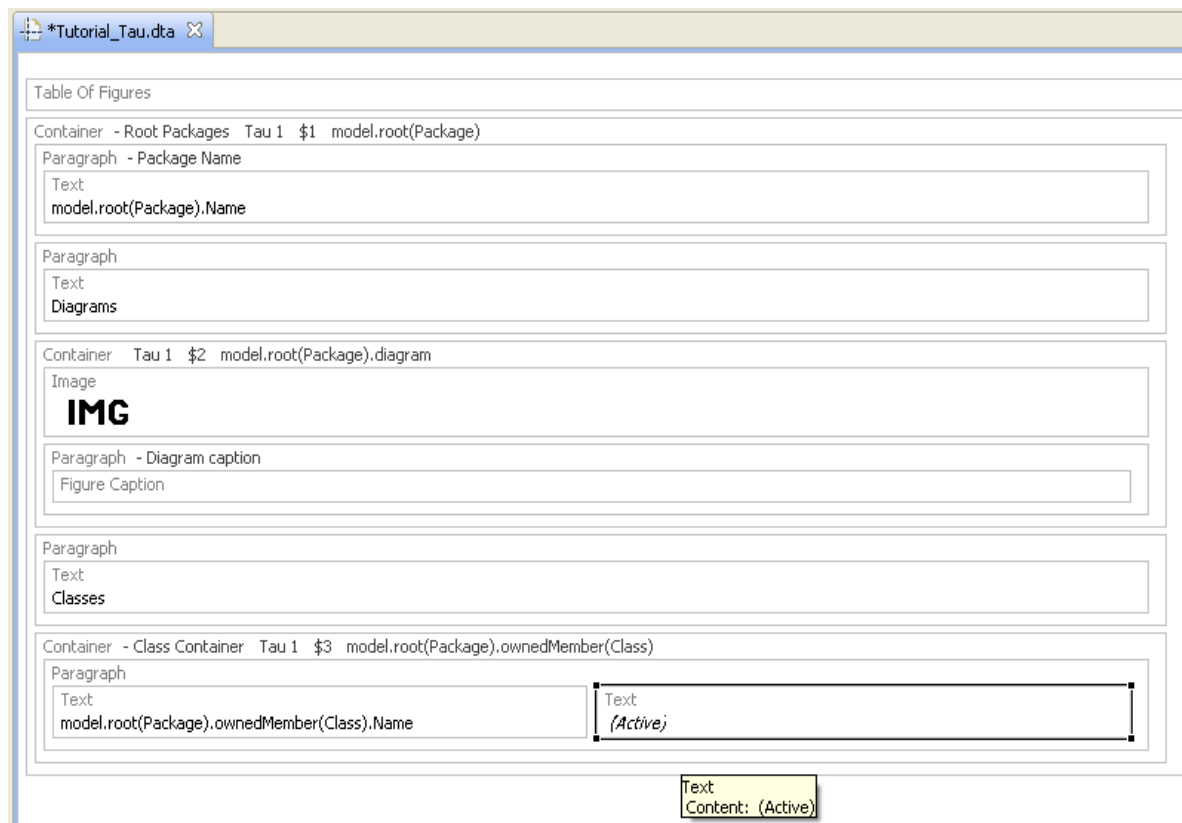


Figure 243

So that this text is added only for active classes a condition need to be specified. The condition is a script using the “isActive” attribute of the class.

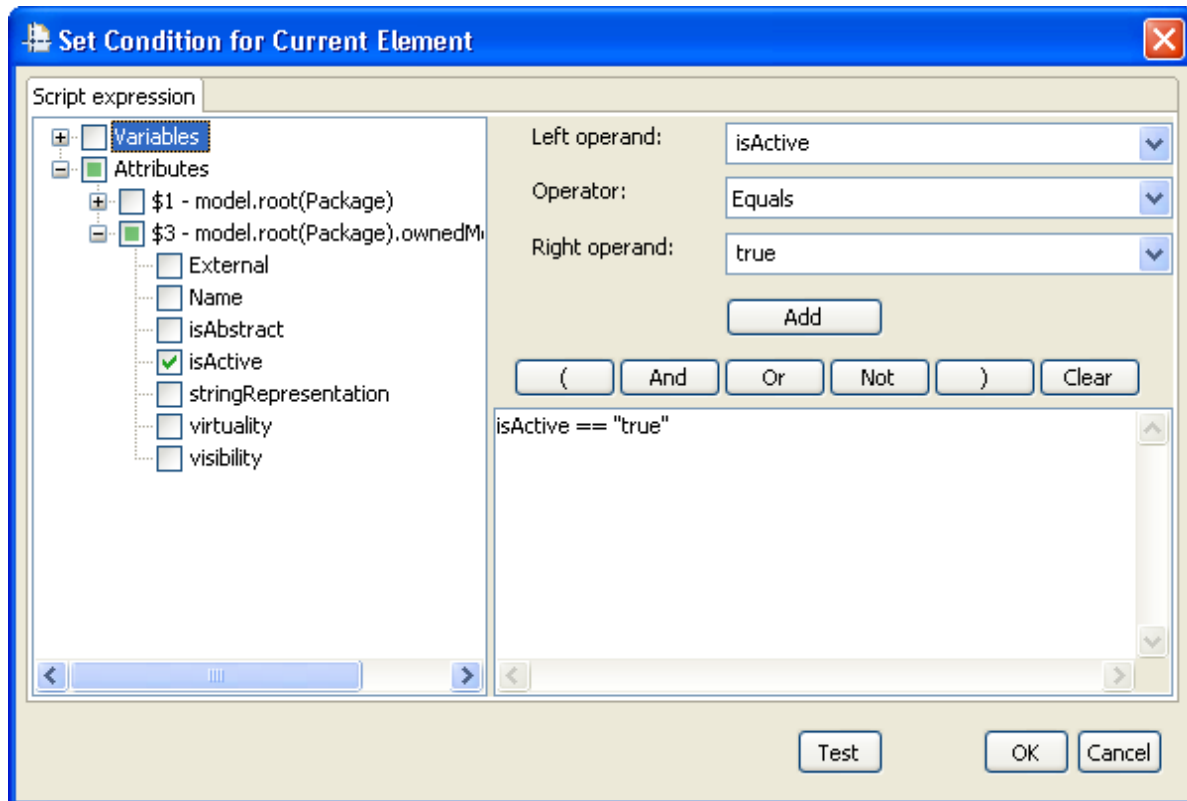


Figure 244

Generate the document

Open the launcher perspective. If the perspective icon is not available please follow the steps described for the DOORS Example.

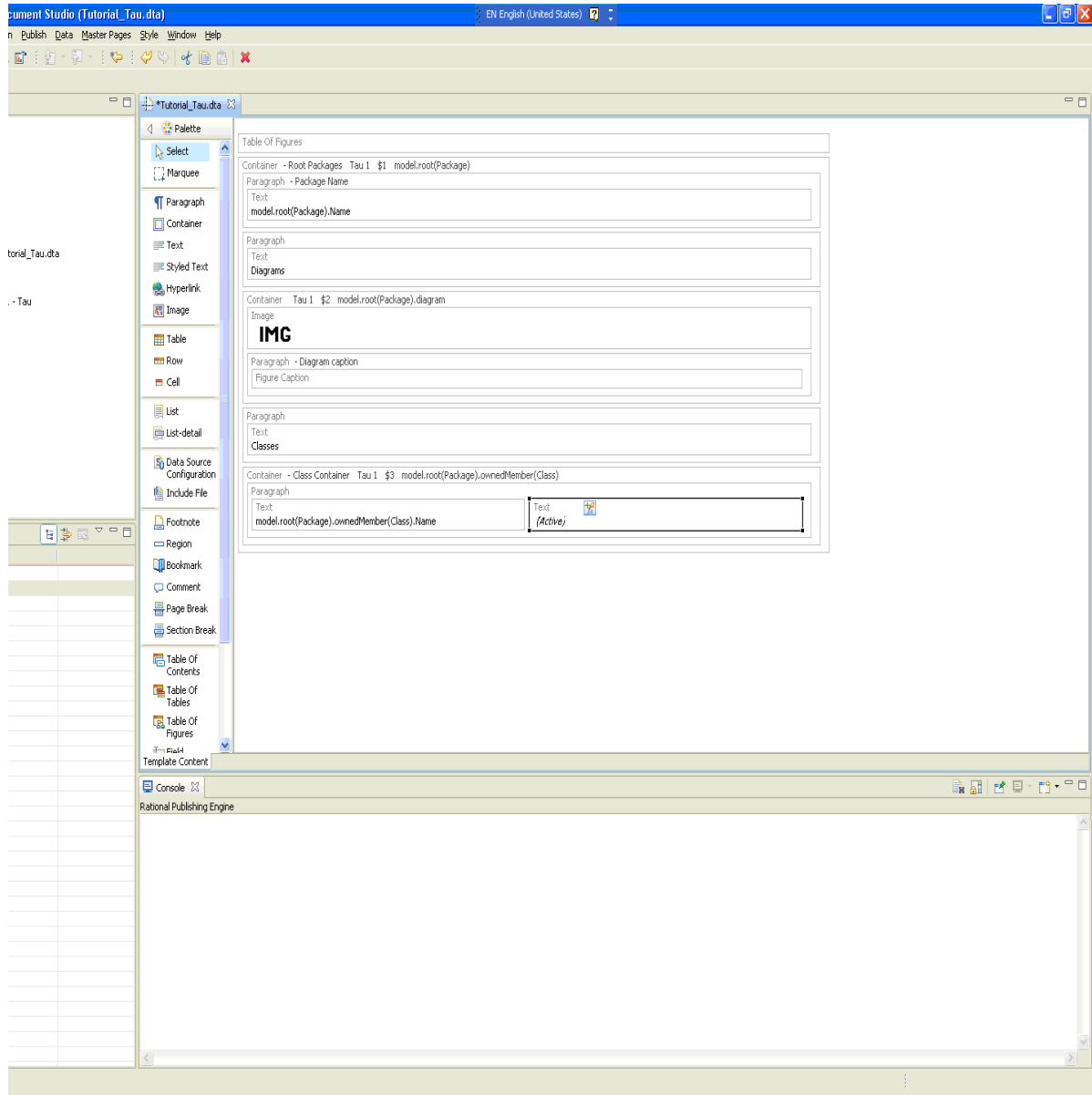


Figure 245

Select the Tau project (.tpp file) you wish to use as concrete data source as described in the Input Section. You can use one of the examples provided with Tau, which can be found in the examples directory below the Tau installation directory (on Windows the default location of the examples is *C:\Program Files\IBM\Rational\TAU\4.3\examples*):

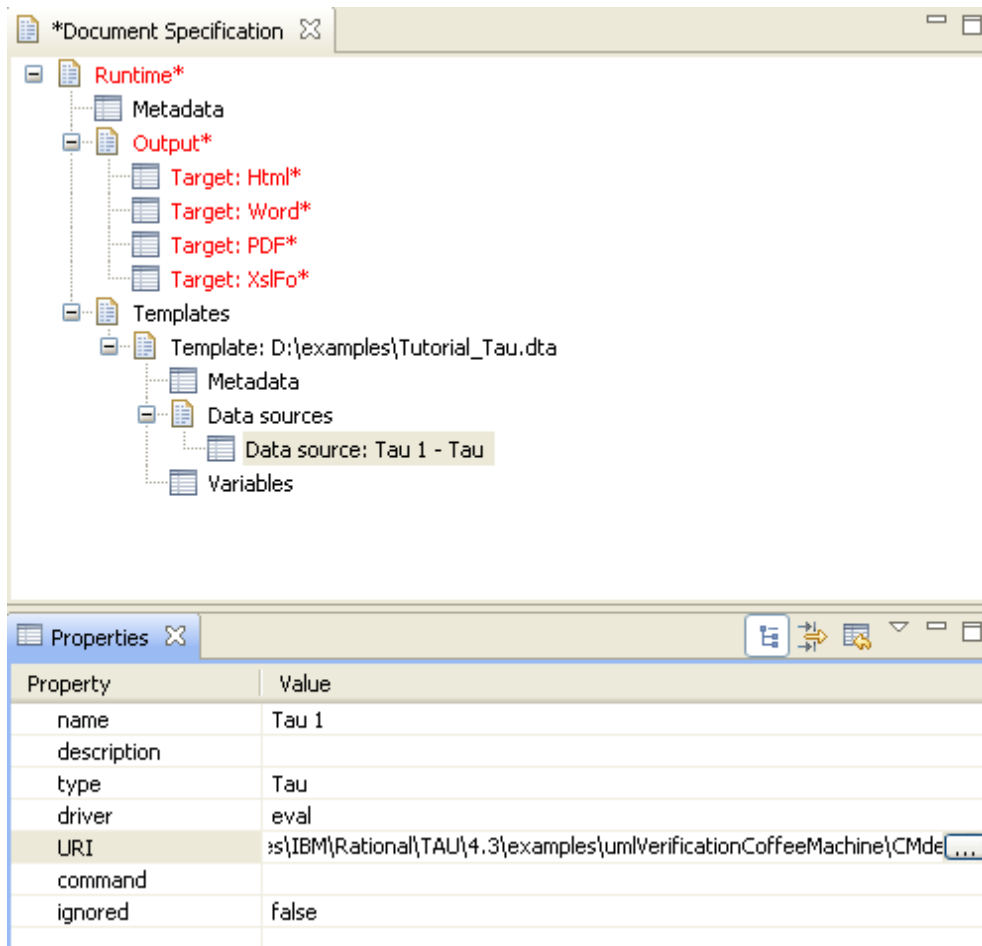


Figure 246

Start the document generation process. RPE will ask you to save the changes to the document template, if any unsaved changes exist.

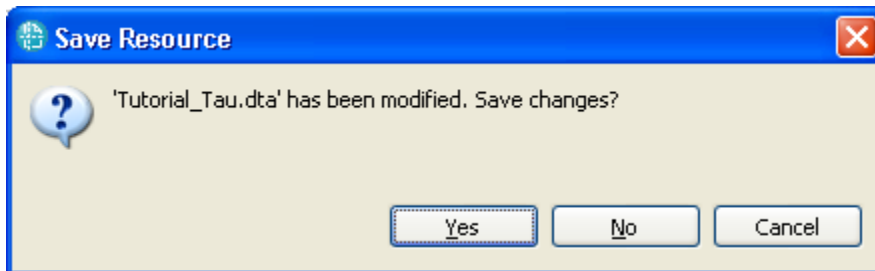


Figure 247

Unless you cancel the save operation, the document generation process will start

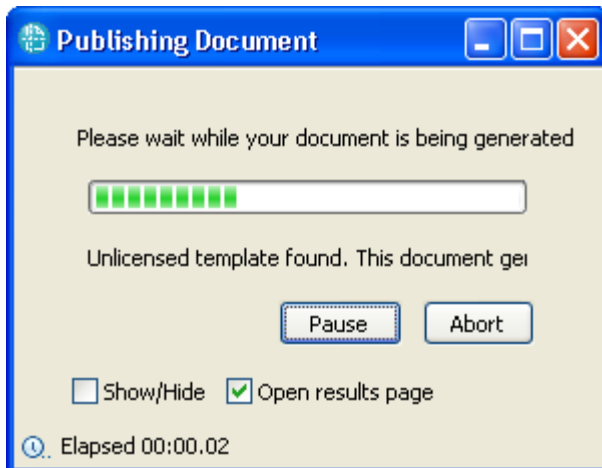


Figure 248

Once finished a window will be displayed allowing you to view/save the results

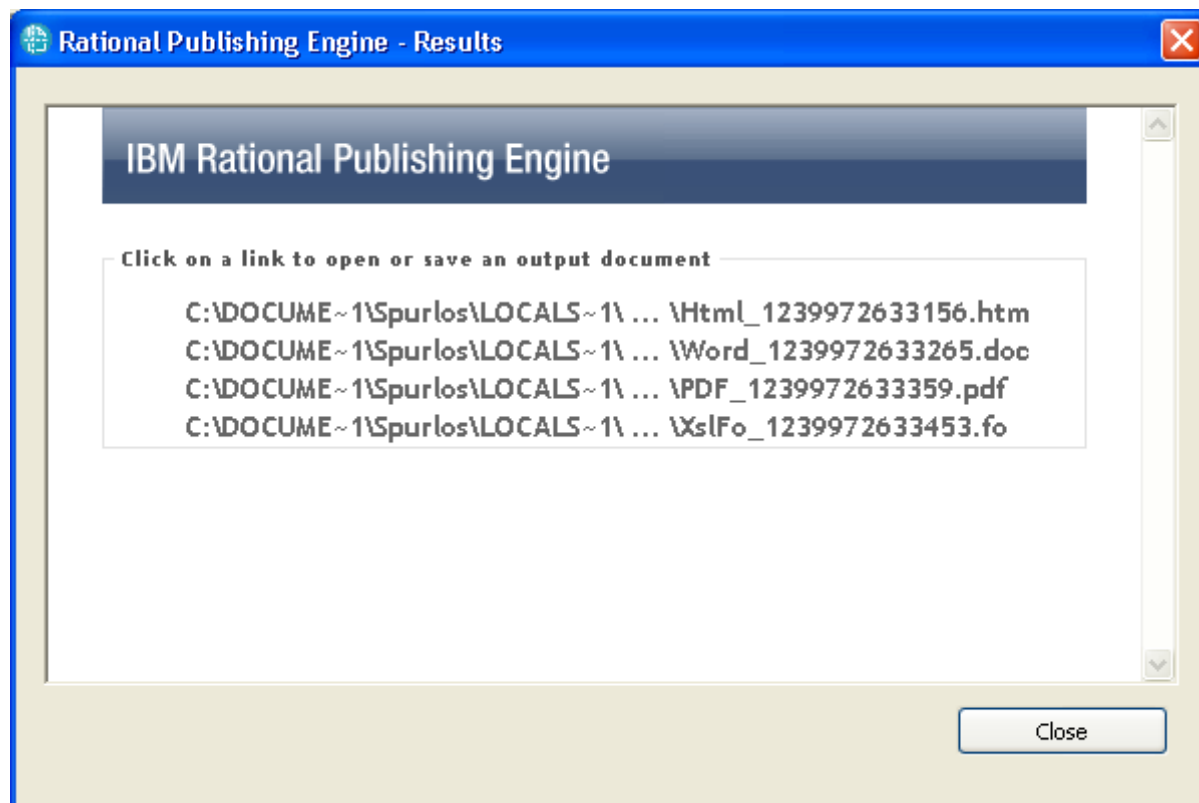


Figure 249

An example for REST Data

The example uses the Rational Insight Data Services and the adaptor for IBM Rational Requisite Pro. This example is also valid for XML data sources, the differences consisting in the lack of the Schema Discovery tool and Data Source Discovery tool for XML.

Obtain the schema

Create a new document and start the REST Schema Discovery.

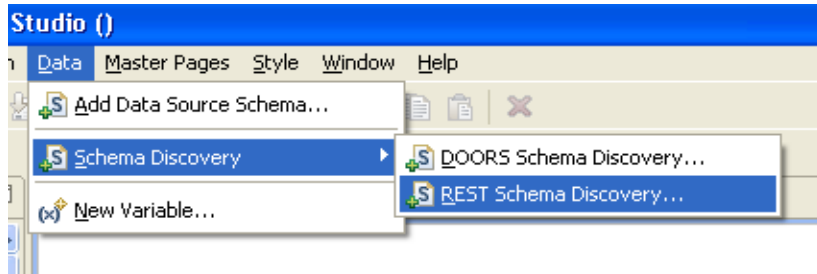


Figure 250

Once the schema discovery is started you need to provide the base URL for the Data Services along with any credentials.

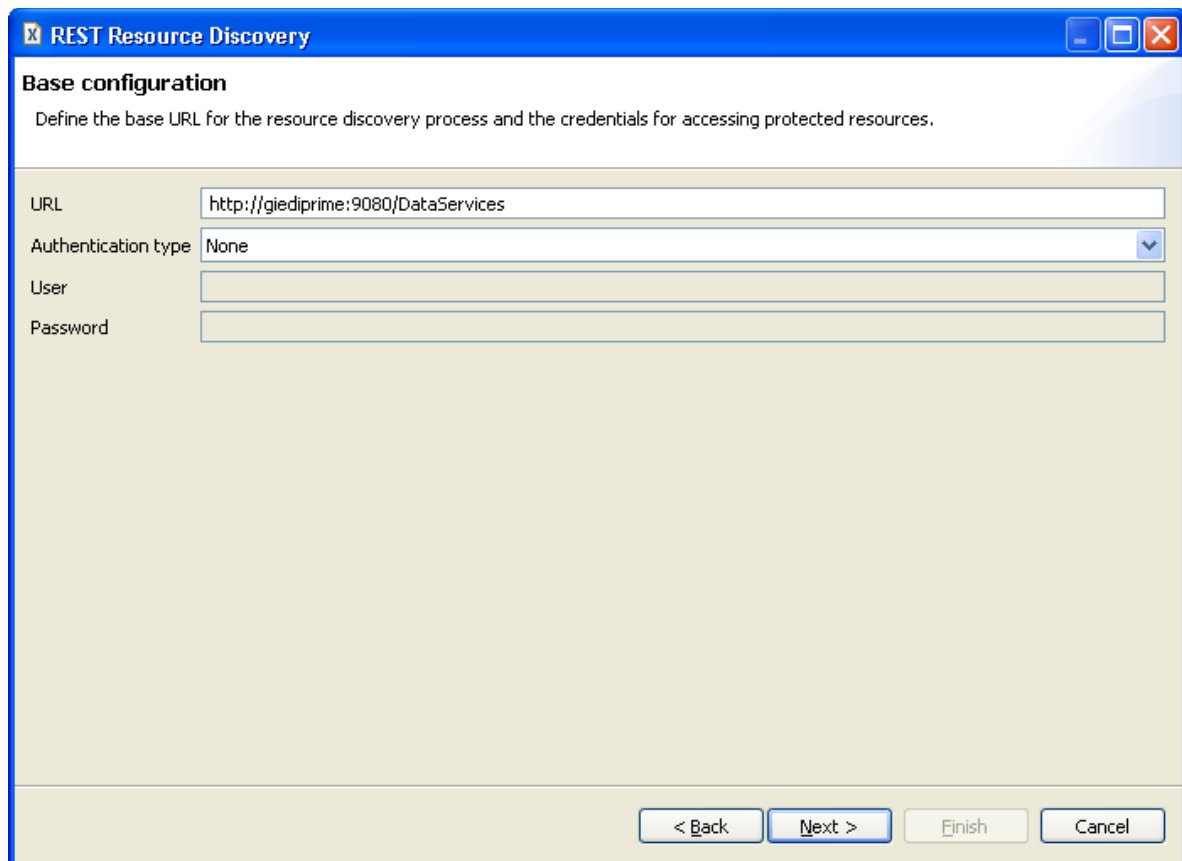


Figure 251

NOTE consult your system administrator for the URL of the Data Services.

Once the base URL and credentials are provided you can define the name of the data source, a description and you have the possibility to provide the Resource's URL directly or use the discovery wizard. Check the "Locate Using Data Services" and continue.

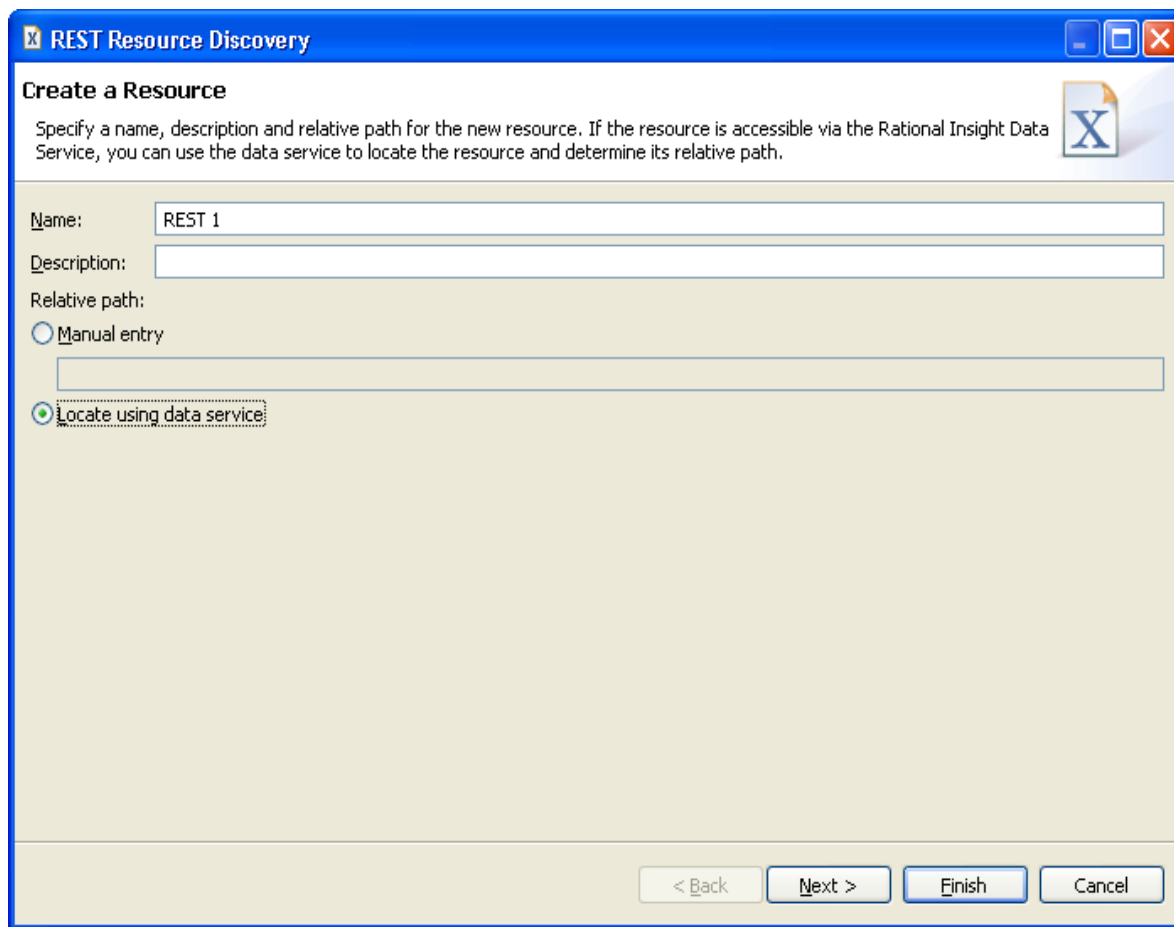


Figure 252

Now you need to navigate the resources until you find the desired one. In this example we will use the Packages resource provided by the RequisitePro adaptor.

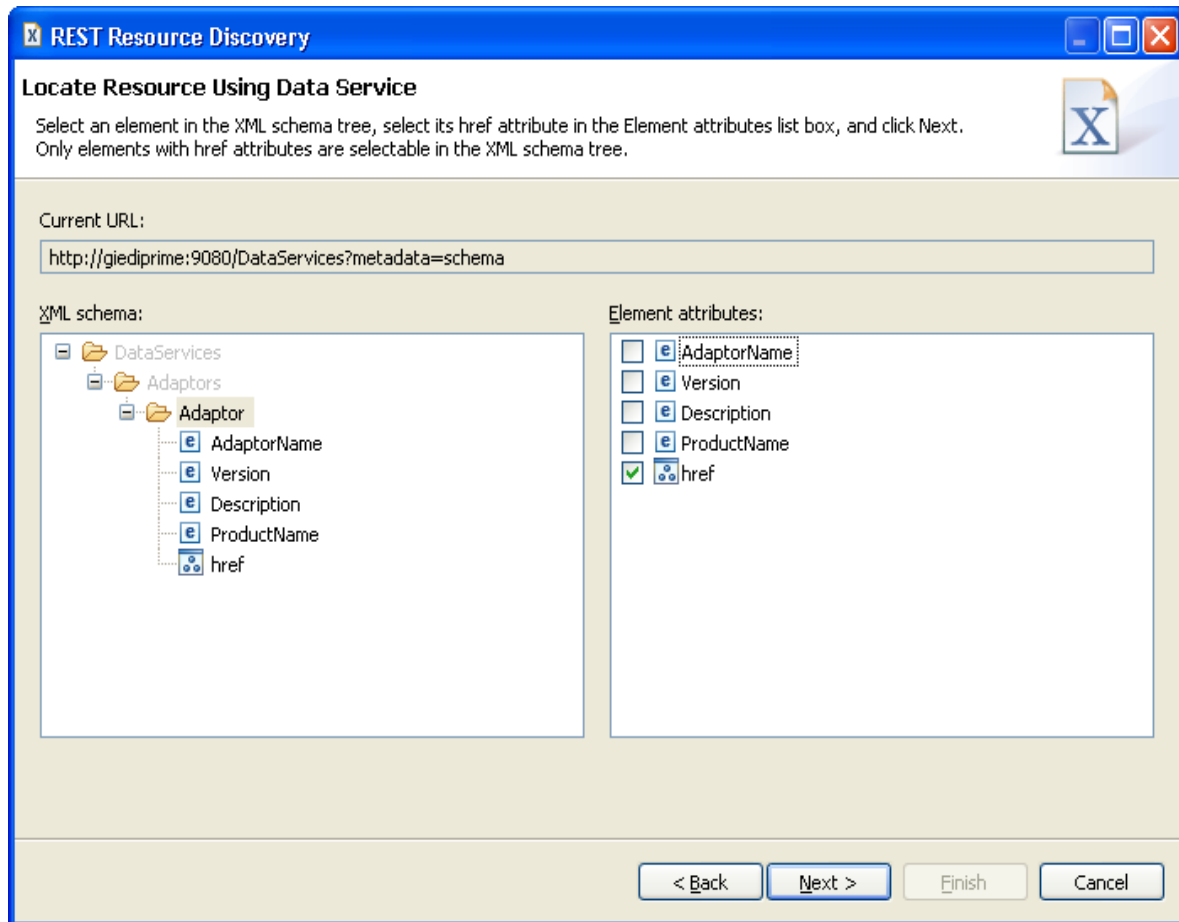


Figure 253

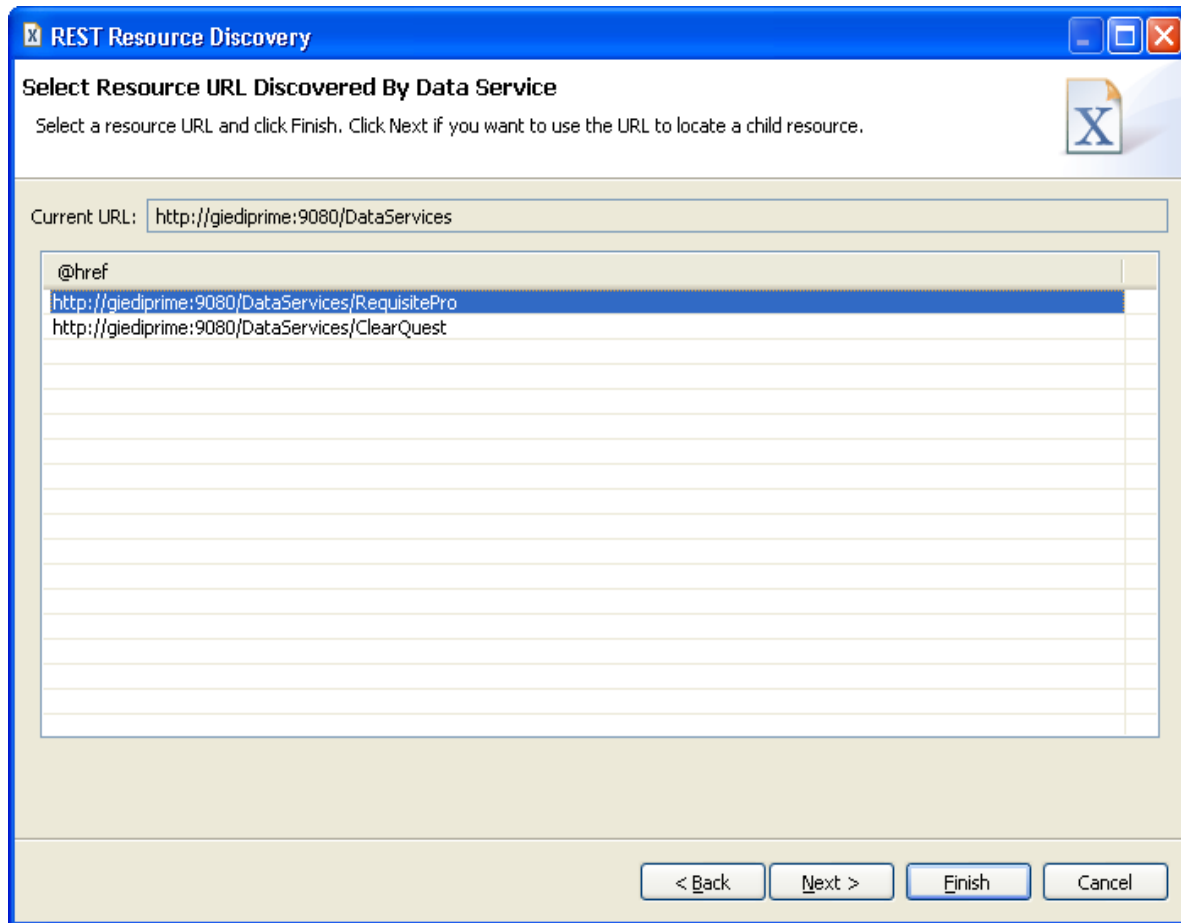


Figure 254

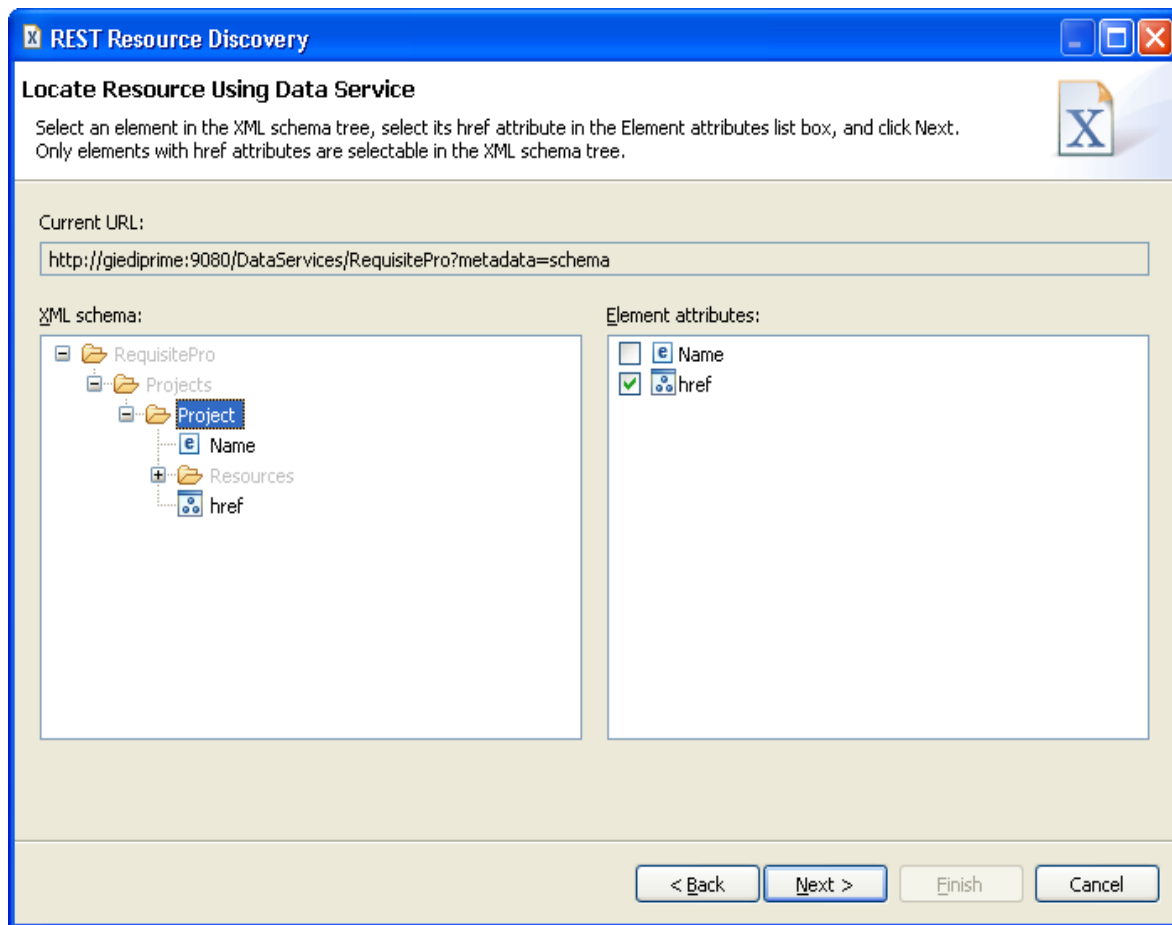


Figure 255

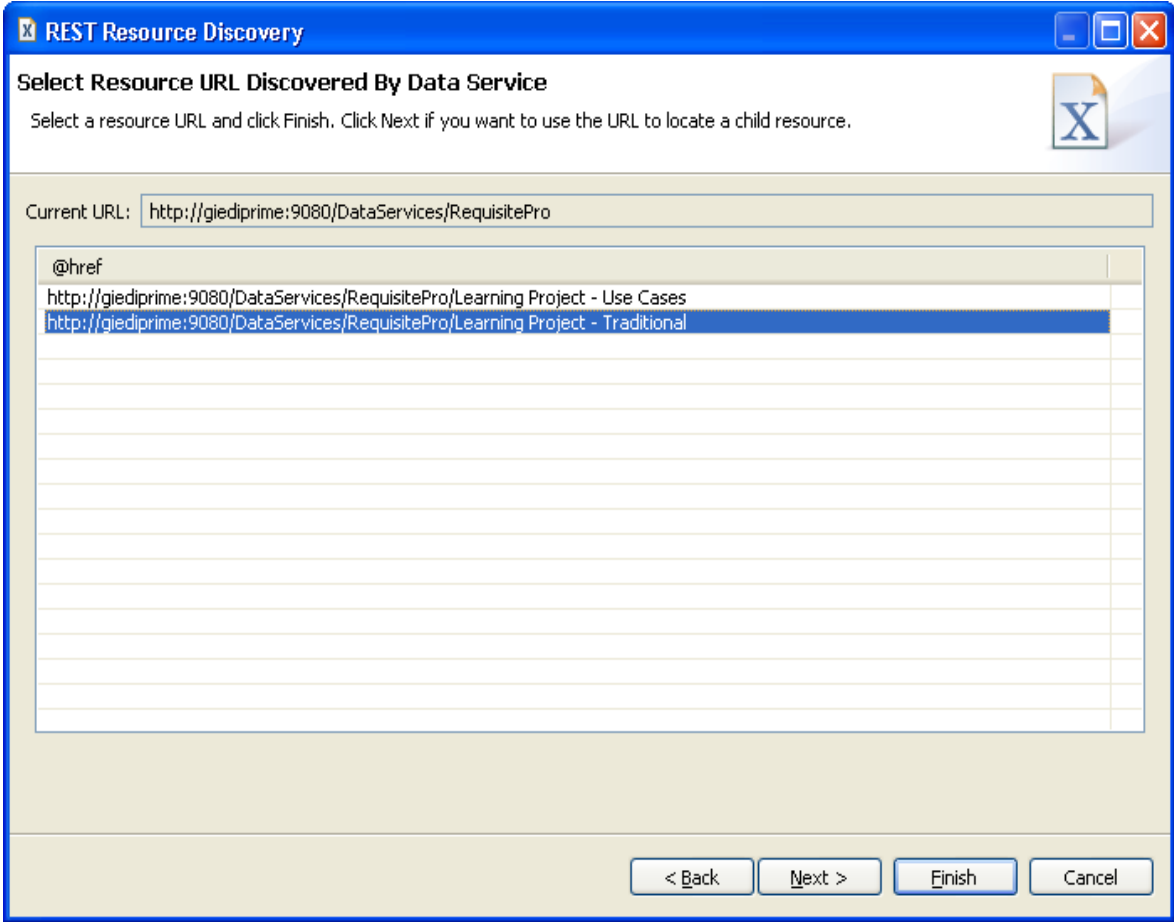


Figure 256

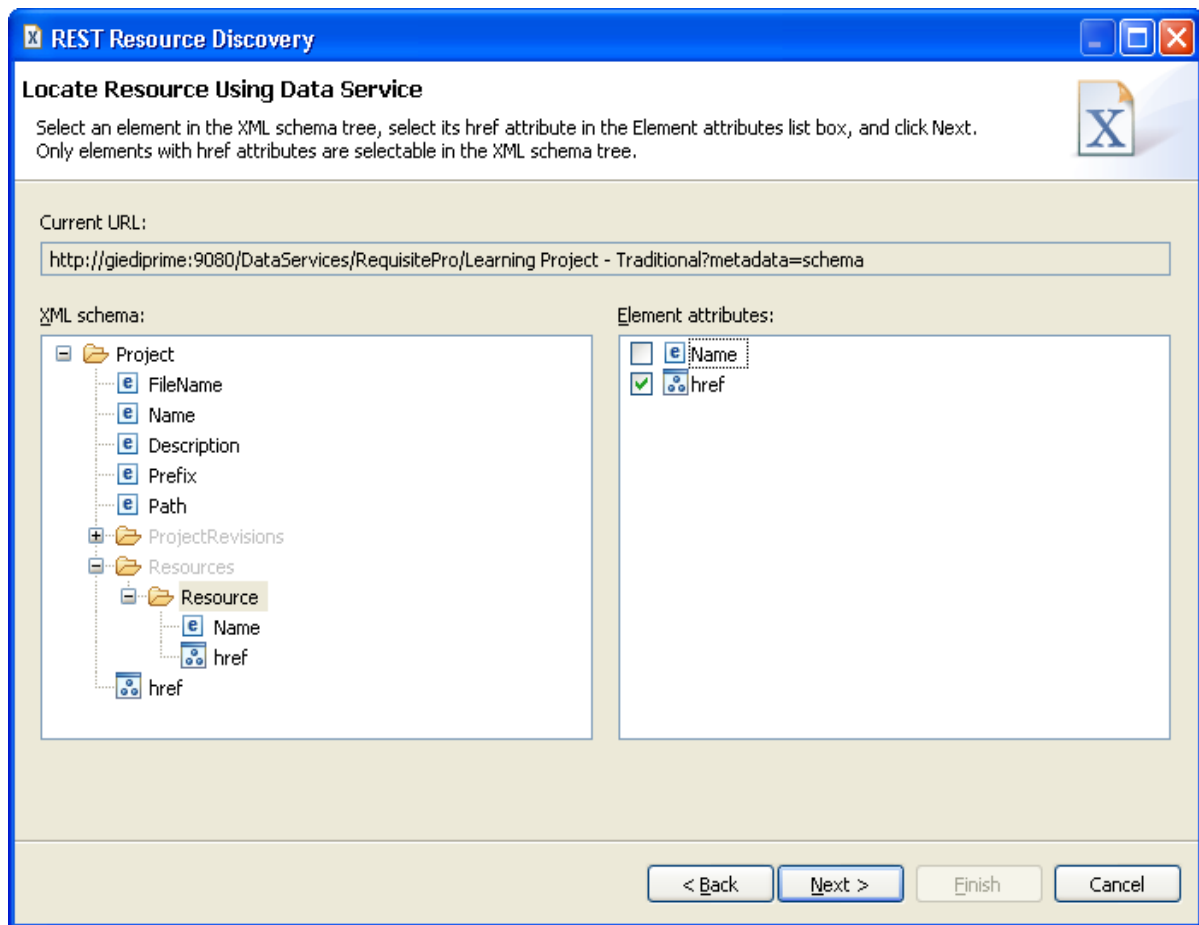


Figure 257

And finally the desired resource ...

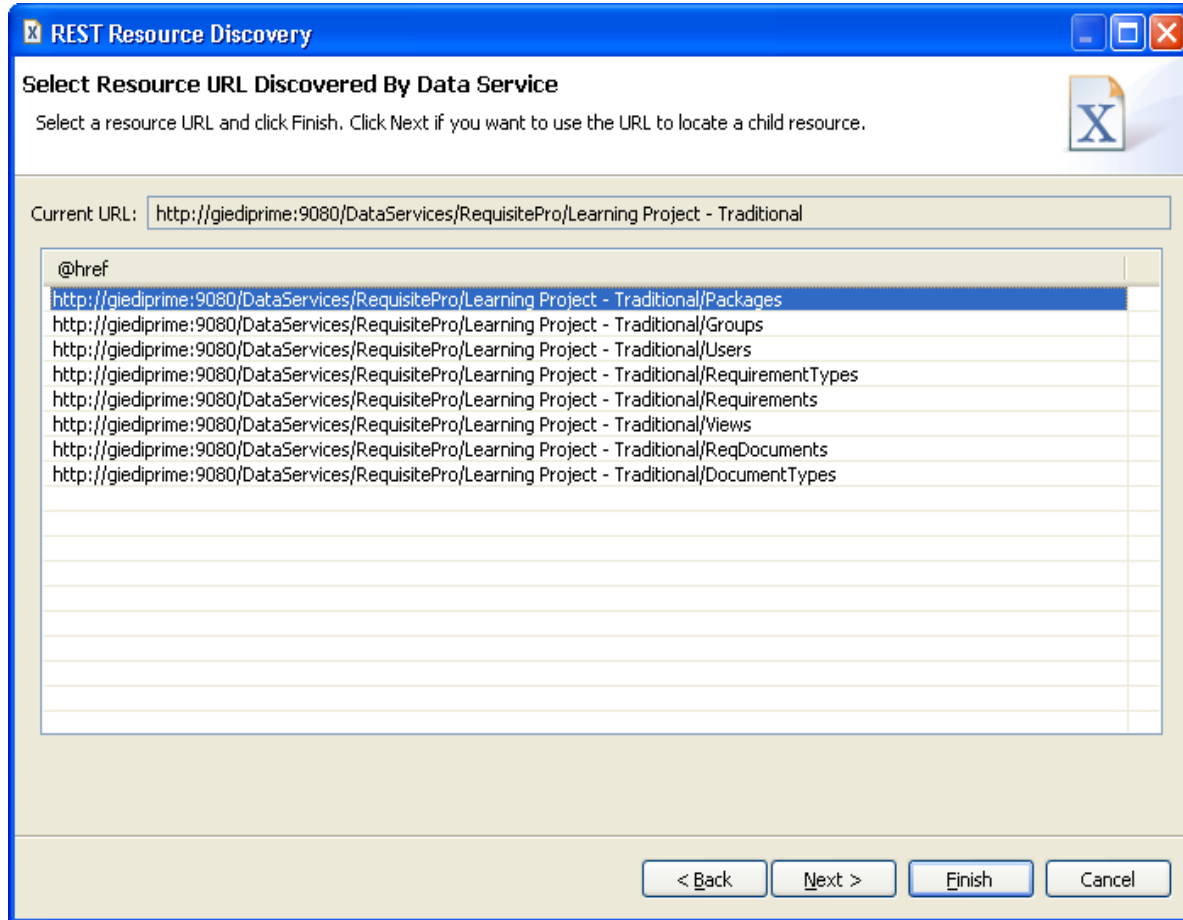


Figure 258

Click finish to have the schema added to the document and save the document template.

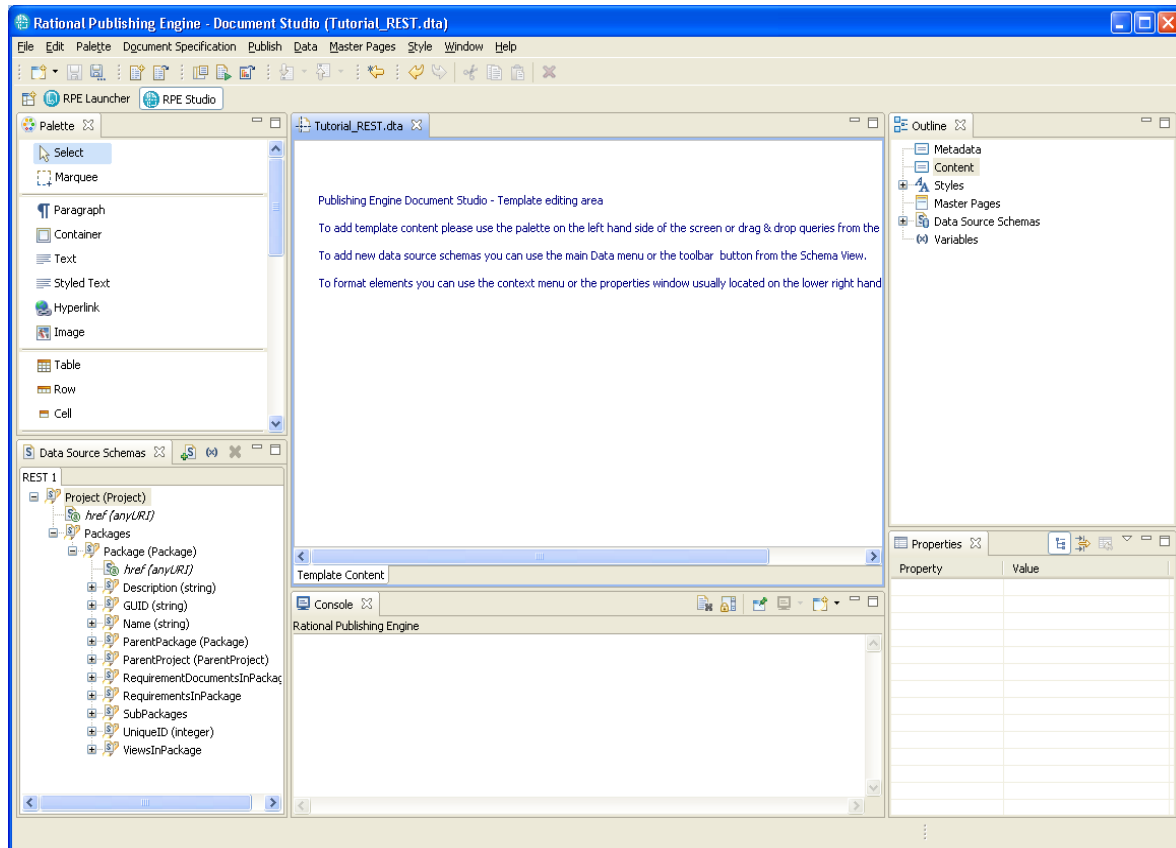


Figure 259

Create the template content

The template content will list the packages, their views and a few attributes from each. Start by adding a container element named “Packages Container” and drag onto it the “Project.Packages.Package” query.

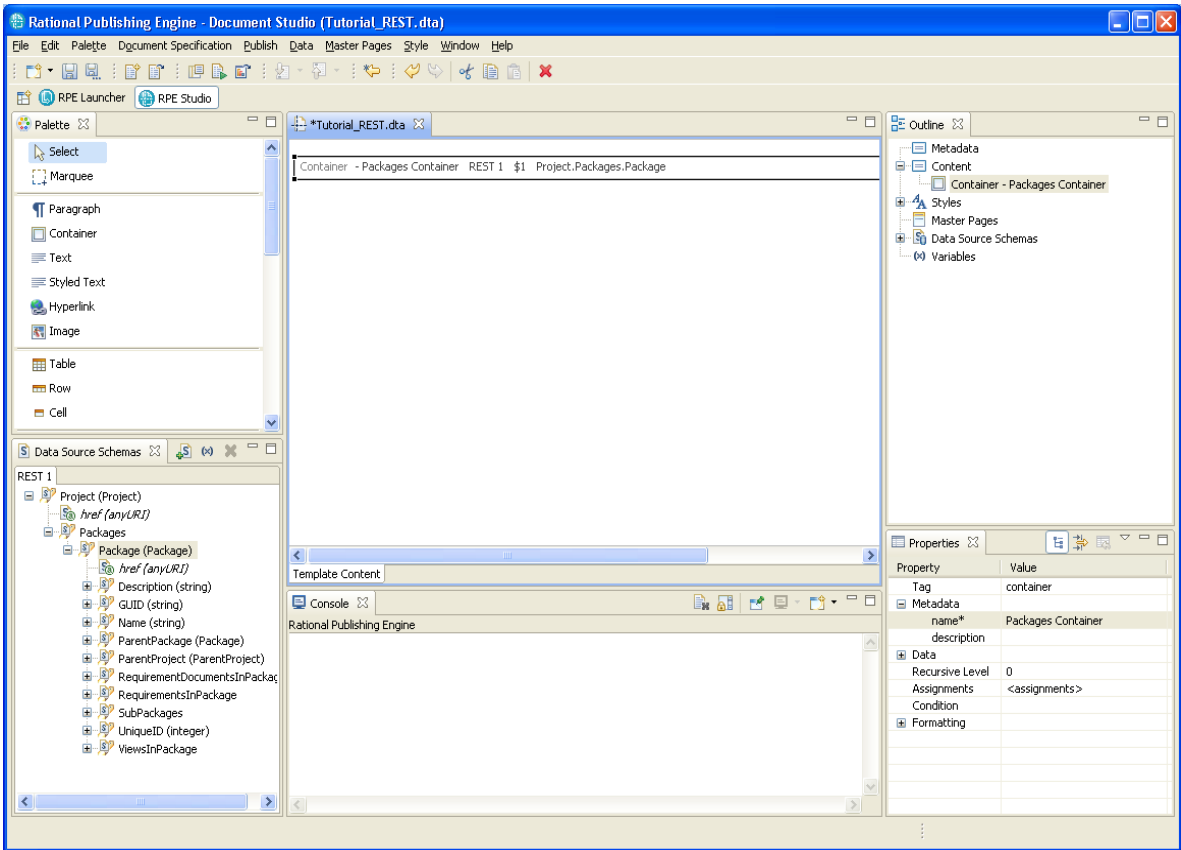


Figure 260

NOTE Save the document template

Inside the “Packages Container” put a paragraph element.

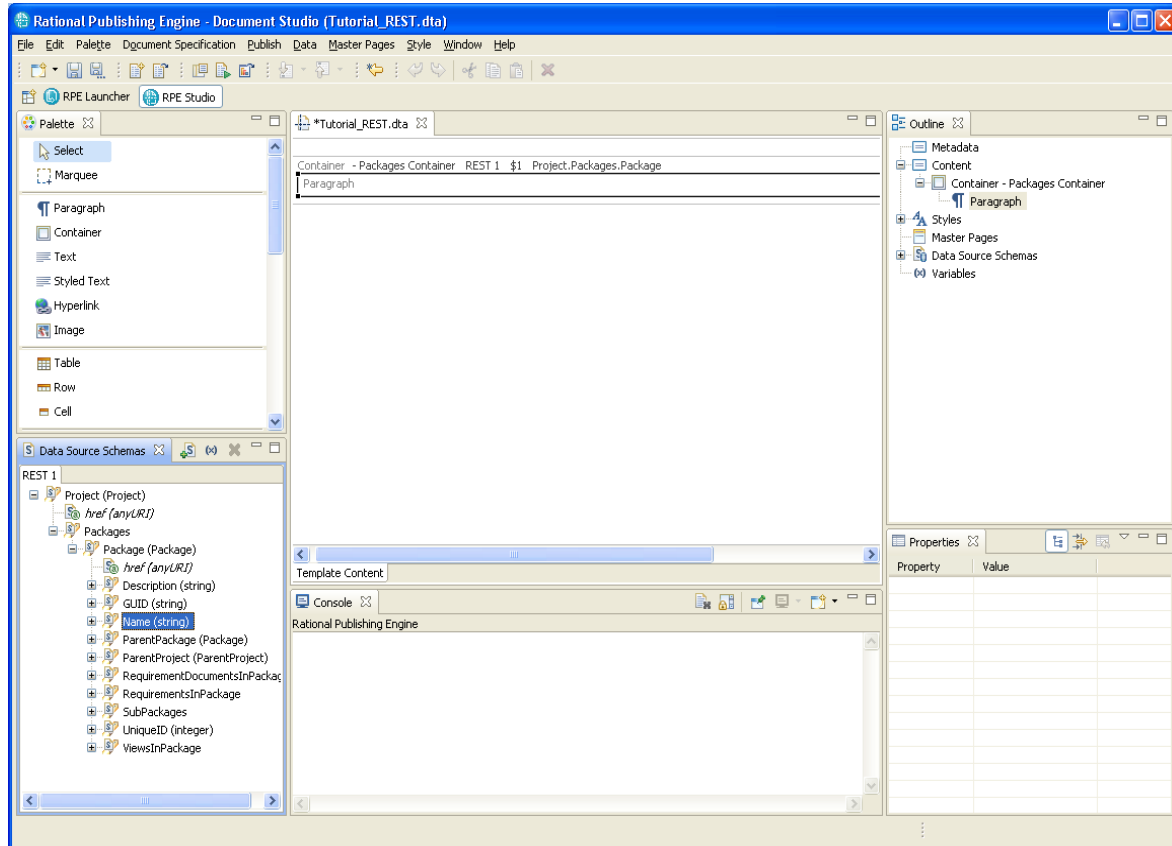


Figure 261

Now drag the “Name” element listed under Package in the schema view. RPE will ask how you want to use that element, as a query or as a value. Choose that you want to use it as a value.

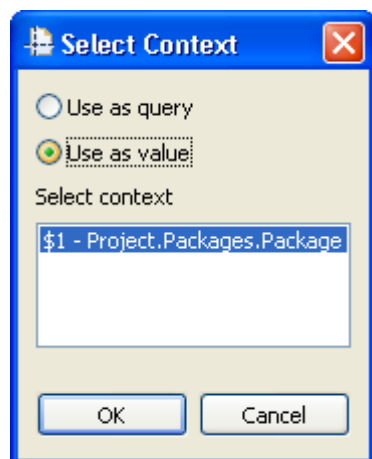


Figure 262

A new text element is created inside the paragraph and its content is set to the Name attribute.

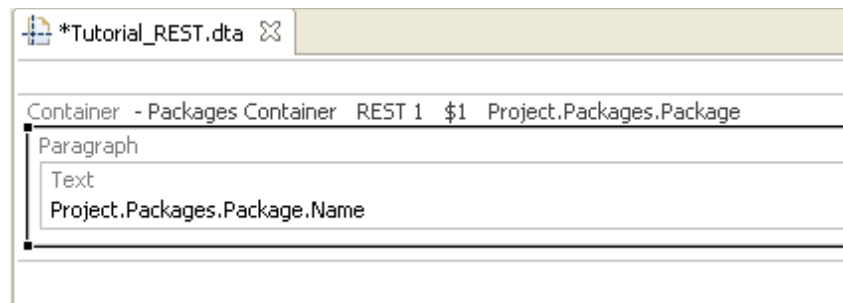


Figure 263

Add a new paragraph that contains the description for the package and format it as italic.

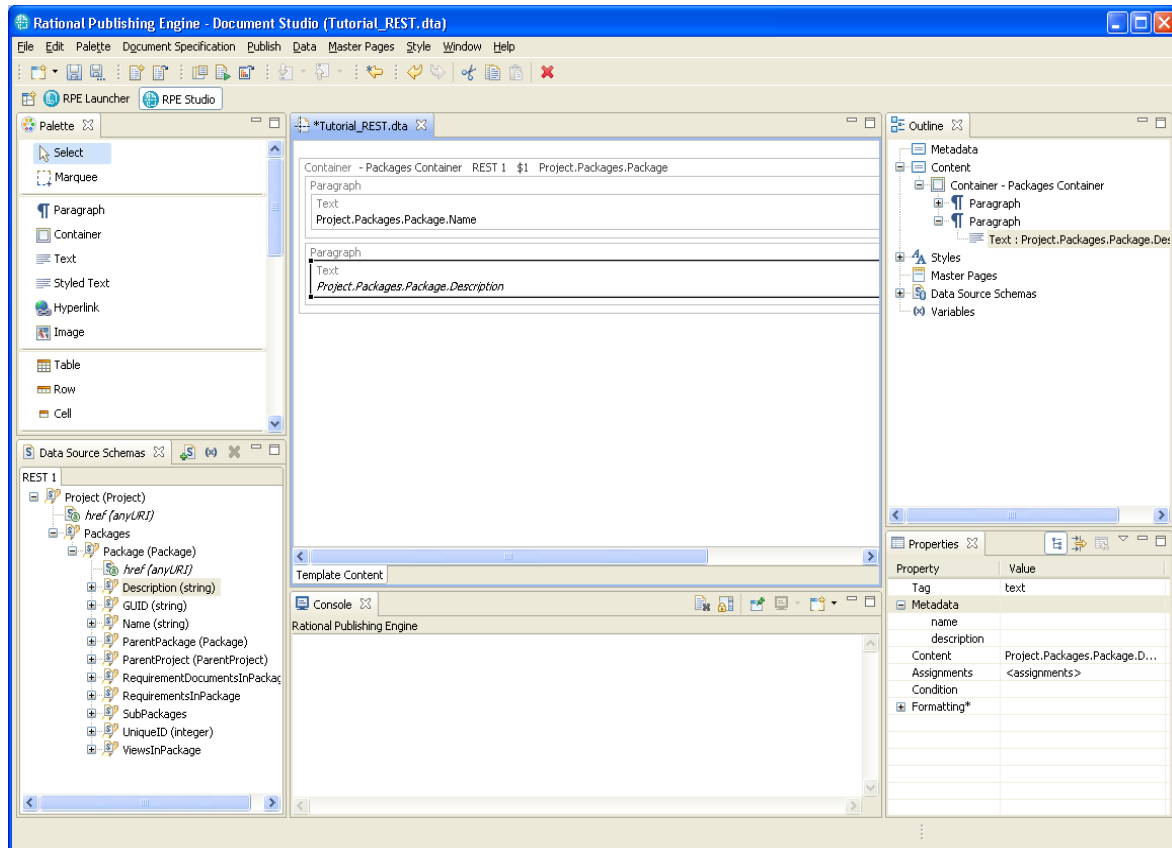


Figure 264

NOTE Save the document template

The next step is to list all the requirements from this package. Create list with a single item and add it to the template.

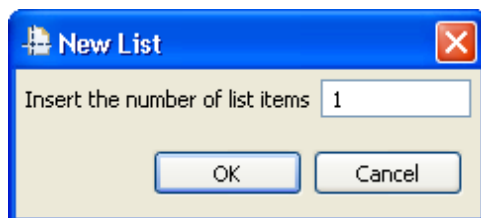


Figure 265

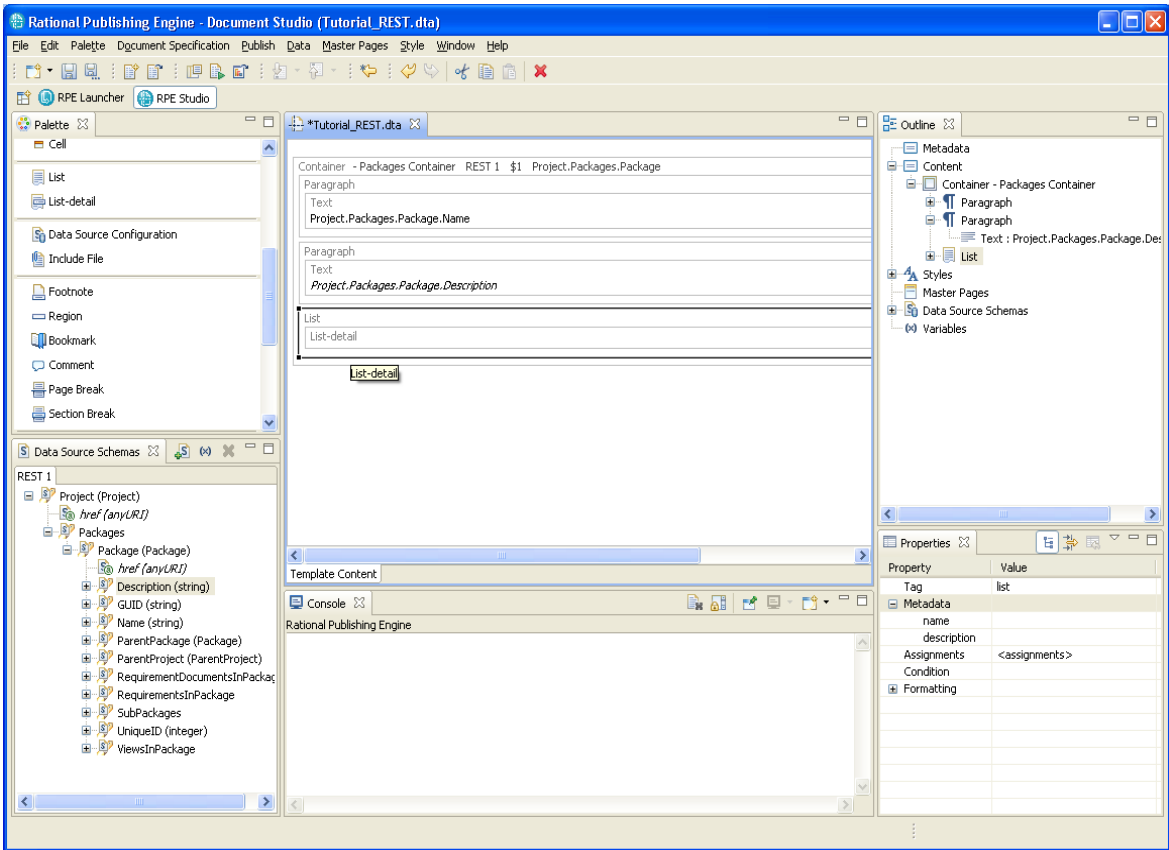


Figure 266

NOTE save the document template

Drag the RequirementsInPackage.Requirement element from the schema view onto the list detail.

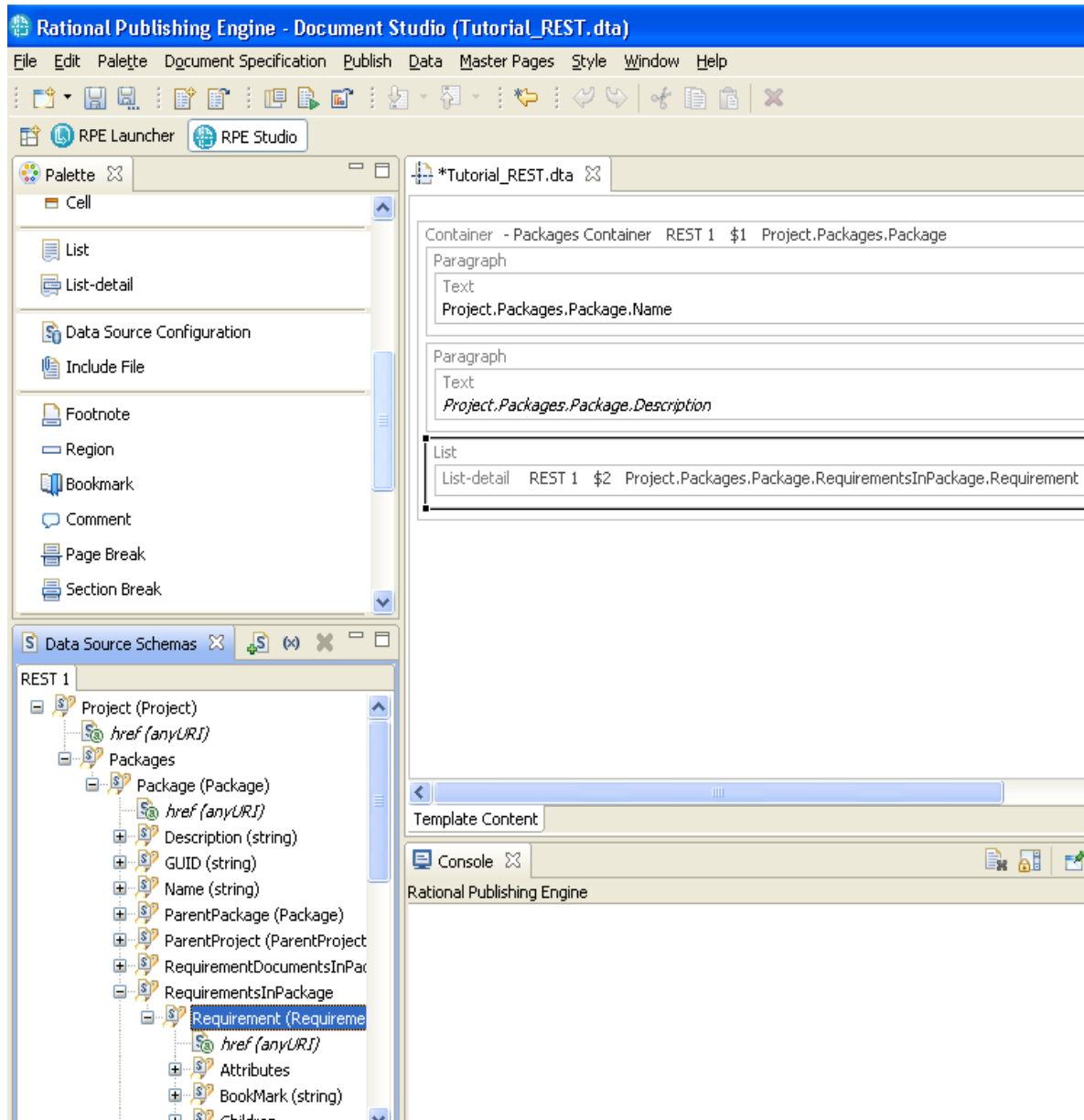


Figure 267

NOTE Remember to save the document template

In the list detail add 3 text elements. Onto the first set the content to be the “FullTag” attribute of the Requirement.

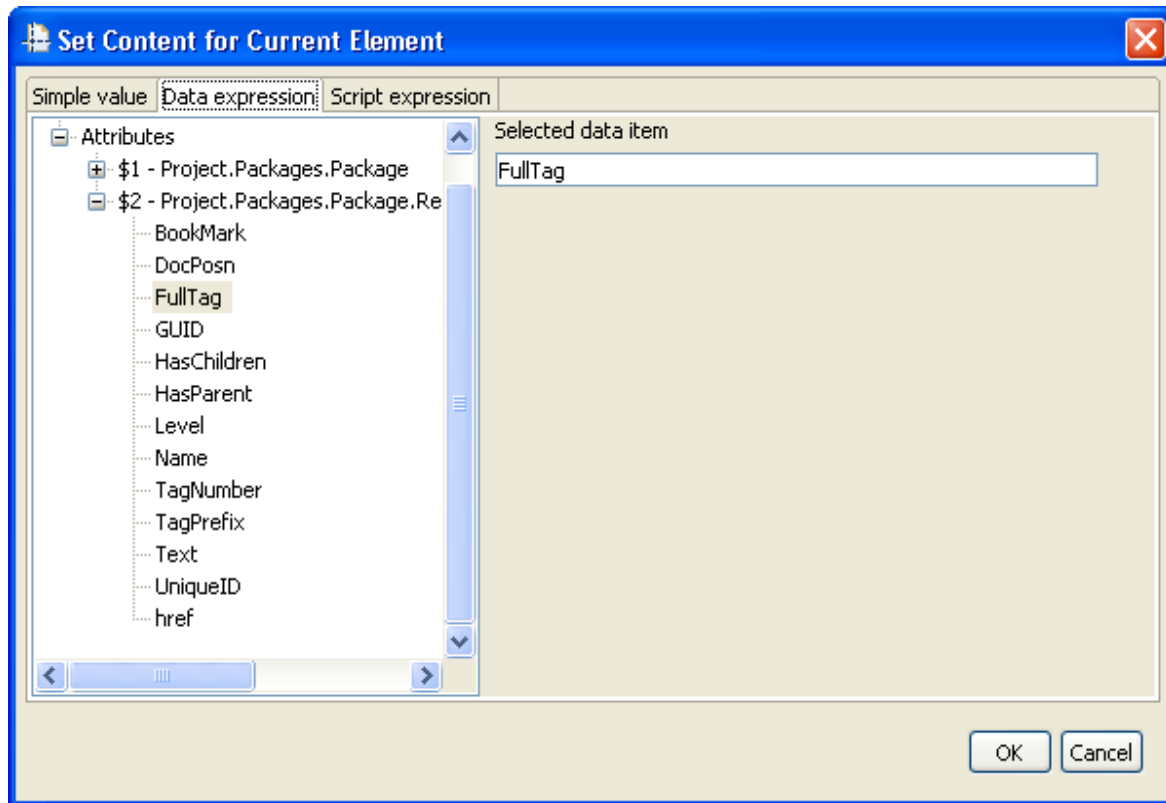


Figure 268

NOTE Remember to save the document template

For the second one set the content to be the static text “ – “

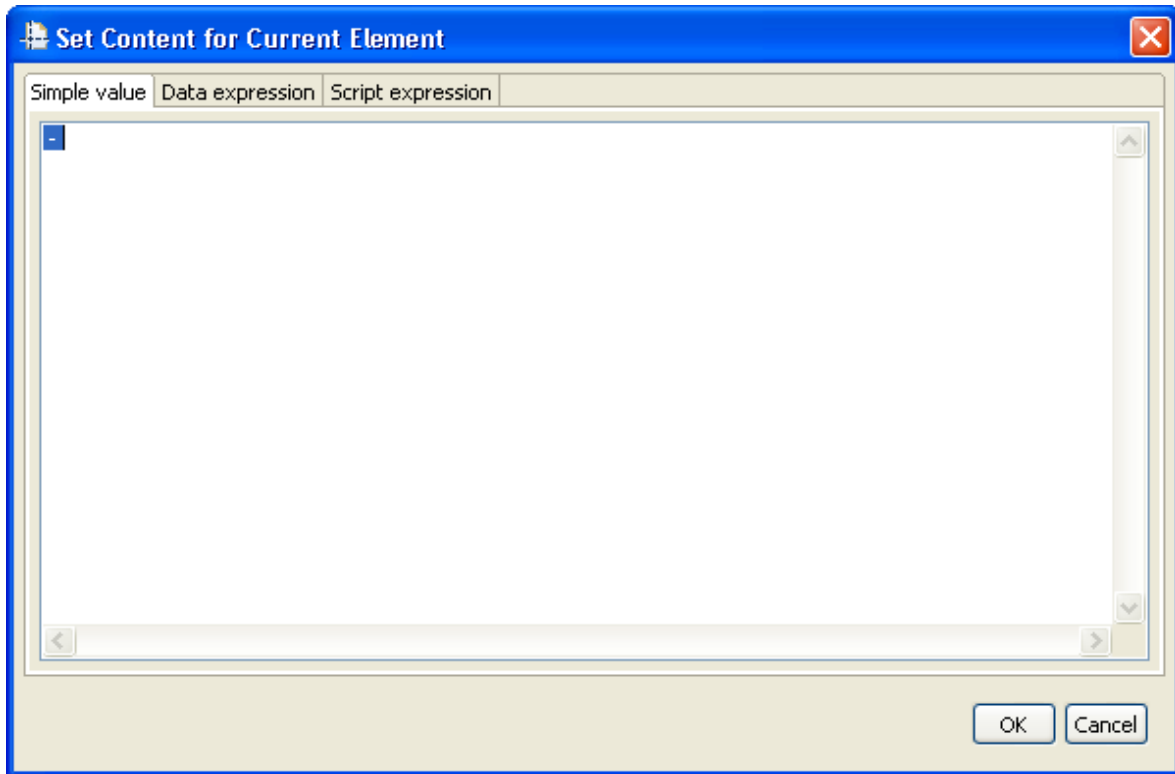


Figure 269

NOTE Remember to save the document template

For the 3rd one set the content to be the “Text” attribute of the requirement.

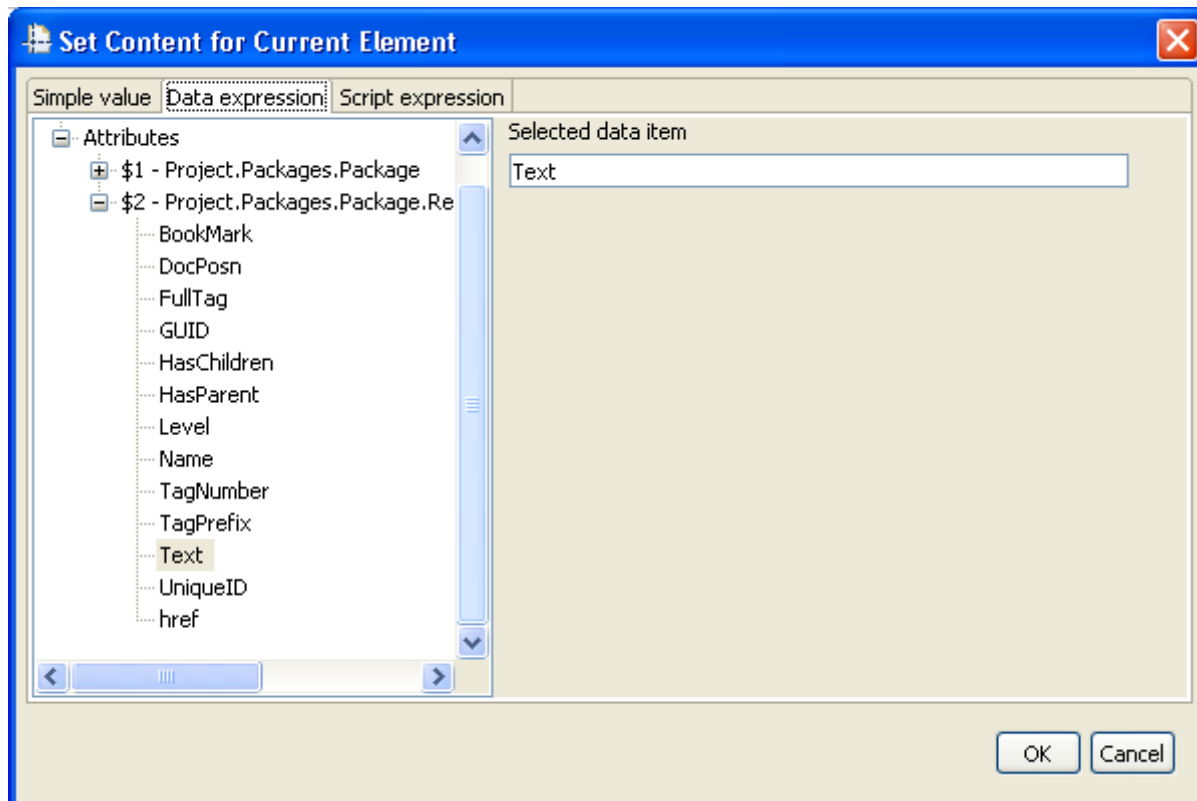


Figure 270

NOTE Remember to save the document template

The template should now look like this.

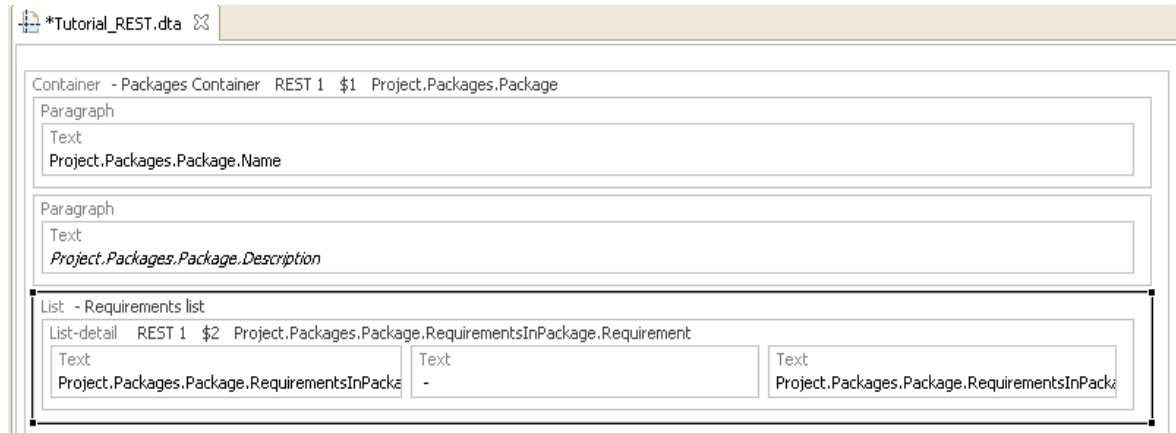


Figure 271

To embellish the template the following changes are performed:

- A Table of Contents is added to the top of the template

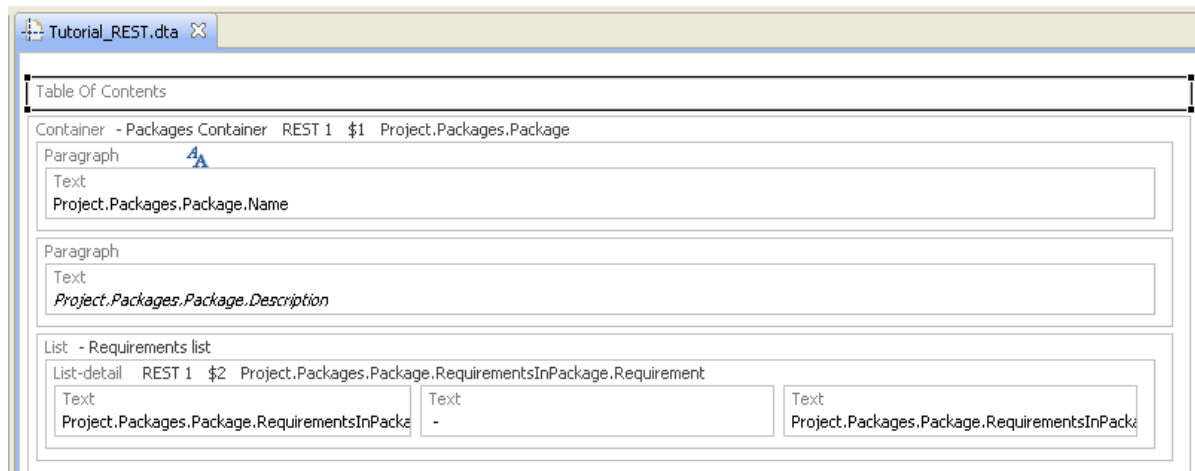


Figure 272

- The paragraph containing the package name is assigned style “1”

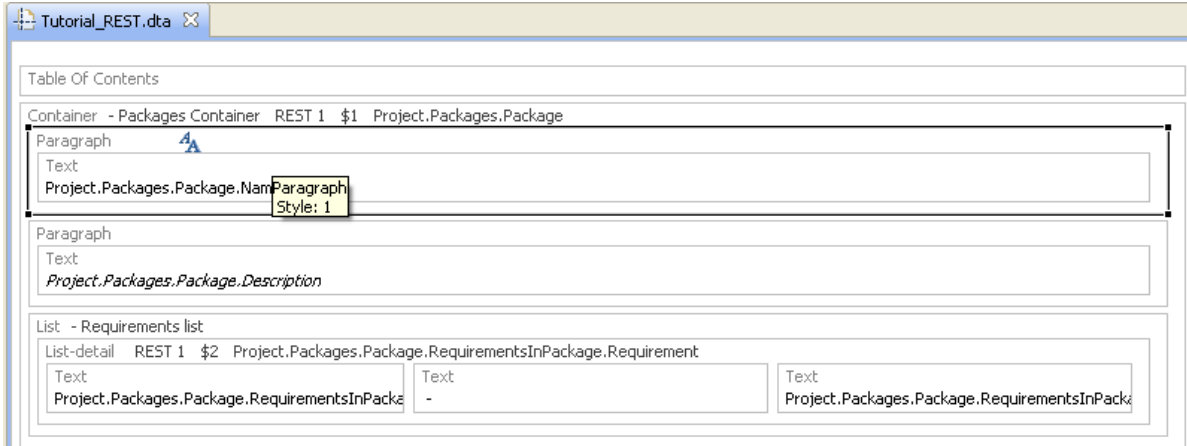


Figure 273

- A condition is set on the paragraph containing the documentation to exclude it from output if the description is empty

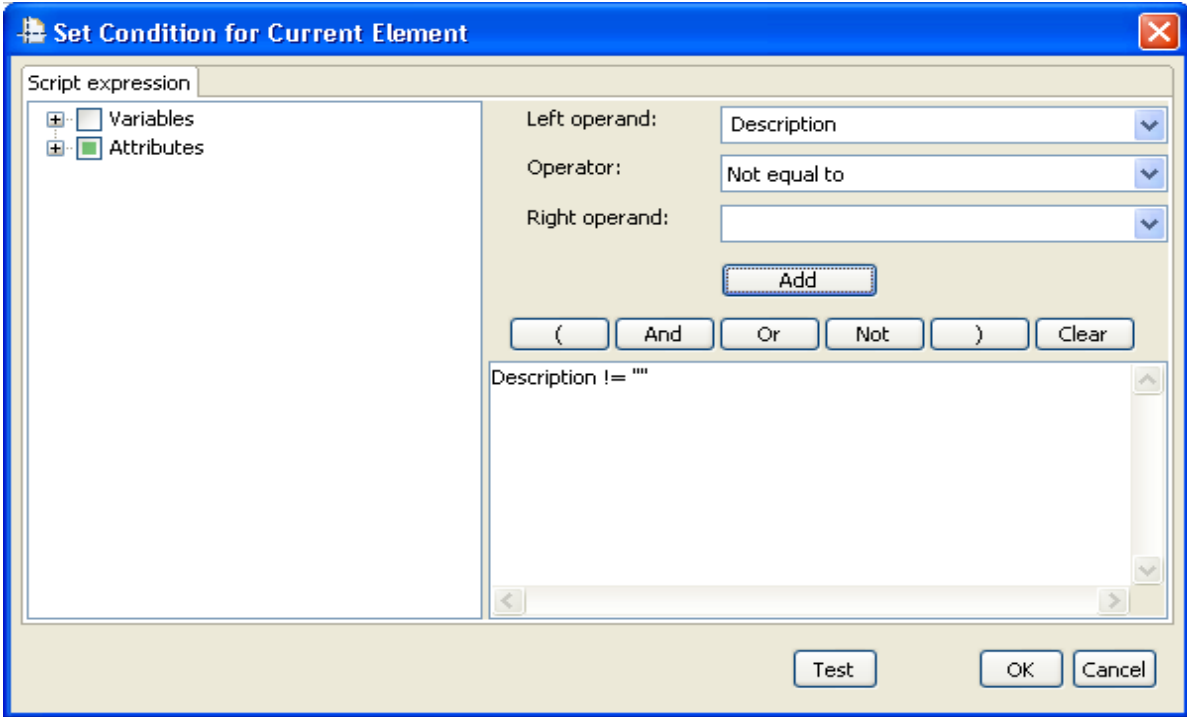


Figure 274

The text containing the requirement ID is replaced by a hyperlink having as content the href attribute of the requirement

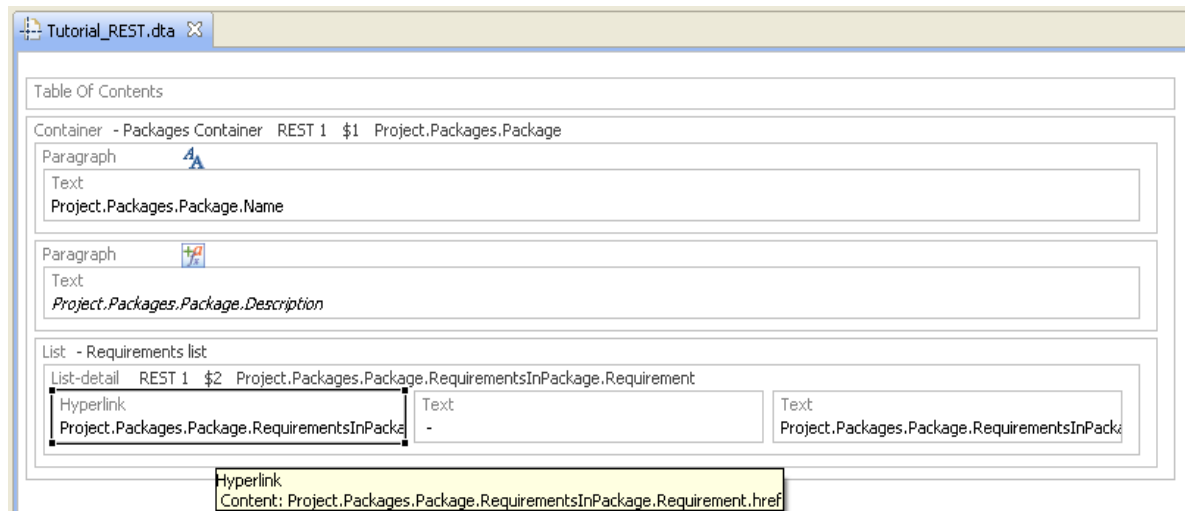


Figure 275

The “display” property of the hyperlink is set to be the “FullTag” attribute of the requirement.

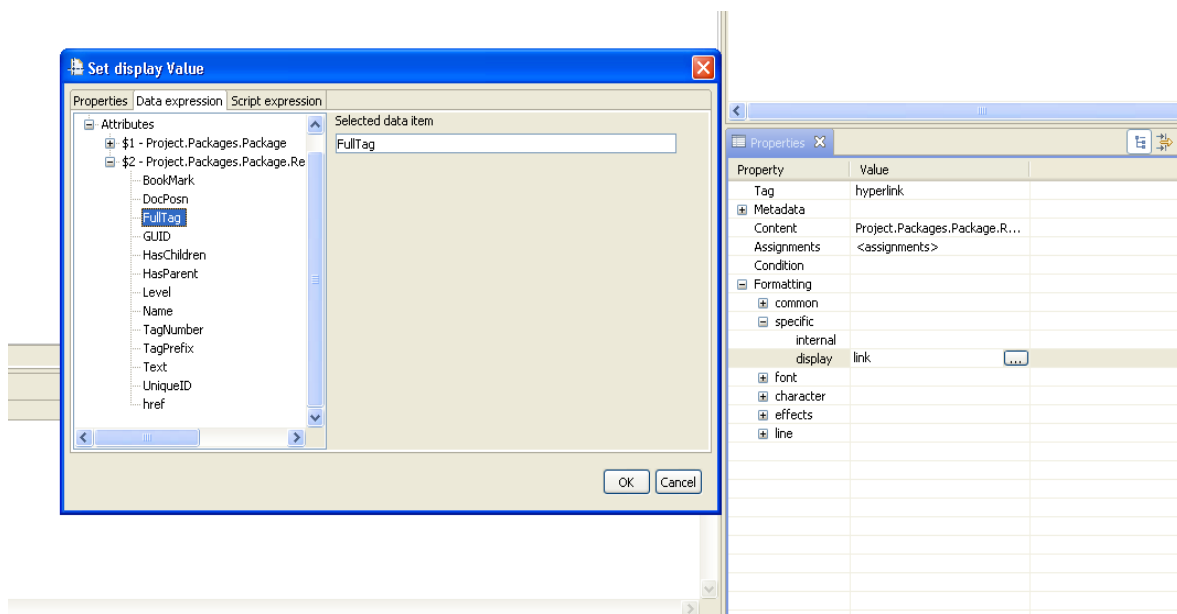


Figure 276

The predefined “Internal Hyperlink” style is applied to the hyperlink

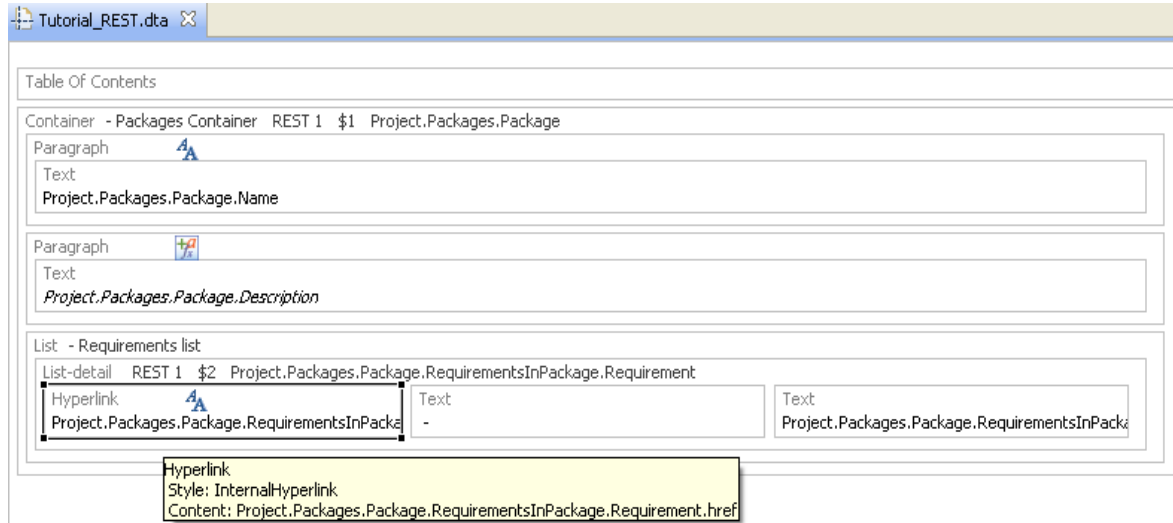


Figure 277

Add an image to the top of the template

NOTE Remember to save the document template

Configuring the data source

Open the Launcher Perspective. See the DOORS example for details on how to do it.

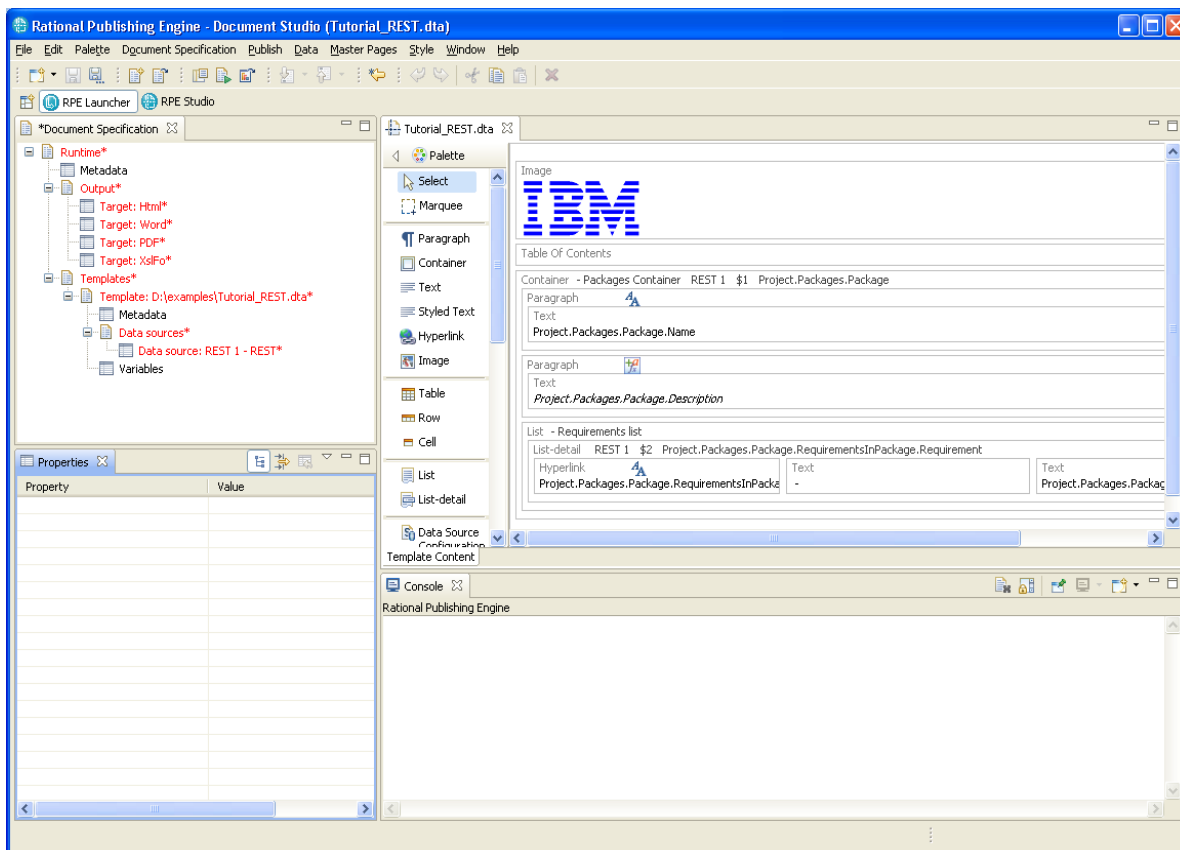


Figure 278

Use the “Configure Data Source” as described in the Input Section to obtain the URL of the data.

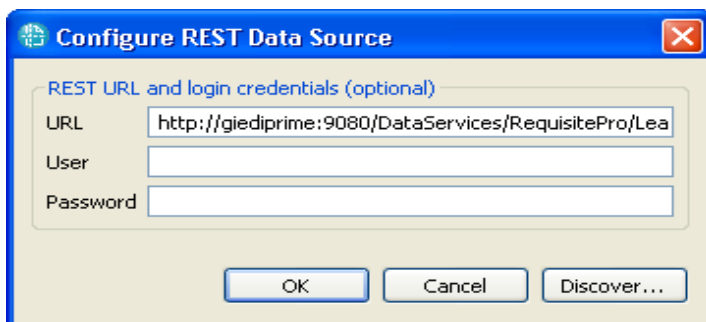


Figure 279

NOTE Add user name and password if the URL location is protected. The user name and password you've used in the resource discovery wizard can be used here as well.

Generate the document

Start the document generation process. If the template contains changes, RPE will prompt for saving the document template.

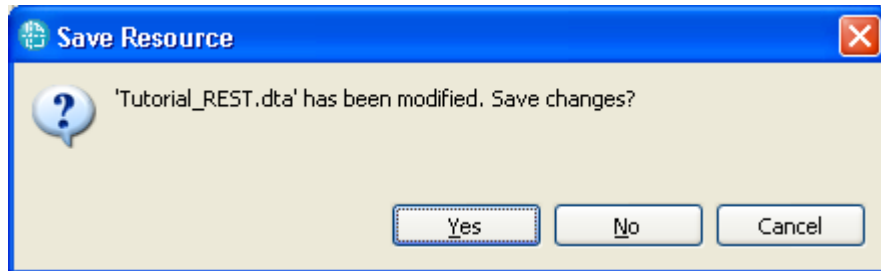


Figure 280

Once the document is saved the document generation process will start.

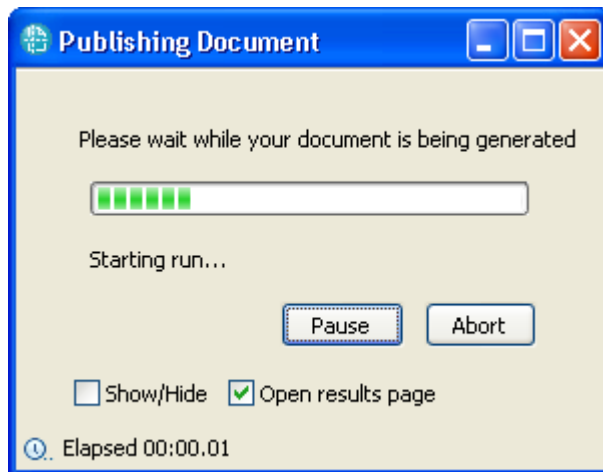


Figure 281

When the document generation is complete, the results window is displayed.

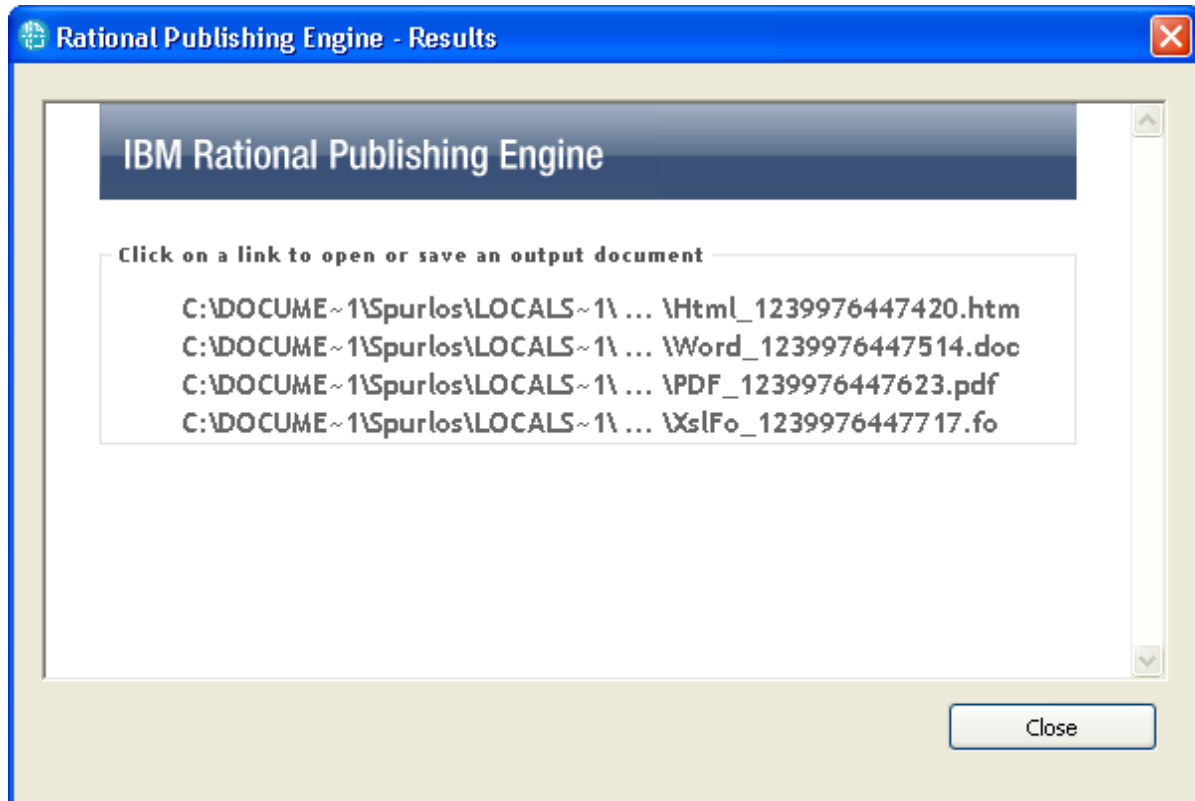


Figure 282

NOTE When RPE generates documents from REST data sources, it will log the URLs it uses. You can copy those URLs from the log and use them in a browser to see the data content.

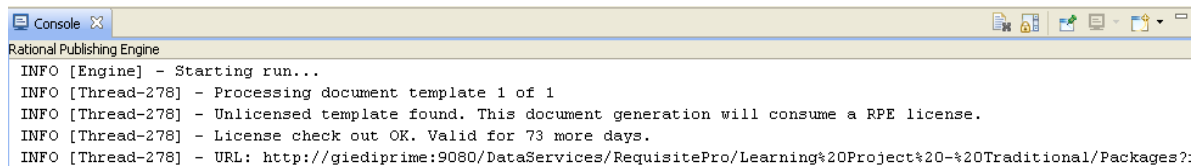


Figure 283

Please note that the URLs are encoded so they might contain strange characters.

```
http://giediprime:9080/DataServices/RequisitePro/Learning%20Project%20-
%20Traditional/Packages?fields=Project/Packages/Package/
(Name%7cDescription%7cRequirementsInPackage/Requirement/
(Text%7cFullTag%7c@href))
```

However, once the URL is loaded in the browser it will become easier to read.

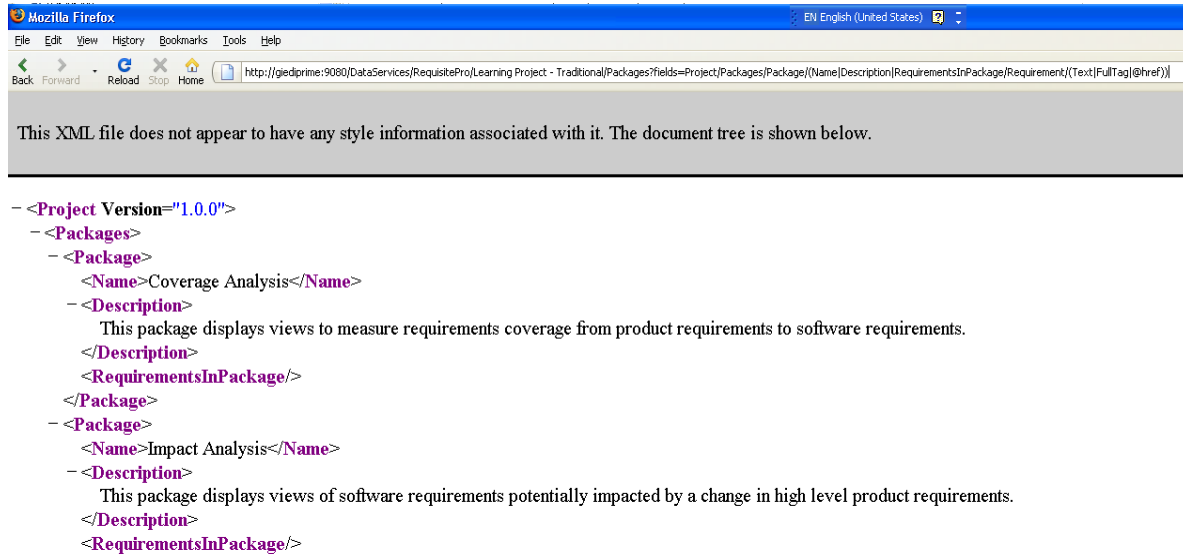


Figure 284

Annex

Library definition file

The library definition file is a simple hierarchical structure of references to templates. Each template is defined in a resource and resources can be grouped in containers.

For each template the URL must be provided and optionally the description.

For each container a label must be provided.

```

- <library>
  - <container>
    <label>Sample shared library</label>
    - <resource>
      <url>http://myhost/rpe/templates/template1.dta</url>
      <description>Sample template for RequisitePRO</description>
    </resource>
    - <resource>
      <url>http://myhost/rpe/templates/template2.dta</url>
      <description>Sample template for DOORS Data</description>
    </resource>
  - <container>
    <label>User defined template</label>
    - <resource>
      <url>http://myhost/rpe/templates/template2.dta</url>
      <description>My first template</description>
    </resource>
    - <resource>
      <url>http://someotherhost/XH4qLFX</url>
      <description>Test template stored in some CVS.</description>
    </resource>
  </container>
</container>
</library>

```

Figure 285

Known problems & limitations

Component	Known problem/limitation
All output formats	All cells in merged groups must have same border properties for uniform output.
All output formats	Overlapped merged cells groups are not supported. If two merged groups have common cells, the output behavior is undefined. HTML, Word, Xsl-Fo : The text from any cell within a merged group of cells that is not in the first cell of that merged group is lost.
All output formats	Master pages are not supported when used on table child elements, list details and containers hosted in tables, rows and lists.
All output formats	Combining max width/max height properties with width/height properties for images results in undefined behavior.
All output formats	Setting the target region for an element a region contained in that same element results in undefined behavior.
All output formats	Span with cells that are not written in a table is not supported. If a cell has the 'col span' or 'row span' property set, and the column index plus the 'col span' value refers to a cell that doesn't exist in the table, or the row index plus the 'row span' value refer to a cell that doesn't exist in the table, the result is undefined.
All output formats	Having more than one Table of Contents, Table of Figures or Table of Tables per template will result in duplicated data.
All output formats	If a child list within a multilevel list has no text before it, an extra bullet appears on the output before that list. The extra bullet is generated by the parent list-detail.
All output formats	If an element has both masterpage and target region, the results are undefined.
All output formats	Only font formatting is inherited by child elements.
All outputs formats	Styled text supports only font formatting.
All output formats	Using invalid values for formatting properties results in undefined behavior.
Document Template	Table cell size cannot be relative to the table size.

HTML output	<p>For cell border style double the border width value of 1 makes the cell borders invisible, as the two borders on each side are collapsed.</p> <p>Setting the cell border width value to 3 makes the 'double' border style to be visible on Html. Thus, the table shall look like the one in Word when the border width is equal with 1.</p>
PDF output	When the header's content overlaps with footer's content, the PDF Document cannot be written. Please design the template and its master pages so that the header and footer do not overlap.
PDF output	Bookmarks and regions add extra blank lines in the output in special circumstances.
PDF output	If a heading has level +2 or more from the previous heading, the numbering for current heading continues the numbering of the previous heading that has the same level. Ex: 1, 1.1, 2, 2.2.1 instead of 1, 1.1, 2, 2.1.1
PDF output	Lists support Arabic numbers and bullets only
PDF output	If an internal hyperlink targets a bookmark located after the first text element in a cell, and if there is no other element after the bookmark, the PDF document cannot be written. Setting the bookmark as the first element in the cell solves the problem.
PDF output	When the page margins overlap, the PDF output will fail.
PDF output	If a referred target region does not exist, the output will fail.
PDF output	<p>If a nested table has cells with cell width set, and if there is at least a column in that table that has no cell width set on its cells, than the nested table overlaps the parent's borders.</p> <p>The workaround is to explicitly set the dimension for each cell.</p>
PDF output	Very large images and tables slow down the PDF generation process.
PDF output	A region is always written in a new section.
PDF output	Headings do not support indentation.
Word, PDF output	If an image has a big ratio between its two dimensions, and if the maximum dimensions are set and have very close values, the image in the outputs might not have the expected dimensions. Setting just one maximum dimension solves this issue.
Word output	Different numbering types on a multilevel list are not supported (for example, "arabic" on level 1 and "bullet" on level 2).

Word output	Restart paragraph numbering is not supported
Word output	If a multilevel list doesn't end with a list-detail at level 1, the numbering for that list-detail that is missing appears on the output.
Word output	The bookmark name provided by the user must be shorter than 20 characters.
Word output	The color palette for text background color (highlight color) is limited in Word. Thus, the color that is displayed in this case is the one that is most appropriate with the color chosen by the user.
Xsl-Fo output	If a heading is situated before the table of contents element, the numbering of the hyperlink in the table of contents is missing on the output side.
Xsl-Fo output	The header&footer's left&right margin are not modified when the page margins properties of the "masterpage" element are modified.

Appendix: Notices

© Copyright 2000, 2009

U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.