

## **IBM® Rational® Rhapsody® Gateway Add On**



## **DOORS Coupling Notes**



***Rhapsody<sup>®</sup>***

**IBM<sup>®</sup> Rational<sup>®</sup> Rhapsody<sup>®</sup>  
Gateway Add On**

**DOORS Coupling Notes**



## **License Agreement**

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner.

The information in this publication is subject to change without notice, and Dassault Systèmes and its affiliates assume no responsibility for any errors which may appear herein. No warranties, either expressed or implied, are made regarding Rhapsody software and its fitness for any particular purpose.

## **Trademarks**

Reqtify is a registered trademark of Dassault Systèmes or its affiliates in the US and/or other countries.

Rhapsody Gateway, IBM, the IBM logo, DOORS and Rhapsody are trademarks or registered trademarks of IBM Corporation.

All other product or company names mentioned herein may be trademarks or registered trademarks of their respective owners.

© Copyright 2001-2011 Dassault Systèmes. All rights reserved.

# Contents

---

<b>Contents .....</b>	<b>5</b>
<b>Introduction .....</b>	<b>7</b>
<b>DOORS Analysis .....</b>	<b>9</b>
DOORS Versions Supported .....	9
Rhapsody Gateway Elements Compared to DOORS Elements .....	9
Section .....	9
Macro-requirement .....	10
Requirement .....	10
Entity .....	10
Reference .....	10
Attribute .....	11
Reference Attribute .....	11
Link .....	11
Text .....	12
DOORS Collaboration Process .....	12
Defining Rhapsody Gateway Requirements .....	12
Importing Requirements from DOORS .....	12
Selecting a DOORS module .....	13
Intermediate XML File with DOORS Objects .....	16
Module Tag .....	18
Object Tag .....	19
Attribute Tag .....	19
Text Tag .....	19
Picture Tag .....	19
Out-link Tag .....	19
In-Link Tag .....	20
DOORS Types in Rhapsody Gateway .....	21
DOORS Basic .....	21
DOORS Advanced .....	22
DOORS Full Hierarchy .....	22
Customized Type Definition .....	22
Creating New Type .....	23
Capturing Links .....	25
Object Captured Twice .....	26
<b>DOORS Modules Samples .....</b>	<b>29</b>
Elevator Specs Module .....	29
Elevator Specs – Advanced Module .....	30
Hierarchical Reqs Module .....	32

Use of Shall for Reqs .....	32
<b>Connection Profiles .....</b>	<b>35</b>
Creating Profiles .....	35
Using Profiles .....	36
<b>Exporting to DOORS .....</b>	<b>37</b>
Exporting Document to DOORS .....	37
Setting DOORS Export .....	37
Synchronizing the Document .....	41
Navigating to DOORS .....	42
Traceability and Links in DOORS .....	43
Attributes Export .....	44
Enumerated Attributes Export .....	44
Specific and Generic Attributes Export .....	45
Capturing Traceability Performed in DOORS .....	45
Capturing Traceability Performed within Rhapsody Gateway .....	45
<b>Troubleshooting.....</b>	<b>47</b>
Changing DOORS Version .....	47
DOORS Client Cannot Run.....	47
DOORS Feature under Linux .....	48
DOORS Installation Problem .....	48
Setting a DOORS Server .....	48
DXL Error: Unknown Attribute .....	48

# Introduction

---

This technical note describes how to capture the traceability information of requirements from DOORS and how to create objects in the DOORS Database using Rhapsody Gateway.

Some standard operations, such as DOORS module import, navigation or export to DOORS in the Rhapsody Gateway tool, require the DOORS tool to be installed.

Refer to the *Customization Guide* and run the Coupling example for more information about what is described in this technical note.



# DOORS Analysis

---

This chapter describes how Rhapsody Gateway captures information items from DOORS and how those information items are analyzed to be used as requirements, attributes, text, and so on.

This chapter covers the following topics:

- ◆ Getting the information items from DOORS.
- ◆ Configuring Rhapsody Gateway according to the Requirements definition in DOORS.
- ◆ Analyzing information to get the same definition in Rhapsody Gateway.

## DOORS Versions Supported

Rhapsody Gateway supports DOORS versions from 7.x to 9.3.

The DOORS 8.0 version is supported only if Patch 5 is installed.

## Rhapsody Gateway Elements Compared to DOORS Elements

Rhapsody Gateway defines the following traceability elements.

### Section

A section is a hierarchical file description element. The following are examples of sections:

- ◆ Heading levels in a Microsoft Word file
- ◆ Tabs in a Microsoft Excel spreadsheet
- ◆ UML packages, diagrams
- ◆ Modules, sub-modules and components of design models

The tree composed by Rhapsody Gateway's sections is the same as the tree composed by the DOORS Object Headings with its hierarchy defined by its levels.

## Macro-requirement

A macro-requirement is a “super-requirement” that includes requirements and passes its properties onto those requirements.

Any new element attached to a macro-requirement (attribute, text, or link other than coverage link) is also attached to the requirements and derived requirements contained within the macro-requirement.

If the macro-requirement is directly referenced by a forward element, all of the requirements that it contains are considered as referenced by this element.

DOORS does not directly include this concept. Instead, DOORS includes a hierarchy of objects, which can be used to decompose a requirement (a DOORS object defined as a requirement). There is no direct relationship between DOORS and Rhapsody Gateway models, but at the User level both solutions support requirements hierarchies.

## Requirement

A requirement expresses either a need or constraints (technical constraints, costs, deadlines, and so on). The requirement is written either in natural language or as an expression—which may be mathematical, geometric, computerized, and so on.

DOORS includes **objects** that are defined as requirements, usually using dedicated DOORS attributes. Rhapsody Gateway captures these typed objects.

## Entity

By defining an entity, the user defines an element that must cover (contain a reference to) a requirement. If a defined entity does not contain any reference, Rhapsody Gateway will display a warning message.

This is quite an advanced notion used only in specific cases, for example to detect a dead code.

An entity cannot be referenced itself.

This notion does not directly exist in DOORS. However, entities can be compared to DOORS objects for which a DXL rule has been defined to verify that there is at least one out-link attached to the object.

## Reference

A reference is the information indicating the coverage (implementation or verification) of a requirement. A reference points to a macro-requirement, requirement, or derived requirement.

In Rhapsody Gateway, the reference can be defined either in a bottom-up direction, where the lower-level element covers the higher-level element, or in a top-down direction, where the higher-level element is covered by the lower-level element.

The references are the equivalent of DOORS **in-links** and **out-links** when these links are used to express requirements coverage.

### Attribute

Attributes complete the requirement. The following are examples of attributes:

- ◆ Type of check—test, observation, and so on.
- ◆ Category—functional, operational
- ◆ Criticality—low, high, and so on.
- ◆ Flexibility—low, high
- ◆ Maturity—source, analyzed, approved, and so on.

Rhapsody Gateway allows you to define attributes to be analyzed in the project files and filters the display in accordance with these attributes.

Rhapsody Gateway attributes are equivalent to DOORS attributes.

### Reference Attribute

A reference attribute is added to a reference to define the type of coverage, such as partial coverage or provisional coverage.

Rhapsody Gateway reference attributes are equivalent to DOORS **Link attributes** applied to coverage links.

### Link

A link is reference information that does not concern coverage. The following are examples of links:

- ◆ Supported by
- ◆ Issued by
- ◆ Checked by
- ◆ Valid under
- ◆ Allocated to
- ◆ Result of

Links are equivalent to DOORS **in-links** and **out-links** when these DOORS links have meanings other than requirements traceability.

## Text

Text is the wording of a traceability element. Rhapsody Gateway attaches the text to the element (section, requirement) detected immediately above it.

Rhapsody Gateway text is equivalent to DOORS Object text.

## DOORS Collaboration Process

### Defining Rhapsody Gateway Requirements

DOORS manages objects, but these elements are not exactly requirements. Some objects are considered to be requirements because of the addition of dedicated information. For example, an attribute like “Requirement = True” is seen as a requirement.

This customization of DOORS objects has to be specified in Rhapsody Gateway in order to allow the capture of objects as requirements. See below for an example in a DOORS module:

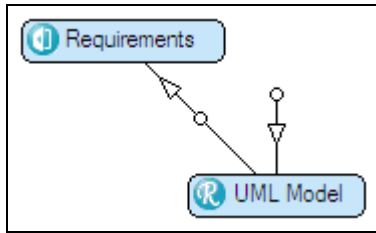
ID		Requirement
HL_Req_1	<b>1 Calling the elevator</b>	False
HL_Req_2	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator	True
HL_Req_3	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	False
HL_Req_6	<b>2 In the elevator</b>	False
HL_Req_7	Once in an elevator, a passenger can select the floor, or a list of floors, where he wants to go to	True

### Importing Requirements from DOORS

High level requirements can be captured from DOORS. Rhapsody Gateway can capture the following elements in a DOORS module:



- ◆ requirement IDs
- ◆ requirement texts
- ◆ attributes
- ◆ sections

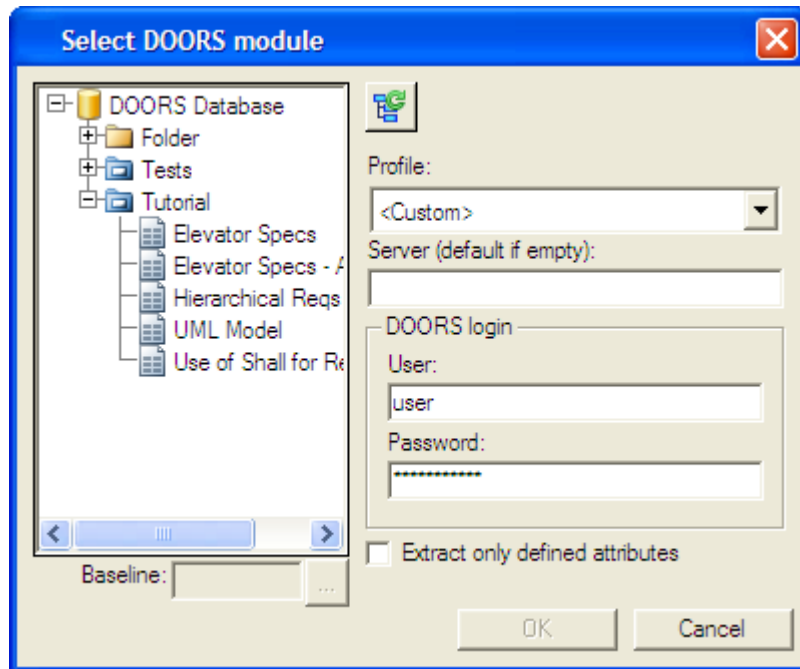
Define a Rhapsody Gateway project to import DOORS requirements, as shown in the following figure.



## Selecting a DOORS module

The DOORS module to be analyzed is selected during the Project Configuration phase in Rhapsody Gateway, as follows:

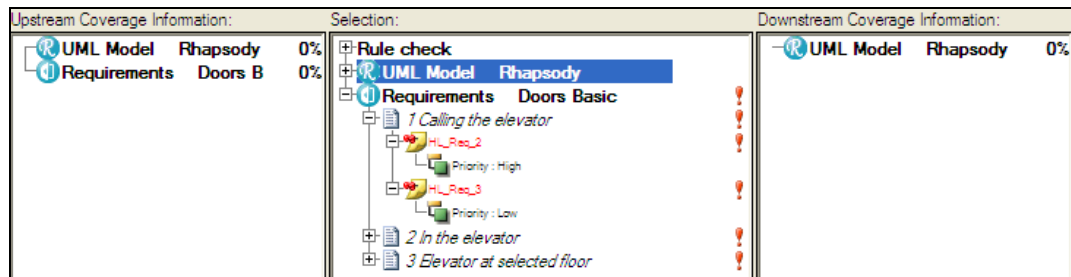
1. In the Project Editor, insert a document then select a type of analysis based on DOORS.
2. Click on the **File or Directory** field, then on the Browse  button. Rhapsody Gateway displays a selection window.
3. Provide a valid login information for DOORS then hit the [Enter] key or click the  button.
4. Rhapsody Gateway displays a view of the DOORS database, allowing the selection of the formal module to be imported. Select the DOORS module.



5. You can also specify a Profile, the Baseline you want to extract and the Server if it is not the default one. To specify a distant server, fill the field with `<PortNumber>@<Machine>`. See the *Connection Profiles* chapter to know how to create a profile.
6. Validate the dialog box contents by clicking **OK**.

Click **OK** or **Apply** in the Rhapsody Gateway project editor to run the analysis.

The requirements contained in the selected module are imported and displayed in the Rhapsody Gateway project workspace.



The attributes are displayed underneath the requirements.

## Specific parameters of a DOORS based type

When you select a type of analysis defined to capture information from DOORS (either a default type or a customized one), a **Variable** drop down list allows you to access some available variables that can be used for the parameters of a selected DOORS Module. The available parameters for a DOORS based type are the following:

- ◆ **Capture diagrams**—Importing DOORS images can be time-consuming. If images are not necessary, select Capture diagrams from the Variable drop-down list and deactivate the option in the Value field. Only images inserted in the DOORS text are taken into account.
- ◆ **Profile**—This parameter specifies the profile to use for the connection if some are defined.
- ◆ **Extract only defined attributes**—Importing a DOORS module that has a large number of DXL attributes can be very time consuming, as DOORS will calculate the value of these attributes before providing them to Rhapsody Gateway. This option can be used to limit the communication between DOORS and Rhapsody Gateway, by extracting only the attributes defined in the type of analysis. For example, if an attribute “Priority” has been defined in the type, only “Priority” values will be extracted from DOORS. The other attributes contained in the module will not be extracted.
- ◆ **Baseline**—This parameter specifies the baseline you want to extract. The value is specified in the selection dialog box.
- ◆ **Server**—This parameter specifies the server you want to consider. The value is specified in the selection dialog box.

- ◆ **Extract objects from**—This parameter specifies the sub-part of the module DOORS that is to be extracted. An extraction criterion specified in the Value field enables you to extract only the desired objects. Once a parent object satisfies the requirements all the sub-objects are extracted. The extraction expression follows the DXL regular expressions syntax which differs from the one used in Rhapsody Gateway (Refer to the DXL documentation). There are two different syntaxes:

- Extract according to DOORS Object ID: [=]RegExp
- Extract according to a specific attribute: Attribute=RegExp

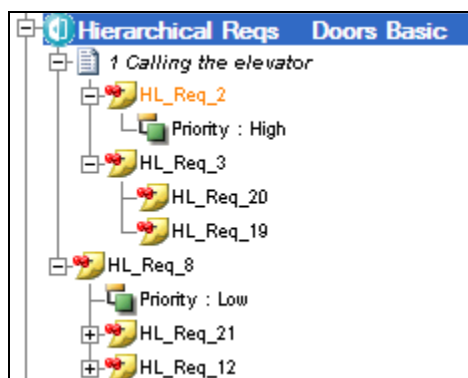
As an example, considering the following DOORS module with the **ToExtract** column:

ID	ReqID	Requirement	Priority	ToExtract
HL_Req_1		<b>1 Calling the elevator</b>	False	Yes
HL_Req_2	Requirement1	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator	True	High
HL_Req_3	Requirement2	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	True	Low
HL_Req_6		<b>2 In the elevator</b>	False	No
HL_Req_8	Requirement3	Each elevator will have a list of floors to visit : Once the elevator has been called by a potential passenger or a passenger has selected a destination, then the elevator will move to the appropriate floor.	True	Low
HL_Req_9		<b>3 Elevator at selected floor</b>	False	No
HL_Req_10	Requirement4	When the elevator has arrived at a floor and the doors have opened, then the passenger can exit the elevator	True	Low
HL_Req_11		This is a new Requirement	False	

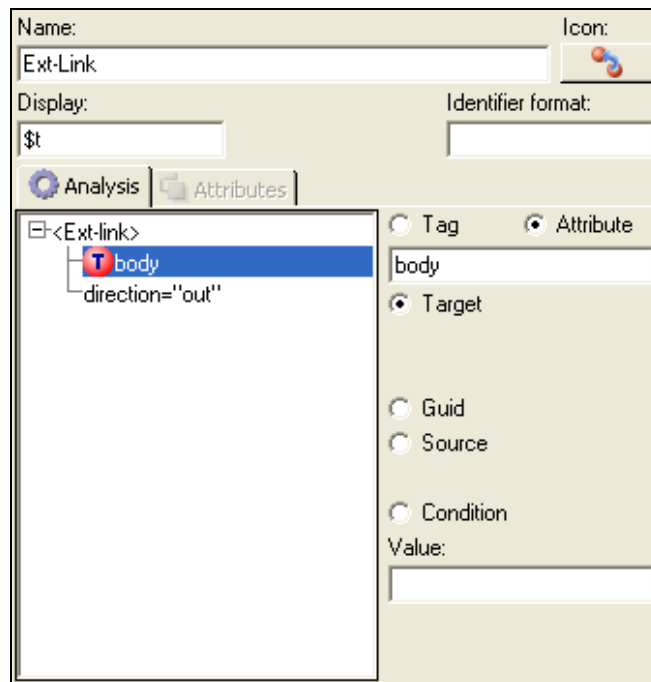
If you enter **ToExtract=^Yes\$** regular expression in the **Value** field for the **Extract objects from**, all the branches which parent verifies this expression are extracted.

Name	Type of Analysis	File or Directory	Ig...	In...	Bl...	Variable	Value
Hierarchical Reqs	Doors Basic	/Tutorial/Hierarchical	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Extract objects from	ToExtract=^Yes\$

The corresponding Rhapsody Gateway view is the following:



- ◆ **With External links**—This parameter enables the conversion of external links from DOORS modules. A customized DOORS type needs to be created to capture the External links. In this XML type, you need to define a link or a reference which can capture for instance:



- ◆ **With Table objects**—This parameter enables the DOORS tables to be extracted, so individual cells could be processed as requirements.

Table objects are converted into XML. The XML tags look like:

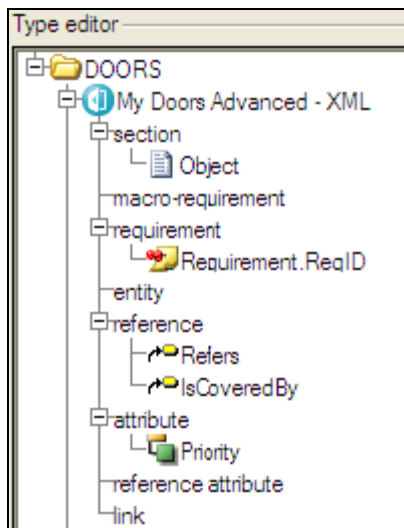
```

- <Object id="XX" table="1">
  <Attribute name="..." value="..."/>
- <Object id="XX" row="1">
  <Attribute name="..." value="..."/>
- <Object id="XX" cell="1">
  <Attribute name="..." value="..."/>
</Object>
- <Object id="XX" cell="2">
  <Attribute name="..." value="..."/>
</Object>
</Object>
<Object id="XX" row="2"> ... </Object>
</Object>

```

## Intermediate XML File with DOORS Objects

Rhapsody Gateway creates an intermediate XML file called `<document_name>.xml` in the **intermediate** sub-directory of the project directory. This file contains the descriptions of all the DOORS objects, along with their properties. The captured objects are provided by the type of analysis such as the definition of the expected requirements, attributes, etc.



### XML Tag

A **tag** is a generic name for an `<element>`. An opening **tag** looks like `<element>`, while a closing **tag** has a slash that is placed before the element's name: `</element>`.

From this point on the term **tag** will refer to the opening of an element.

### XML Attribute

**Attributes** are used to specify additional information about the element. An **attribute** for an element appears within the opening **tag**. The syntax for including an **attribute** in an element is:

```
<tag attributeName="value">
```

### Application to the XML file built by the converter

The converter communicates with DOORS and builds an XML file with the obtained information.

### Note

---

As the DOORS terms are similar to XML terms, the analysis of the XML file can be confusing.

**name="..."** is an XML attribute,

## Object Tag

An **Object** tag must be generated for each DOORS object in the module. The object tag must define the following XML attributes:

- ◆ An “id” attribute containing the **Object ID**,
- ◆ A “number” attribute containing the hierarchical position in the DOORS module.

An **Object tag** can contain another **Object tag** because elements can be nested.

## Attribute Tag

An **Attribute** tag is generated for an attribute having a specific value. This Attribute tag defines the following XML attributes:

- ◆ A “name” attribute containing the **DOORS Attribute Name**,
- ◆ A “value” attribute containing the **DOORS Attribute Value**.

For each object, all attributes are listed using the same syntax.

## Text Tag

A **Text** tag is generated as soon as the Object Text contains some text. The Text tag contains the **DOORS Object Text**.

## Picture Tag

A **Picture** tag is generated for an image located under the Object Text attribute. The Picture tag contains the binary in hexadecimal encoding.

## Out-link Tag

An **Out-link** tag is generated for links going out of the current object. **Out-links** are extracted only when the type of analysis defines a reference capture. This Out-link tag defines the following XML attributes:

- ◆ A “linkModule” attribute containing the name of the link module that defines the link.
- ◆ A “target” attribute containing the target identifier **Target Object** which can be:
  - Either the **Element Guid** or **Element Identifier** if the target object is exported,
  - An identifier attribute,
  - The DOORS **Object ID** when none of the above is applicable.

- ◆ A “targetModule” attribute containing the module that defines the target object.

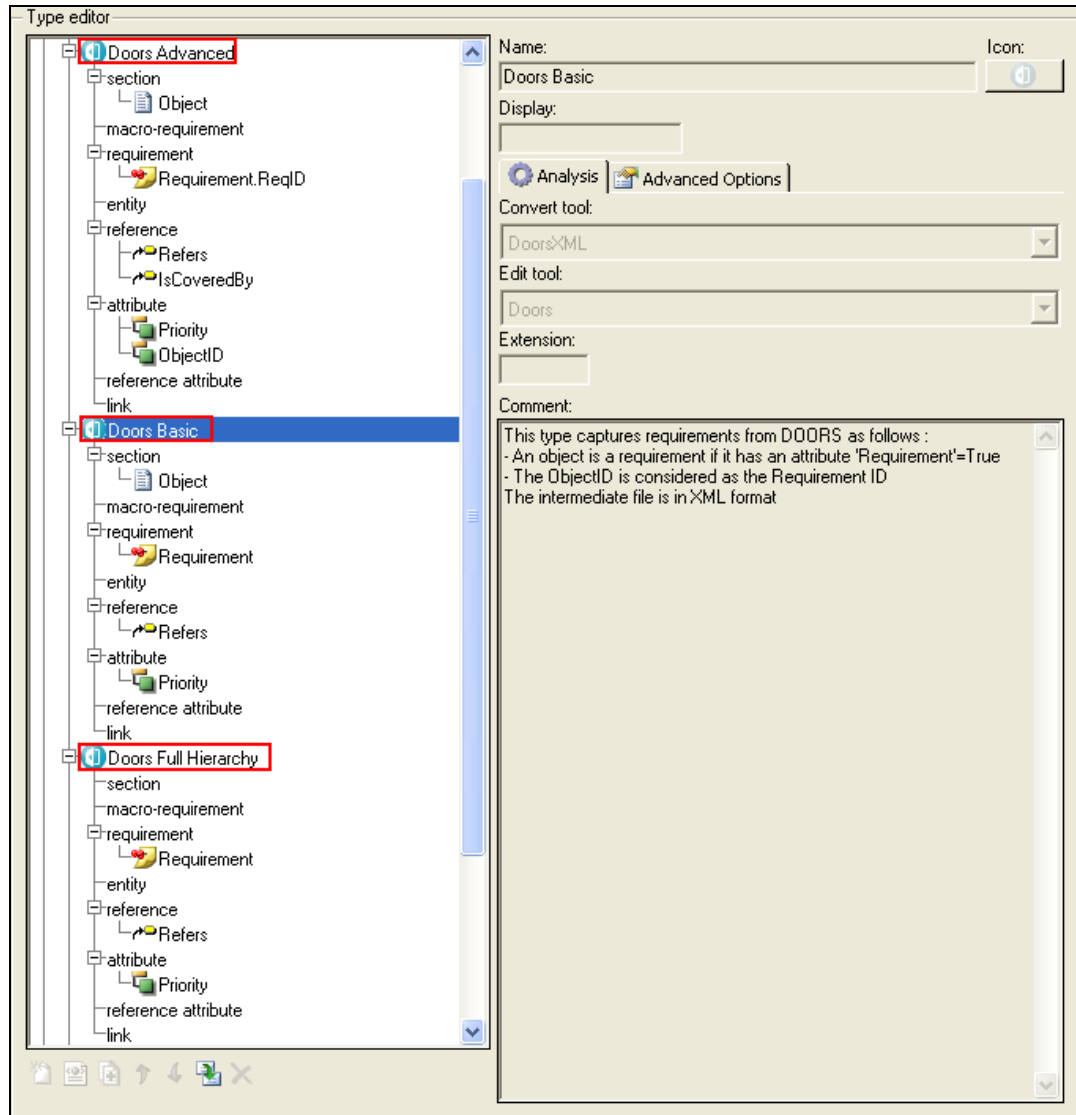
## In-Link Tag

An **In-link** tag is generated for links coming to the current object. **In-links** are extracted only when the type of analysis defines an inverse reference capture, and when the source object is defined in an exported module. This In-link tag defines the following XML attributes:

- ◆ A “linkModule” attribute containing the name of the link module that defines the link.
- ◆ A “source” attribute **NameOfLinkModule** containing the target identifier which can either be the **Element Guid** or **Element Identifier**.
- ◆ A “sourceModule” attribute containing the module that defines the source object **SourceObject**.

## DOORS Types in Rhapsody Gateway

The DOORS coupling package provides three Rhapsody Gateway types for DOORS analysis: **DOORS Basic**, **DOORS Advanced** and **DOORS Full Hierarchy**.



### DOORS Basic

This type captures requirements from DOORS as follows:

- ◆ An object is a requirement if it has a DOORS attribute 'Requirement'=True
- ◆ The ObjectID is considered as the Requirement ID

The intermediate file is in XML format.

In the following module DOORS example, if you choose the DOORS ObjectID **HL\_Req\_3** for the field Value, the extracted ObjectID in Rhapsody Gateway is **HL\_Req\_3** and all its children i.e. **HL\_Req\_20** and **HL\_Req\_19**.

Hierarchical Reqs	ID	ReqID		Requirement
1 Calling the elevator	HL_Req_1		<b>1 Calling the elevator</b>	False
A potential passenger can b	HL_Req_2	Requirement1	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or down button to call the modif - Updated	True
The potential passenger wai	HL_Req_3	Requirement2	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	True
This is a sub requiremen				
This is another sub requ				
2 In the elevator	HL_Req_20	Req2.1	This is a sub requirement of HL_Req_3	True
2.1 Section	HL_Req_19	Req2.2	This is another sub requirement of HL_Req_3	True
Each elevator will have a list				
This is a sub requiremen				
My children requirement	HL_Req_6		<b>2 In the elevator</b>	False
3 Elevator at selected floor	HL_Req_14		<b>2.1 Section</b>	False
When the elevator has arriv				

## DOORS Advanced

This type captures requirements from DOORS as follows:

- ◆ An object is a requirement if it has an attribute 'ObjectType'=Requirement
- ◆ The ObjectID is the value of an attribute 'ReqID'

The intermediate file is in XML format.

## DOORS Full Hierarchy

This type captures requirements from DOORS as follows :

- ◆ Every DOORS object is considered as requirement
- ◆ The whole hierarchy of the DOORS module is maintained
- ◆ The ObjectID is considered as the Requirement ID
- ◆ The Object Heading is captured as the Requirement Label when it is available

The intermediate file is in XML format.

## Customized Type Definition

As DOORS manages objects and Rhapsody Gateway expects requirements, it is necessary to specify how objects can be captured as requirements in the XML file. This capture can be performed using the definition of Types of Analysis. The Types of analysis also contain definitions of Attributes, Text, etc.

Application Engineers, the Support Team, and advanced users using this *Coupling Note* can perform this customization work.

## Creating New Type

Select **File > Edit Types** to launch the Type Editor and select a DOORS based type in the Requirement Tools folder.

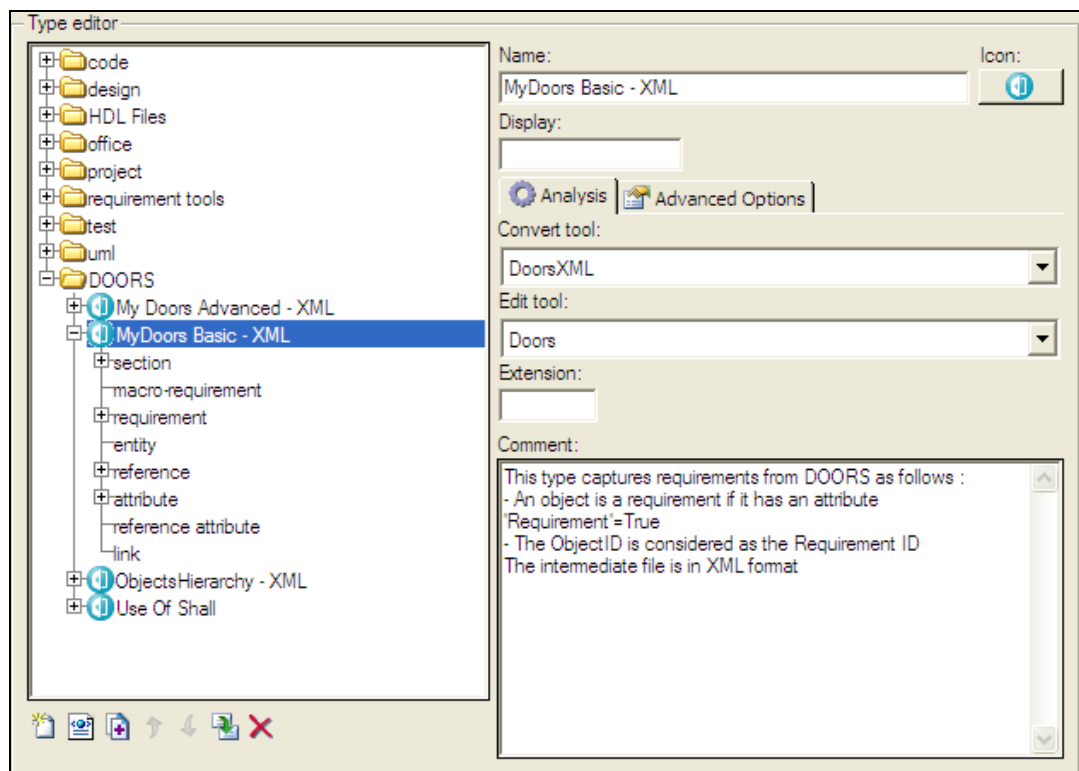
Duplicate this type in order to create your own. Right-click the DOORS type in the Type Editor, and select **Duplicate**.

### Note

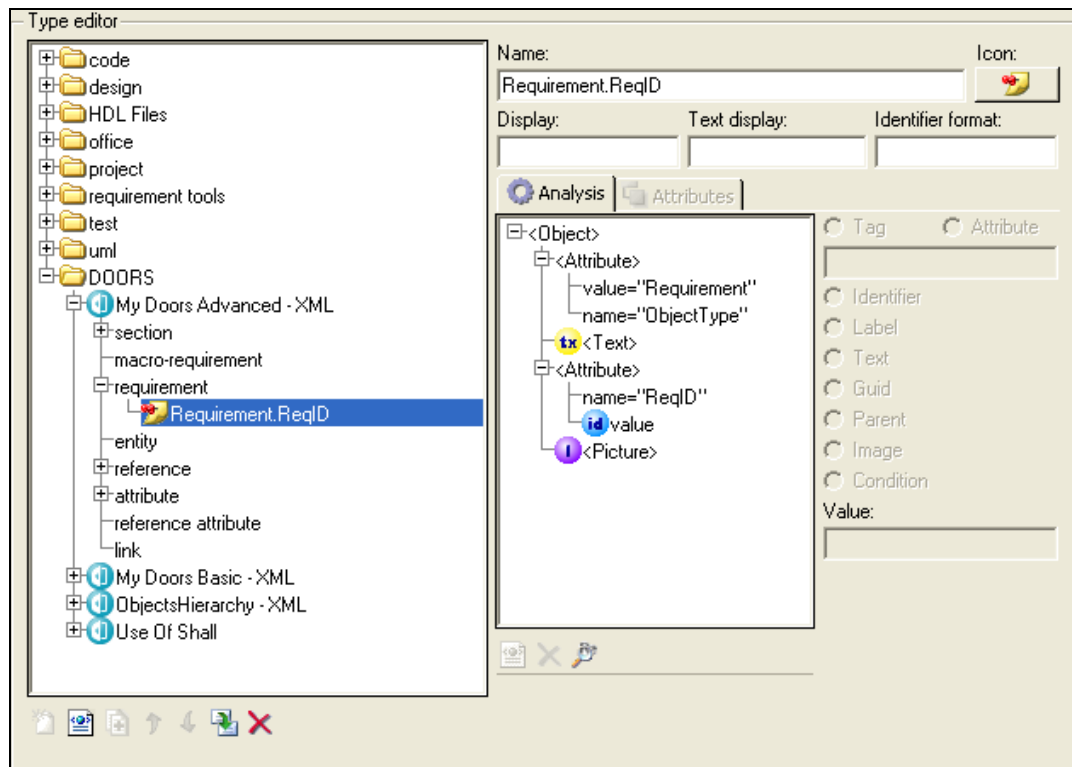
You can use the helpful information found in the DOORS example. This example contains several customized cases.

With the DOORS type selected, fill in the following fields:

- ◆ **Name**—Enter a name for the type, for example SRS DOORS
- ◆ **Display**—Use this field to display a different name in the main window
- ◆ **Convert tool**—Select DOORS XML
- ◆ **Edit tool**—Select DOORS
- ◆ **Extension**—Leave blank
- ◆ **Comment**—Add comments about the type definition in this text area



Each element of the type (Requirement, Attribute) has to be defined using the XML syntax allowing you to capture the relevant information in the intermediate file.



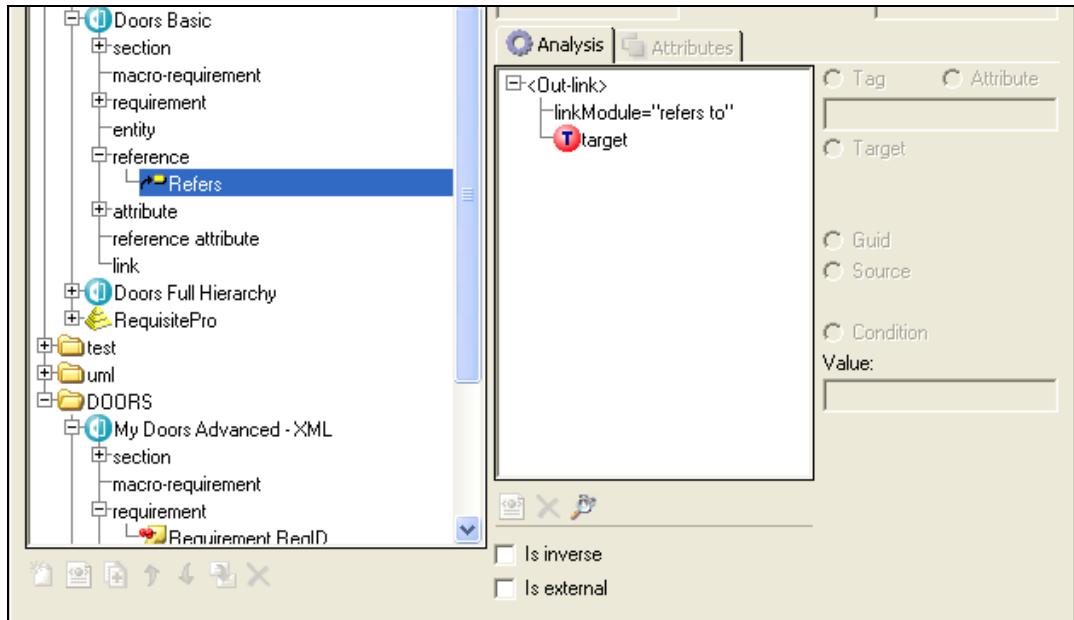
Refer to the *Customization Guide* for further information about the different fields.

### Note

Using the **Identifier format** for high level requirements capture causes the link synchronization to fail when lower level requirements are exported to DOORS. Indeed, “export to DOORS” feature can no longer find the corresponding DOORS Objects once its identifier was altered by the identifier format.

## Capturing Links

In the provided DOORS types, the links are captured using the value "refers to" for the "**linkModule**" Attribute.

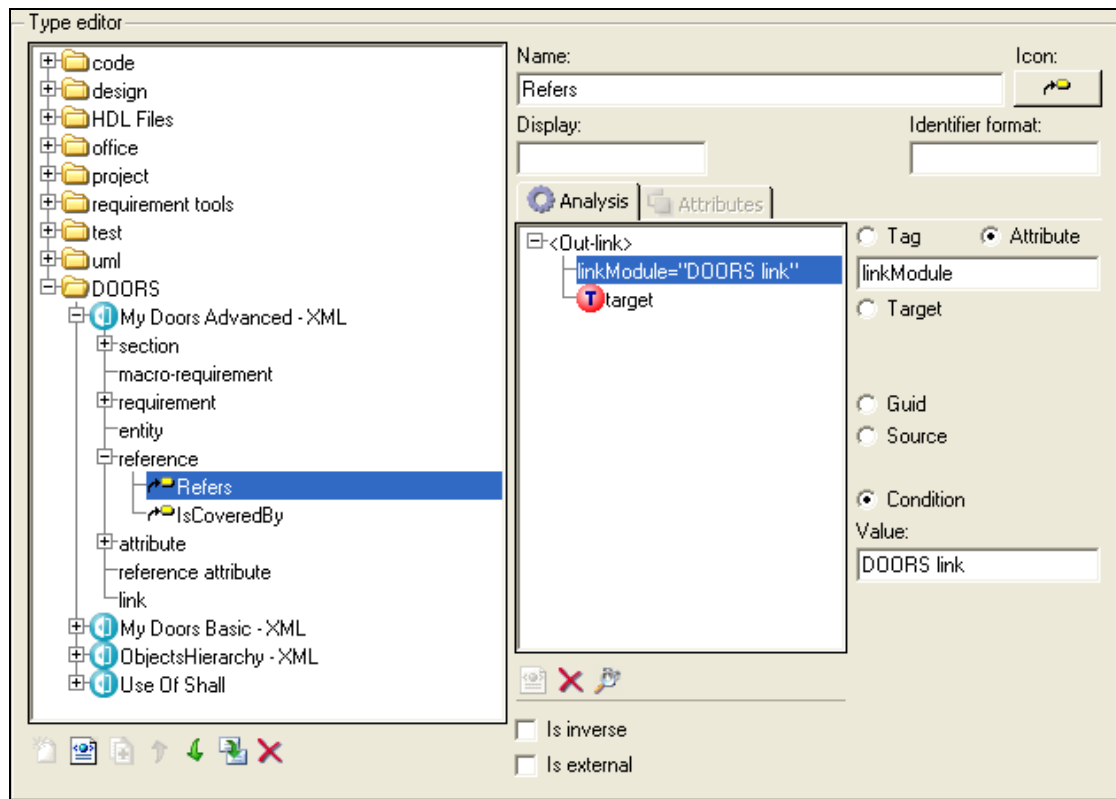


Frequently in DOORS module, link modules are named "DOORS link". To capture the DOORS links with specific name in Rhapsody Gateway, you need to use a customized DOORS type.

If you are using a provided DOORS type, duplicate this type, otherwise modify the current type.

To capture a specific link module, follow these steps:

1. In the customized type, create a new reference or modify an existing one.
2. Select the link. In the XML area, select the **linkModule** element, its value is displayed in the **Value** field.
3. Change the name of the **linkModule** in the **Value** field. Apply the changes.



4. The customized type should be applied to the DOORS module to find the link modules this time.

## Object Captured Twice

When we are capturing a requirements hierarchy, sometimes some unwanted objects are also captured. For instance, it happens when a text object contains a title and some text such as follows:

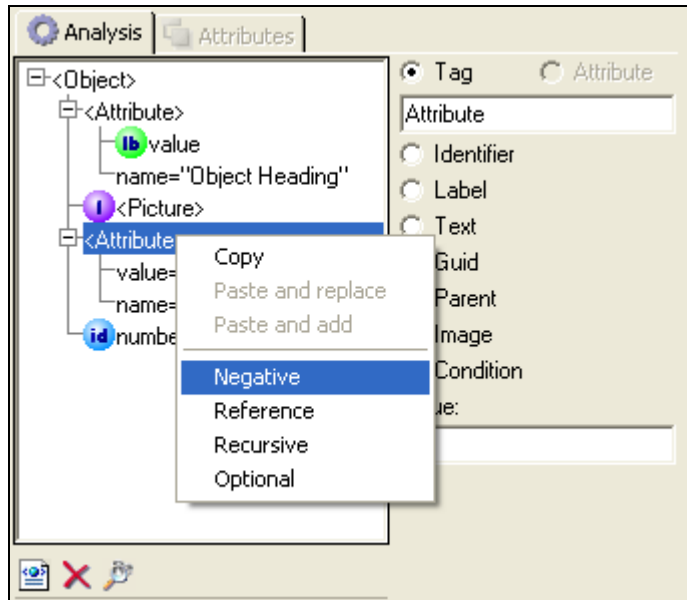
ID		Requirement	Priority
HL_Req_2	<b>1.1 Calling the elevator</b>  A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator.__MAJ	True	High
HL_Req_3	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	False	

Using the DOORS Basic on this example, for the HL\_Req\_2, the Title is captured twice: one time as a section title and one time as a requirement. We need to find a criterion to avoid the twice capture of the requirement. In order to, we will add a condition to capture only object that are not Requirement in the Section capture.

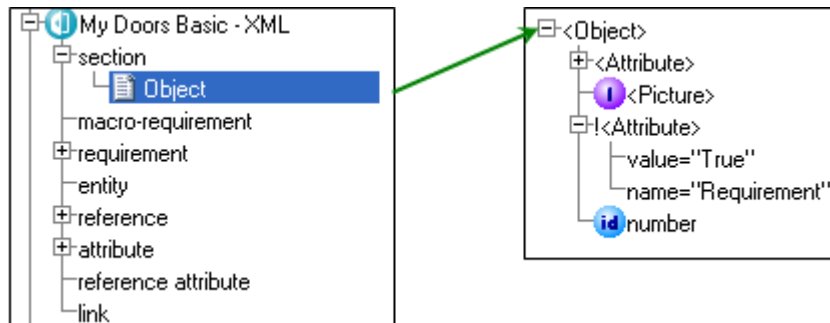
If you are using a provided DOORS type, duplicate this type, otherwise modify the current type.

To create a negative condition, follow these steps:

1. In the customized type, create a new attribute in the **section** part.
2. Assign the name 'Requirement' to this Attribute's name and 'True' to its value.
3. Now select the new Attribute then right click to choose the **Negative** item to apply to this element.



4. The Section XML definition looks like the following one:



5. Launch the analysis of your project.



# DOORS Modules Samples

---

The following examples come from the DoorsDemo example located in the directory `examples\coupling\DOORS`, which shows how to define a XML type of analysis in order to capture DOORS requirements.

## Elevator Specs Module

This example uses the extraction according to DOORS Object ID. In this module:

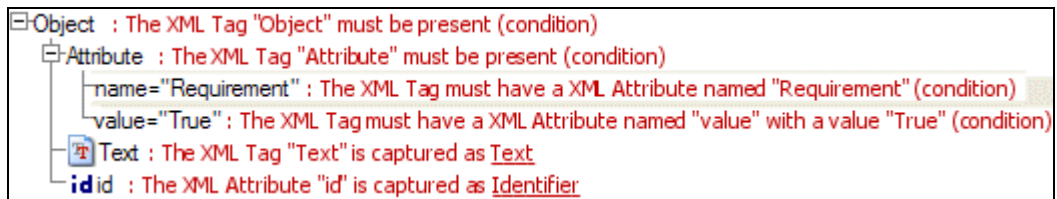
- ◆ An object is a requirement if it has an attribute Requirement = True
- ◆ The Requirement ID is the ObjectID

ID		Requirement
HL_Req_1	<b>1 Calling the elevator</b>	False
HL_Req_2	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator	True
HL_Req_3	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	False
HL_Req_6	<b>2 In the elevator</b>	False
HL_Req_7	Once in an elevator, a passenger can select the floor, or a list of floors, where he wants to go to	True
HL_Req_8	Each elevator will have a list of floors to visit : Once the elevator has been called by a potential passenger or a passenger has selected a destination, then the elevator will move to the appropriate floor.	False
HL_Req_9	<b>3 Elevator at selected floor</b>	False
HL_Req_10	When the elevator has arrived at a floor and the doors have opened, then the passenger can exit the elevator	True

In this case the intermediate file looks like:

```
<Object id='HL_Req_2' number='1.0-1' Element_Ident:
  <Attribute name='Absolute Number' value='2' />
  <Attribute name='Created By' value='John Doe' />
  <Attribute name='Created On' value='03 December' />
  <Attribute name='Created Thru' value='Manual In' />
  <Attribute name='Last Modified By' value='John' />
  <Attribute name='Last Modified On' value='19 Ja' />
  <Attribute name='Priority' value='High' />
  <Attribute name='Requirement' value='True' />
  <Text>A potential passenger can be on any of tl
</Text>
```

For the requirement capture, the XML element to be defined in the type of analysis is:



## Elevator Specs – Advanced Module

This example uses the extraction according to a specific attribute. In this module:

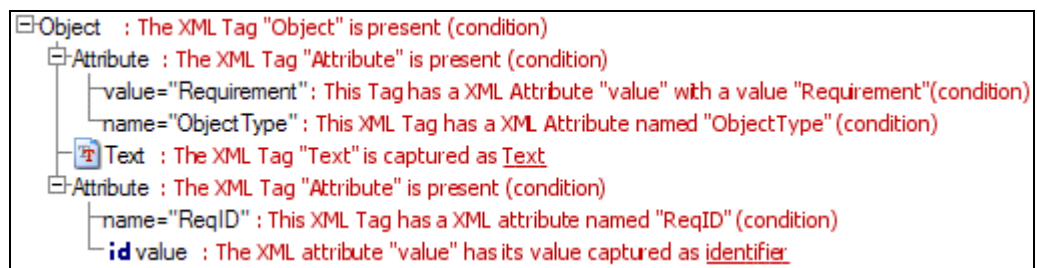
- ◆ An object is a requirement if it has an attribute ObjectType = Requirement
- ◆ The Requirement ID is the value of an attribute ReqID

ID	ReqID		ObjectType
1		<b>1 Calling the elevator</b>	
2	REQ1	A potential passenger can be on any of the floors and can call an elevator by pressing either the up or button to call the elevator.	Requirement
3	REQ2	The potential passenger waits for the doors to open before entering into the elevator. The potential passenger now becomes a passenger	Requirement
6		<b>2 In the elevator</b>	
7	REQ3	Once in an elevator, a passenger can select the floor or a number of floors where he wants to go to. Modif	Requirement
8		Each elevator will have a list of floors to visit : Once the elevator has been called by a potential passenger or a passenger has selected a destination, then the elevator will move to the appropriate floor.	Comment
9		<b>3 Elevator at selected floor</b>	
10	REQ4	When the elevator has arrived at a floor and the doors have opened, then the passenger can exit the elevator. Modif	Requirement

In this case the intermediate file looks like:

```
<Object id='2' number='1.0-1' Element_Identifier='' Element_Type=
  <Attribute name='Absolute Number' value='2' />
  <Attribute name='Created By' value='John Doe' />
  <Attribute name='Created On' value='03 December 2003' />
  <Attribute name='Created Thru' value='Manual Input' />
  <Attribute name='Last Modified By' value='John Doe' />
  <Attribute name='Last Modified On' value='24 January 2006' />
  <Attribute name='ObjectType' value='Requirement' />
  <Attribute name='Priority' value='High' />
  <Attribute name='ReqID' value='REQ1' />
  <Text>A potential passenger can be on any of the floors and ca
</Text>
```

For the requirement capture, the XML element to be defined in the type of analysis is:

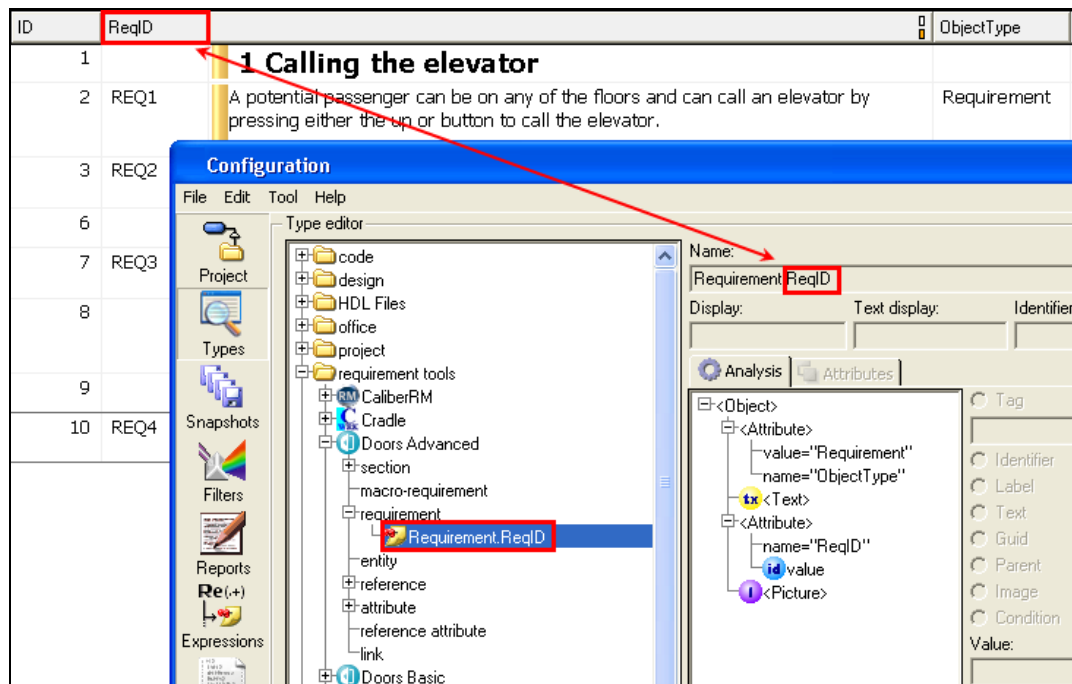


## Important

In such cases, the requirement ID is not the DOORS ObjectID. Therefore, it is necessary to declare the name of the DOORS attribute containing the requirement ID. This will provide Rhapsody Gateway with the information, allowing you to find the DOORS objects for navigation and link creation.

In this example we consider the **ReqID** attribute. If you use a different attribute, you also need to change the name of the requirement element.

- ◆ The default name of the requirement element in the Type is **Requirement.ReqID** (see below).
- ◆ Always use the syntax `Requirement.<your attribute name>`



## Hierarchical Reqs Module

This example uses the extraction according to a specific attribute. In this module:

- ◆ An object is a requirement if it has an attribute *Requirement = True*
- ◆ The Requirement ID is the value of an attribute *ReqID*
- ◆ Requirements are “Level1” objects, and some of them have “Level 2” requirements underneath

In fact, this case is equivalent to the previous one in terms of XML element definition. In the framework of the tutorial, this type is used to demonstrate how requirements elements can be nested.

## Use of Shall for Reqs

This example uses the extraction according to DOORS Object ID. In this module:

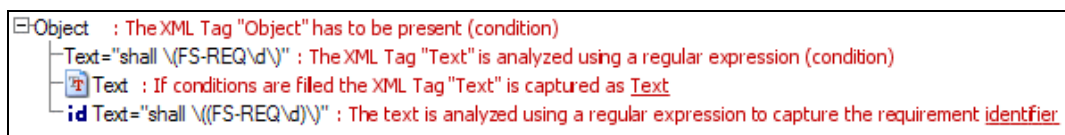
- ◆ There is no specific attribute to define a requirement
- ◆ An object is a requirement if its text contains the word “shall” followed by the requirement ID
- ◆ The Requirement ID is the ID FS-REQxx between parenthesis after “shall”

ID		Priority
1	This Module is used to demonstrate the capture of requirements with the formalism : - an object is considered as a requirement if its text contains the word "shall" - the Requirement ID is the ID mentioned after the word "Shall"	
3	<b>1 Operational Requirements</b>	
4	<b>1.1 Available Fuel</b> The Fuel Forecourt shall (FS-REQ1) have fuel available with a down time of no more than 1 hour per month.	High
6	<b>1.2 Image Recording</b> Before fuel pumping is allowed, an electronic image of the Car and the Motorist shall (FS-REQ2) be taken and recorded.	Low
8	<b>1.3 Fuel Display</b> As Fuel is supplied, the Motorist shall (FS-REQ3) be constantly updated with the amount of fuel pumped and the cost of the amount pumped.	High
10	<b>1.4 Fuelling</b> The motorist fills their car with fuel manually.	
12	<b>1.5 Payment Options</b>	

In this case the intermediate file looks like:

```
<Object id='4' number='1.1'>
  <Attribute name='Absolute Number' value='4'>
  <Attribute name='Created By' value='Administrator'>
  <Attribute name='Created On' value='18 January 2006'>
  <Attribute name='Created Thru' value='Manual Input'>
  <Attribute name='Last Modified By' value='John Doe'>
  <Attribute name='Last Modified On' value='27 January 2006'>
  <Attribute name='Object Heading' value='Available Fuel'>
  <Attribute name='Paragraph Style' value='&lt;Object Text:SRS_
  <Attribute name='Priority' value='High'>
  <Attribute name='Section' value='False'>
  <Text>The Fuel Forecourt shall (FS-REQ1) have fuel available
```

For the requirement capture, the XML element to be defined in the type of analysis is:



As a reminder on regular expressions:

- ◆ \ ( and \ ) mean “the specific character “(“ and “)””
- ◆ \d means “one digit”; for “one or several digits”, use \d+
- ◆ a string between ( ) is captured as a result: Results of (FS-REQ\d) will be FS-REQ1, FS-REQ2, etc. if such IDs exist.

The *Customization Guide* presents the syntax of regular expressions that can be helpful for Rhapsody Gateway customization.



# Connection Profiles

---

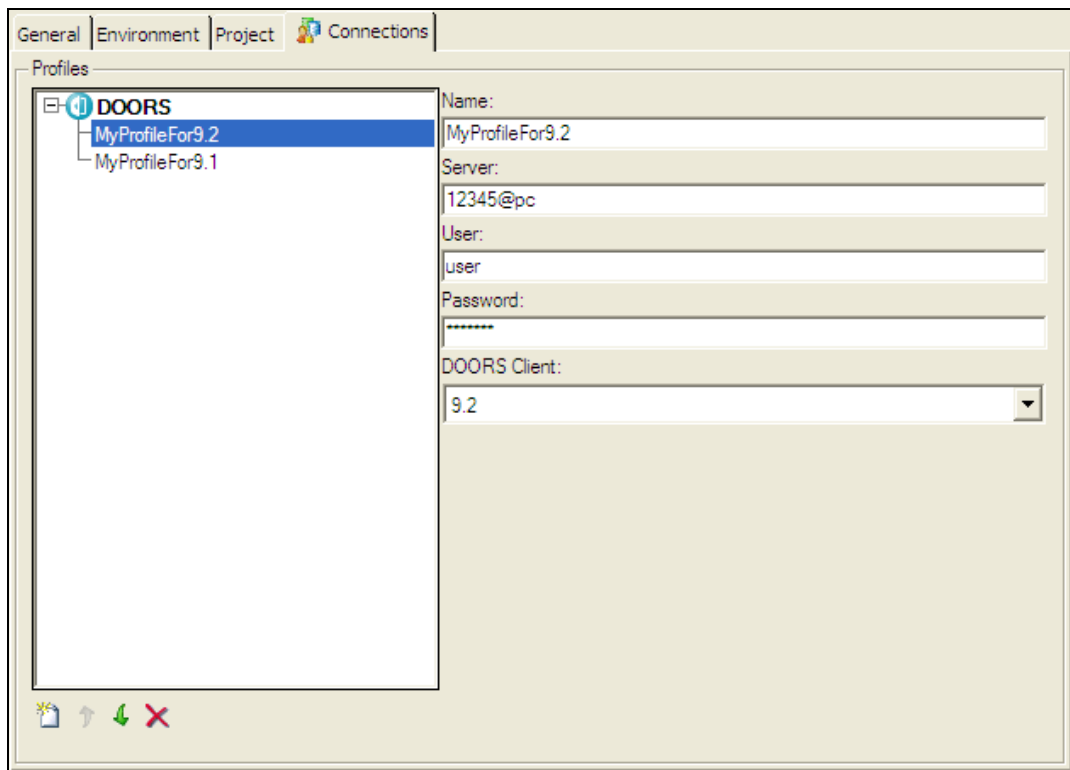
Connection parameters can be defined to configure the servers you often connect to. In order to, you can define few connections profiles to easily change of DOORS connection.

## Creating Profiles

Open the **Connections** tab of the **Options** dialog box in order to create a profile.

To create a new DOORS profile, follow these steps:

1. Select DOORS in the Profile area.
2. Click the **Add new profile** button. A blank profile appears.
3. Fill out the fields of the profile:
  - ◆ **Name**—Name the profile.
  - ◆ **Server**—Enter the address of the server to connect to. To specify a distant server, fill the field with <PortNumber>@<Machine>.
  - ◆ **User**—Enter the login of the user.
  - ◆ **Password**—Enter the user password.
  - ◆ **DOORS Client**—Select the DOORS Client version.



## Using Profiles

The created profiles can be used in the dialog box which select a DOORS module at project creation, and in the DOORS Export dialog box.

See the *Selecting a DOORS module* and *Setting DOORS Export* sections to visualize profile usage.

# Exporting to DOORS

---

Rhapsody Gateway allows you to upload lower level information into DOORS with the traceability related to DOORS requirements. This can be accomplished with or without diagrams (obviously they need to be captured before exporting). During the Export of elements to DOORS, some traceability information is automatically created into DOORS:

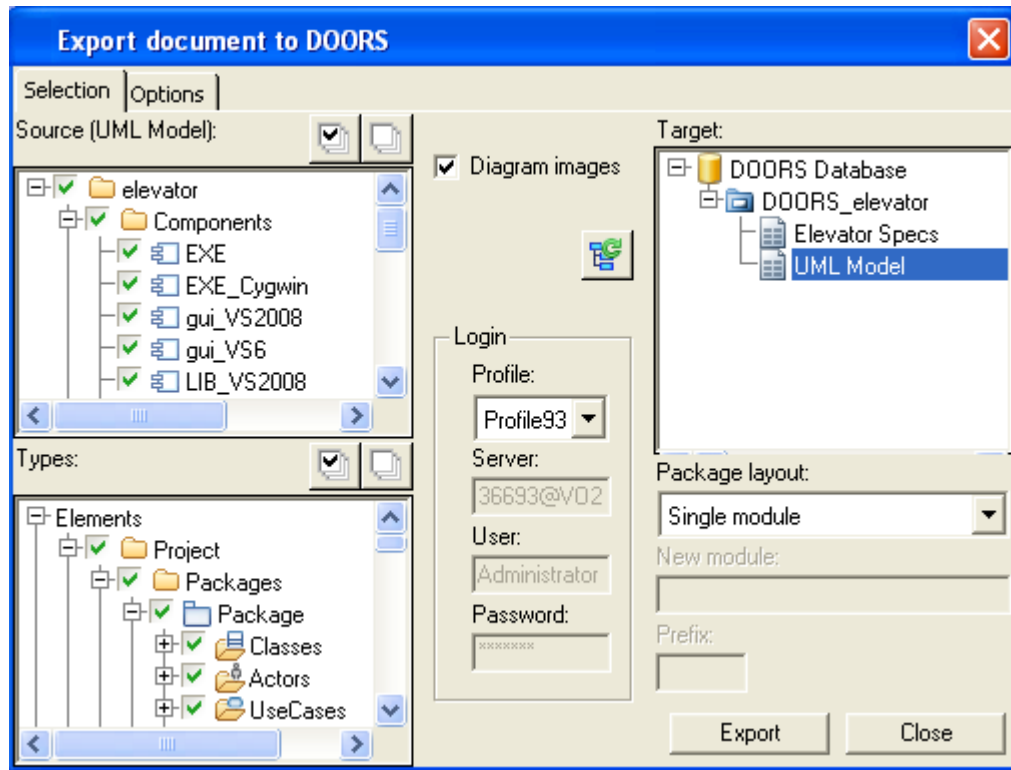
- ◆ **Modules** containing elements captured from interfaced tools,
- ◆ **Link Modules** containing the traceability information.

## Exporting Document to DOORS

### Setting DOORS Export

To export a document to DOORS, it is necessary to configure export settings.

Select the low-level document to export then click the **Export Document to DOORS** option from the Rhapsody Gateway **Tools** menu. The following **Export configuration** dialog box is displayed:



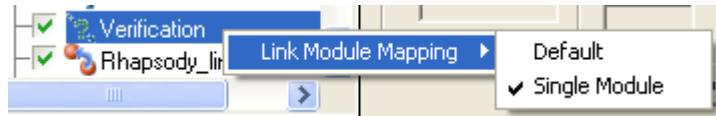
In the **Selection** tab of this window, following standard options in the form of lists, view, or fields are available:

- ◆ **Source** list—displays the current model hierarchy. Select the parts of the hierarchy that are to be uploaded.
- ◆ **Types** list—displays a hierarchic ordering sub-selection of the elements which are selected in the Source list. Within this subset, select the types that you want to upload. (i.e. the export content depends on the **Keep children of filtered elements** option).

Exported link types are also displayed in this area. A contextual menu is available on a link type to choose the behavior of the link after the export between:

- **Default**—Selecting this option, links are created as usual in a DOORS link module corresponding to the link or reference type name


- **Single module**—Selecting this option, links are created in the DOORS link module specified in the dialog box provided clicking this menu item.



### Note

---

It is recommended to delete the exiting link modules before modifying this option.

- ◆ **Login fields**—If the target view is empty, enter a valid user name and password and click on the  button to display the DOORS database.
  - **Profile box**—Select a DOORS profile if some were defined in the **Connections** tab from Rhapsody Gateway Options. Refer to the Connection Profiles chapter for more details.
  - **Server field**—Enter a server name if you want to export created DOORS objects to another server. To specify a distant server, fill the field using <PortNumber>@<Machine> syntax.
- ◆ **Target view**—Use this view to select a location in the DOORS database.
- ◆ **Diagram images** option—Use this option if you want to have the images of the document to be taken into account in the exported module to DOORS.

### Note

---

It is necessary to activate the image capture in the document prior to the export.

- ◆ **Package layout** option—Use this option to choose how modules will be organized after the DOORS export.
  - **Single module** This is the regular behavior, only one module is created.
  - **One module per package** Select this value if you want to split the original document into several DOORS modules. Consequently a project is selected instead of a module and one module is created for each package inside the project.
  - **One folder module per package** Select this value if you want to split the original document into several DOORS modules. Consequently a project is selected instead of a module and one folder module is created for each package inside the project to reflect the package structure of the model.

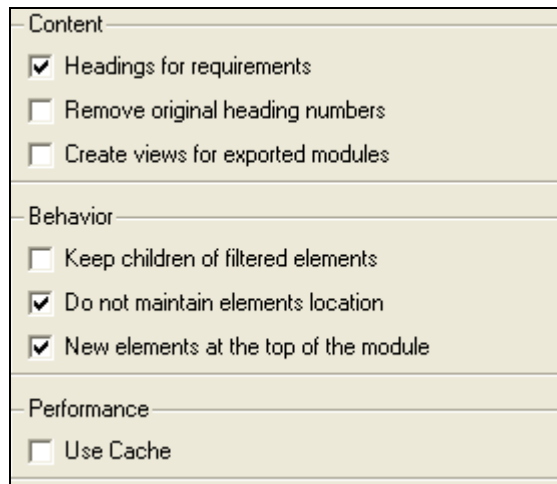
### Note

---

**One module per package** and **One folder module per package** are only available if a Rhapsody model has been selected to be exported to DOORS. In the case of a Word or other document type to be exported to DOORS, this option is not available.

- ◆ **New module** field—Enter a module name. The default name will be the name of the document defined in the Rhapsody Gateway project editor for the uploaded document.
- ◆ **Prefix** field—Enter a prefix if you want the created DOORS objects to have a particular prefix.

In the **Options** tab, advanced options are available, as shown below:



Content	
<input checked="" type="checkbox"/>	Headings for requirements
<input type="checkbox"/>	Remove original heading numbers
<input type="checkbox"/>	Create views for exported modules
Behavior	
<input type="checkbox"/>	Keep children of filtered elements
<input checked="" type="checkbox"/>	Do not maintain elements location
<input checked="" type="checkbox"/>	New elements at the top of the module
Performance	
<input type="checkbox"/>	Use Cache

- ◆ **Heading for requirements**—Use this option if you do not want to affect the DOORS objects heading of the exported requirements.
- ◆ **Remove original heading numbers**—Use this option if you want to remove the numbering of the original document.
- ◆ **Create views for exported modules**—Use this option to create a DOORS View which displays the exported attributes. This view is accessible from **View > Manage View**. It works only if DOORS runs in GUI mode.
- ◆ **Keep children of filtered elements**—Use this option if you want to export children objects of previously filtered elements (i.e. in the “Types” filter of the Selection tab).
- ◆ **Do not maintain elements location**—This option is available when a model supporting the GUID identification is exported. Use this option if you want to launch successive exports without taking into account the parent-children relationship of the elements and without maintaining the elements position.
- ◆ **New elements at the top of the module**—This option is available when “Do not maintain elements location” is selected. If this option is selected, the new elements created during the export are always created at the top of the module even if they are children or siblings of already exported objects.
- ◆ **Use cache**—Use this option if you want to store the conversion results of an exported module likely to be used again. As long as the module remains unchanged, the cache is re-used to spare the reloading of the module.

### Note

---

The **Use cache** option can also be used to get a preview of the export, which allows the user to save time in case of further export.

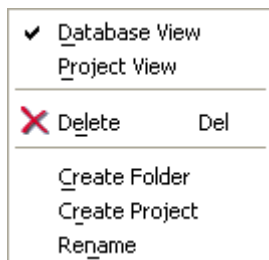
### Note

---

When the export is completed, the configuration is saved and it will be displayed by default in the export dialog box when reopened.

## Contextual Menu

The target area of the Export window contains a contextual menu.



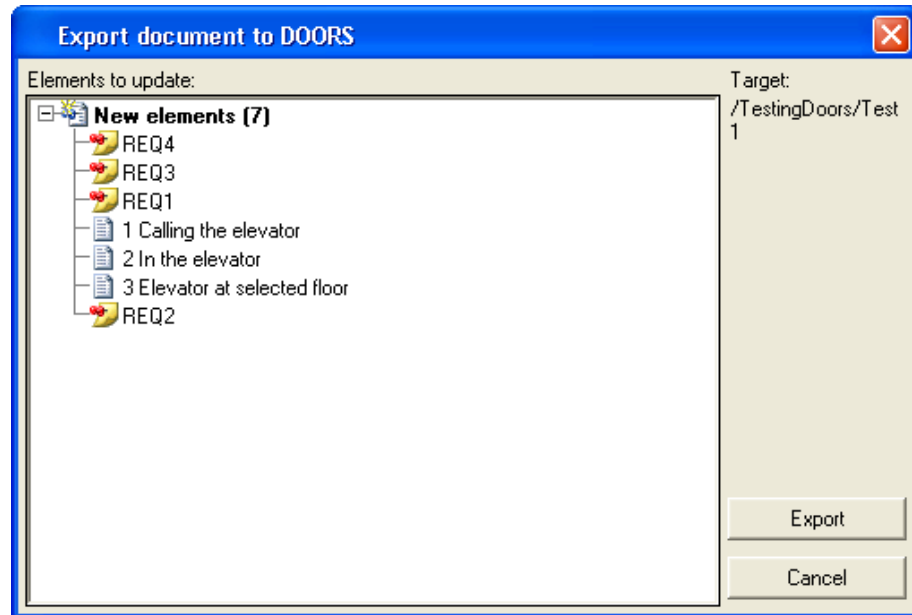
- ◆ **Database View**—Use this option to display the tree of the target view centers on database point of view.
- ◆ **Project View**— Use this option to display the tree of the target view centers on project point of view.
- ◆ **Delete**—Use this option to delete a project when exporting to DOORS.
- ◆ **Create Folder**—Use this option to create a folder when exporting to DOORS.
- ◆ **Create Project**—Use this option to create a project when exporting to DOORS.
- ◆ **Rename**—Use this option to rename a project or a folder when exporting to DOORS.

## Synchronizing the Document

Follow these steps to perform the export:

1. Make sure everything is configured in the **Export** dialog box.
2. Click **Export** to begin the synchronization. Rhapsody Gateway communicates with DOORS in order to check what needs to be updated.

An information window appears:

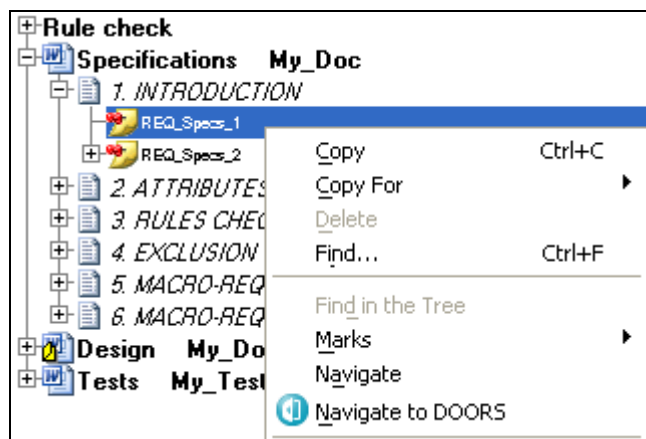


If it is the first upload, you will have all the imported document elements presented as “New elements”.

3. Click **Export** to launch the upload of the information into the DOORS database. Use the **Cancel** option to close the window without any action in DOORS.

## Navigating to DOORS

Once a document has been exported to DOORS, the feature **Navigate to DOORS** is available, as shown below:



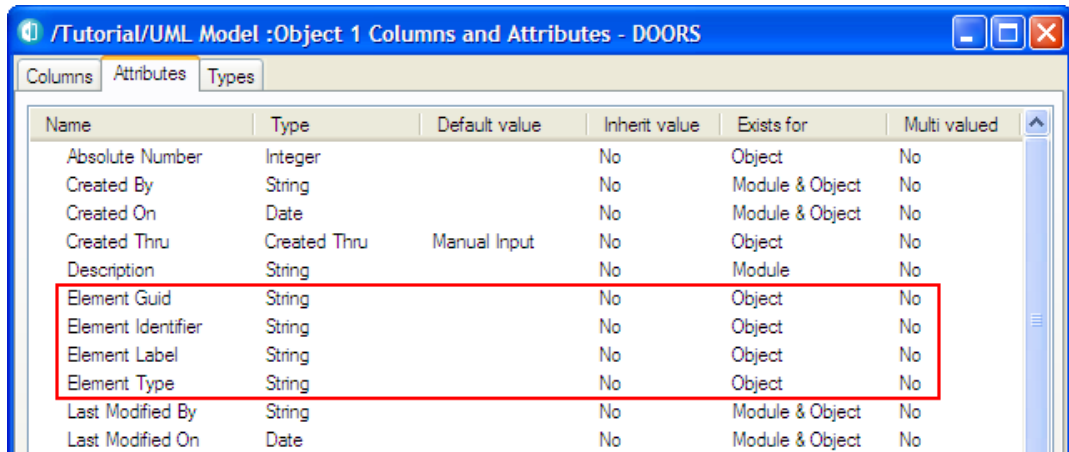
To navigate between Rhapsody Gateway and the documents or projects exported to DOORS, right click the **Navigate to DOORS** option in the main window, or select it in the **Edit** menu.

## Traceability and Links in DOORS

Rhapsody Gateway defines several ways to capture traceability information. One method uses **References** and **Links** in the type definition (see the *Customization Guide* and the *User Manual*). The traceability managed by Rhapsody Gateway is uploaded into DOORS by creating link modules. One link module is created in DOORS for each kind of **Reference** defined in Rhapsody Gateway for the type applied to the uploaded document.

Rhapsody Gateway creates the following three or four attributes for the imported DOORS module visible from the **Columns and Attributes** window:

- ◆ Element Guid (if the type manages Guid)
- ◆ Element Identifier
- ◆ Element Label
- ◆ Element Type



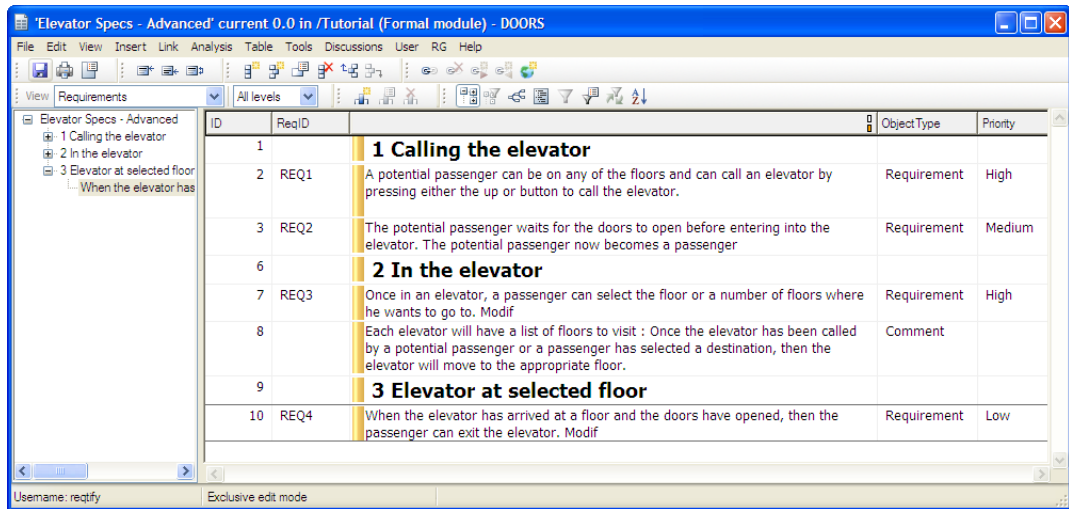
Name	Type	Default value	Inherit value	Exists for	Multi valued
Absolute Number	Integer		No	Object	No
Created By	String		No	Module & Object	No
Created On	Date		No	Module & Object	No
Created Thru	Created Thru	Manual Input	No	Object	No
Description	String		No	Module	No
Element Guid	String		No	Object	No
Element Identifier	String		No	Object	No
Element Label	String		No	Object	No
Element Type	String		No	Object	No
Last Modified By	String		No	Module & Object	No
Last Modified On	Date		No	Module & Object	No

These attributes are not supposed to be managed by the user; they are used by Rhapsody Gateway to manage objects.

The text elements captured by Rhapsody Gateway are uploaded into DOORS as **Object text** and diagrams are also inserted as **Object text**.

### Important note

The low level (uploaded) document is considered the master for low level information. A change in the DOORS **Object text** uploaded as text of a low level element will be erased by the next reload action and replaced by the new text element. The “best practice” is to make the modification in the tool used at low level, not in DOORS.



### Note

User defined attributes can be added in this created DOORS module. Additional links can originate from or lead to this module as well and Rhapsody Gateway will not modify them.

Rhapsody Gateway does not create empty modules, for instance, a user could have defined a reference element “Implements” in his type, but no “Implements” information is detected. In this case, the link module “Implements” will not be created (see the *Capturing Links* section).

## Attributes Export

All the Rhapsody Gateway attributes are created as DOORS attributes for the imported module.

For each new attribute a new column is created.

## Enumerated Attributes Export

Like other attributes, when enumerated values are declared in the Rhapsody Gateway attribute type, an enumerated attribute type will be created in DOORS.

Multi-values attributes from Rhapsody Gateway enumerated types, are also exported in DOORS. A specific column displays all the values.

ID	SDD_Like.doc	Font
1	<b>1 1. BASIC INFORMATION</b>	
2	<b>1.1 REQ_Design_1</b> Text of the requirement. The requirement here	bold italic

## Note

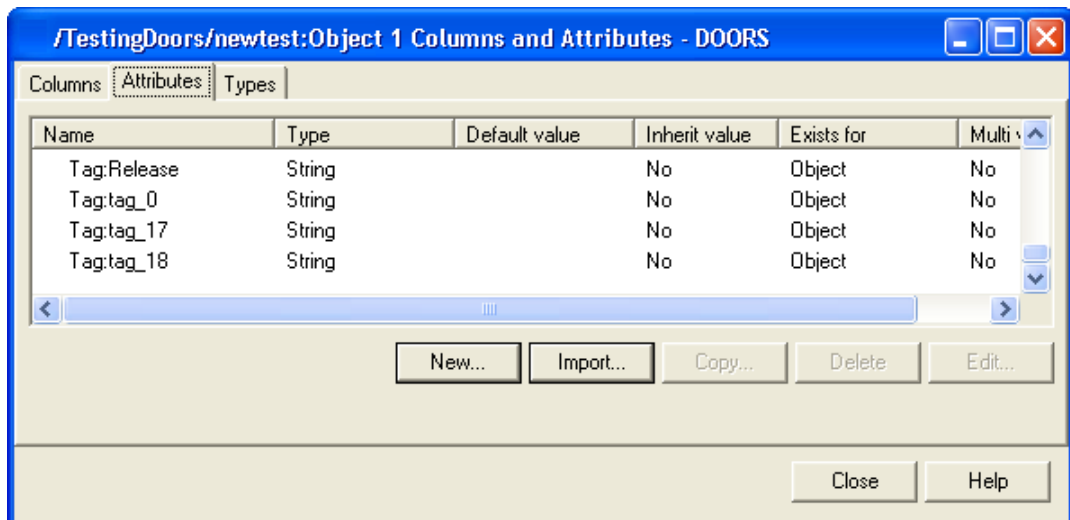
The export of multi-values attributes for non-enumerated types is not possible.

## Specific and Generic Attributes Export

A **generic attribute** is an attribute, which captures both an attribute name **label** and a value **identifier**. Other attributes are named **specific**.

If several specific attributes are captured from Rhapsody Gateway, only one attribute is created. This attribute is named <typeName>.

The behavior of a generic attribute export is different from the export of a specific attribute. Each instance of a generic attribute becomes a DOORS object attribute named <typeName>:<label>. It is initialized with the value captured in the **identifier** field. Below is an example of a new attribute creation with typeName=Tag.



Attributes are filled with the value captured for attributes by Rhapsody Gateway.

## Capturing Traceability Performed in DOORS

If links are created inside DOORS between a module and a lower-level document previously uploaded by Rhapsody Gateway, this traceability can be captured by Rhapsody Gateway if the applied type of analysis is defined to analyze these DOORS links.

## Capturing Traceability Performed within Rhapsody Gateway

Rhapsody Gateway allows you to create traceability links using the graphical view.

Traceability links are created in DOORS when you select **Export document to DOORS** from the **Tools** menu.

# Troubleshooting

---

The tasks in this section can help you solve the most common problems with DOORS usage.

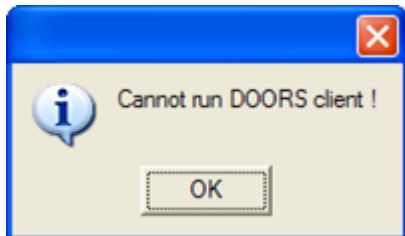
## Changing DOORS Version

If a user wants to change the version of the DOORS client, first he has to launch the other DOORS client version before launching the analysis.

## DOORS Client Cannot Run

The user may encounter troubles during his conversion or navigation to DOORS modules.

A box like the following one appears:



The possible cases for this trouble can be:

- ◆ the environment variable `DOORS_SCRIPT` does not exist and the following key is lacking in the register  
`HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{value}\LocalServer32`  
where {value} is the default value located under:  
`HKEY_CLASSES_ROOT\DOORS.Application\CLSID`
- ◆ the environment variable `DOORS_SCRIPT` does not point to a DOORS client.

To solve this problem ask your administrator to re-launch the DOORS client to update the register.

## DOORS Feature under Linux

The environment variable `DOORS_SCRIPT` should exist and should point to the DOORS client executable to let activate the DOORS feature under Linux.

## DOORS Installation Problem

An installation problem can appear if DOORS is installed after Rhapsody Gateway. To solve the problem, there are two solutions:

1. uninstall Rhapsody Gateway, and then install again Rhapsody Gateway
2. manually set the following "String value" under the "Config" key in the registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Telelogic\DOORS\<DOORS
Version>\Config => AddIns = <Your Rhapsody
Directory>\Gateway\config\misc\DOORS\rg;
```

## Setting a DOORS Server

The DOORS default server is stored in the registry at the following location:

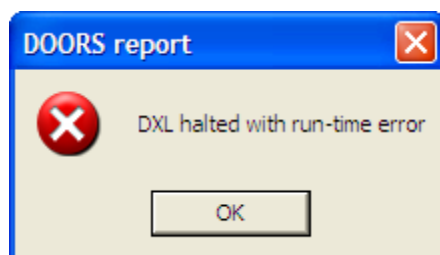
```
HKEY_LOCAL_MACHINE\SOFTWARE\Telelogic\DOORS\<DOORS Version>\Config
=> Data
```

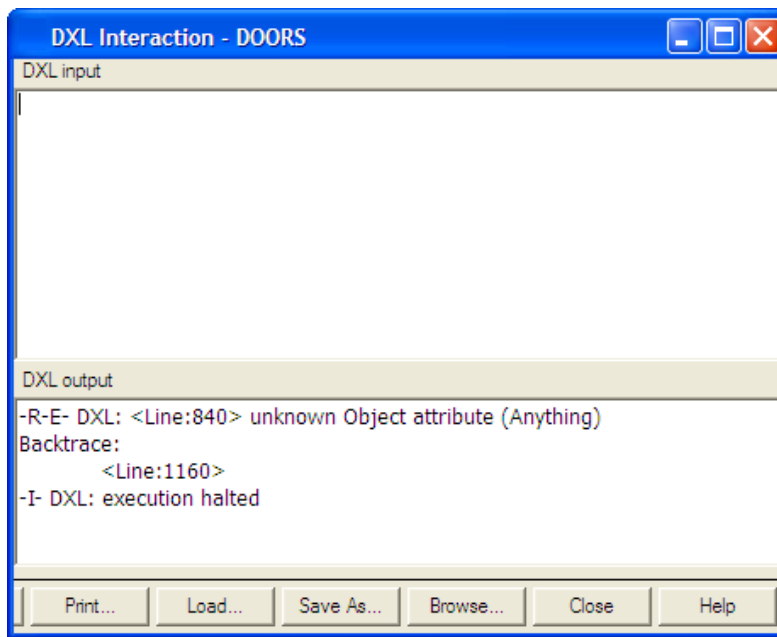
An easy way to change the DOORS Server is to set the `DOORS_SERVER` environment variable to define a new server path. Use the Rhapsody Gateway Options window to set the `DOORS_SERVER` variable. The server defined using this variable will be used instead of the default one.

If the above proposal does not solve the problem or if you need help for other problems do not hesitate to contact the Support Team.

## DXL Error: Unknown Attribute

This error can arise if a particular syntax for Requirement types name is used. In this case, the following dialog boxes are displayed:





The suffix of the Requirement type name is used by **Export to DOORS** as the 'identifier attribute' to detect requirements.

Consequently the syntax **Prefix.Suffix** for Requirement types name shall not be used if the Suffix does not aim at pointing out an attribute name.

For more information, please refer to the 'Elevator Specs – Advanced Module' example of this Coupling note.