prostep ivip
Documentation

Referring to ReqIF1.2

ReqIF Implementation Guide

Version 1.7, 16 December 2019

Status: Released

Abstract

This specification is an additional, informal document concerning the OMG ReqIF1.2 standard.

It answers implementation specific questions that have intentionally been left out of the OMG ReqIF standard because they do not directly concern the exchange format but may be relevant for tool implementations.

Disclaimer

This document is a prostep ivip Documentation (PSI Documentation), referring to ReqIF1.2. Those are freely available for all prostep ivip e.V. members. Anyone using these recommendations is responsible for ensuring that they are used correctly.

This PSI Documentation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using PSI Documentations must assume responsibility for his or her actions and acts at their own risk. The prostep ivip Association and the parties involved in drawing up the PSI Documentation assume no liability whatsoever.

We request that anyone encountering an error or the possibility of an incorrect interpretation when using the PSI Documentations contact the prostep ivip Association (psi-issues@prostep.com) immediately so that any errors can be rectified.

Copyright

I. All rights on this PSI Documentation, in particular the copyright rights of use and sale such as the right to duplicate, distribute or publish the Documentation remain exclusively with the prostep ivip Association and its members.

II. The PSI Documentation may be duplicated and distributed unchanged, for instance for use in the context of creating software or services.

III. It is not permitted to change or edit this PSI Documentation.

IV. A suitable notice indicating the copyright owner and the restrictions on use must always appear.

Contents

List of Figures

Abbreviations and Definitions

ReqIF:	Requirements Interchange Format
ReqIF tool:	A tool that exports ReqIF compliant XML documents from a source requirements authoring tool and/or imports them in a target requirements authoring tool.
Requirements authoring tool:	A tool used that is capable of creating and modifying requirements. In the context of this specification, this need not be a tool marketed as “Requirements Management Tool”.

1 Introduction

1.1 Motivation and target of this specification

This specification is an additional, informal document concerning the OMG ReqIF1.2 standard.

It is intended to be read by vendors who implement a tool for the OMG ReqIF1.2 standard.

The purpose of this specification is to provide recommendations on how to solve specific implementation related problems that have intentionally not been dealt with in the OMG ReqIF1.2 standard because they do not directly concern the exchange format but the handling of ReqIF XML documents by tools.

1.2 How to read this document

Each recommendation within this document is addressed in a sub-section of chapter 2.

2 Recommendations

The following sections describe non-normative recommendations for tools implementing the ReqIF standard. Implementing these recommendations is not required for standard compliance. However, adhering to the recommendations simplifies implementation and fosters interoperability to existing implementations.

2.1 Naming conventions for ReqIF file extensions

The following file extensions should be used for ReqIF files:

- .reqif for a single ReqIF XML file (instead of the usual .xml extension)
- .reqifz for ZIP archives containing ReqIF files and additional files, e.g. images (instead of the usual .zip extension). Do not use the.zip file extension for these files.

An example ReqIF XML file could therefore be called **example1.reqif**, an example ReqIF ZIP archive could be called **example2.reqifz**.

For.reqifz archives, ReqIF tools must use the standard ZIP data format, which is based on the Deflate Compression. ReqIF tools are not allowed to use some proprietary format of tools like 7-Zip.

Using these file extensions simplifies the task of locating ReqIF files in a file system tree, which is specifically important for ReqIF importing tools. ReqIF tools should nevertheless be able to process files with the usual extensions .xml and .zip.

2.2 Character Encoding of ReqIF XML files

In order to increase interoperability between ReqIF tools, the following conventions should be followed when exporting ReqIF XML files:

1. A ReqIF exporting tool should always use UTF-8 as the character encoding in ReqIF files. That means that the following line should be included as first line in every ReqIF file: `<?xml version="1.0" encoding="UTF-8"?>`
2. A ReqIF exporting tool should represent all special characters either by their proper encoding in UTF-8, or by their UTF-8 reference.

As an example for 2., the umlaut ä should either be represented

- in binary form (not as text): `11000011 10100100`
- or as a UTF-8 reference (as text in the ReqIF file): `&#228;`

2.3 Naming conventions for system attributes

Different requirements authoring tools may have different internal data models to store the requirements data. To exchange requirements between tools with different data models, a mapping between the tools' data models need to be established. While it is possible to establish a mapping on a tool by tool basis, this task becomes increasingly difficult when more and more additional tools are considered.

This chapter proposes conventions to provide a single mapping for common elements of requirements data models. Such a mapping – when implemented by several ReqIF tools - eliminates the need to map on a tool by tool basis. Note that however, no ReqIF tool can *rely* on another tool actually producing attributes as defined by this convention, as the conventions are non-normative.

The term "system attribute" is used here for any attribute of a requirement, specification or relation between requirements that are provided "out-of-the-box" by a requirements authoring tool, that is: the attribute does not need to be created by the user of that requirements authoring tool, but is managed by the tool itself.

The following table gives an overview of system attributes of commercial tools that are currently on the market. The table explains how to map these attributes to ReqIF. Note that the attribute names defined by this convention MUST be exported case-sensitive, but a ReqIF importing tool SHOULD be forgiving when the attribute name is not case-sensitive.

ReqIF Attribute level	IBM DOORS System Attribute Name	PTC Integrity System Attribute Name	Visure IRQA System Attribute Name	ReqIF exporter behaviour first export of specification (Step Export1)	ReqIF Attribute Name Representation during export (Step Export1)	ReqIF concept to store the Attribute Value during export (Step Export1)
SpecObject	*Absolute Number*	*Item ID*	*Code*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName** := **ReqIF.ForeignID**	**SpecObject::AttributeValueString::theValue**
SpecObject	*Created By*	*Created By*	*Author*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName := ReqIF.ForeignCreatedBy**	**SpecObject::AttributeValueString::theValue**
SpecObject	*Created On*	*Created Date*	*Element Creation Date*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionDate::longName := ReqIF.ForeignCreatedOn**	**SpecObject::AttributeValueDate::theValue**
SpecObject	*Created Thru*	n/a	n/a	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionEnumeration::longName := ReqIF.ForeignCreatedThru**	**SpecObject::AttributeValueEnumeration::theValue**
SpecObject	*Last Modified By*	*Modified By*	*Active Version Author*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName := ReqIF.ForeignModifiedBy**	**SpecObject::AttributeValueString::theValue**
SpecObject	*Last Modified On*	*Modified Date*	*Last Modification Date*	Use existing ReqIF concept	Use **SpecObject::lastChange** attribute	Fill value in **SpecObject::lastChange**
SpecObject	*Object Heading*	n/a	n/a	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionXHTML::longName** := **ReqIF.ChapterName**	**SpecObject::AttributeValueXHTML::theValue**
SpecObject	*Object Number*	n/a	n/a	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName := ReqIF.ChapterNumber**	**SpecObject::AttributeValueString::theValue**
SpecObject	*Object Short Text*	n/a	*Name*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionXHTML::longName** := **ReqIF.Name**	**SpecObject::AttributeValueXHTML::theValue**
SpecObject	*Object Text*	*Text*	*Description*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionXHTML::longName** := **ReqIF.Text**	**SpecObject::AttributeValueXHTML::theValue**
SpecObject	n/a	n/a	n/a	Create ReqIF AttributeDefinition/Value (if the user wants to explicitly transmit deleted objects).	**SpecObject::AttributeDefinitionBoolean::longName := ReqIF.ForeignDeleted** (If the datatype of the definition has a default value, it should be false. If the datatype has no default value, and the attribute is missing for a certain SpecObject, it means false. The importing ReqIF tool must import all objects as they are, including the attribute ReqIF.ForeignDeleted)	**SpecObject::AttributeValueBoolean::theValue**
SpecObject	n/a	*Text Attachments*	n/a	Do not export as ReqIF attribute, but reference the text attachment from the **ReqIF.Text** attribute to which they are attached.	n/a	n/a
SpecObject	n/a	n/a	*Associated Files*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName := ReqIF.AssociatedFiles**	**SpecObject::AttributeValueString::theValue** (If there is more than one associated file, separate file URLs by whitespace)
SpecObject	n/a	*Category*	n/a	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionEnumeration::longName := ReqIF.Category**	**SpecObject::AttributeValueEnumeration::theValue**
SpecObject	n/a	n/a	*Active Version*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName := ReqIF.ForeignRevision**	**SpecObject::AttributeValueString::theValue**
SpecObject	n/a	n/a	*Change Description*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName** := **ReqIF.ChangeDescription**	**SpecObject::AttributeValueString::theValue**
SpecObject	n/a	n/a	*Fit-Criteria*	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionString::longName** := **ReqIF.FitCriteria**	**SpecObject::AttributeValueString::theValue**
SpecObject	n/a	n/a	*Discussion*	n/a	n/a	n/a
SpecObject	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecObject::AttributeDefinitionXHTML::longName** := **ReqIF.Description**	**SpecObject::AttributeValueXHTML::theValue**

Figure 1: Naming conventions for system attributes 1/2

Specification	*Name*	*Document Short*	*Name*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionXHTML::longName** := **ReqIF.Name**	**Specification::AttributeValueXHTML::theValue**
Specification	*Prefix*	n/a	n/a	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName** := **ReqIF.Prefix**	**Specification::AttributeValueString::theValue**
Specification	*Created By*	*Created By*	*Author*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName := ReqIF.ForeignCreatedBy**	**Specification::AttributeValueString::theValue**
Specification	*Created On*	*Created Date*	*Creation Date*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionDate::longName := ReqIF.ForeignCreatedOn**	**Specification::AttributeValueDate::theValue**
Specification	*Last Modified By*	*Modified By*	*Active Version Author*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName := ReqIF.ForeignModifiedBy**	**Specification::AttributeValueString::theValue**
Specification	*Last Modified On*	*Modified Date*	*Active Version Date*	Use existing ReqIF concept	Use**Specification::lastChange** attribute	Fill value in **Specification::lastChange**
Specification	n/a	*State*	n/a	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionEnumeration::longName** := **ReqIF.ForeignState**	**Specification::AttributeValueEnumeration::theValue**
Specification	n/a	*Project*	n/a	Create ReqIF AttributeDefinition/Value	**Specification:AttributeDefinitionString::longName** := **ReqIF.Project**	**Specification::AttributeValueString::theValue**
Specification	n/a	*Document ID*	n/a	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName** := **ReqIF.ForeignID**	**Specification::AttributeValueString::theValue**
Specification	*Description*	*Shared Text*	*Description*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionXHTML::longName** := **ReqIF.Description**	**Specification::AttributeValueXHTML::theValue**
Specification	n/a	n/a	*Active Version*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName := ReqIF.ForeignRevision**	**Specification::AttributeValueString::theValue**
Specification	n/a	n/a	*Change Description*	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName** := **ReqIF.ChangeDescription**	**Specification::AttributeValueString::theValue**
Specification	n/a	n/a	TBD	Create ReqIF AttributeDefinition/Value	**Specification::AttributeDefinitionString::longName** := **ReqIF.ForeignBaseline** (this attribute is to inform human users, not to be processed by machines. It defaults to the tool internal "version" of the baseline. When exporting, the exporting ReqIF tool must allow the user to edit the value.)	**Specification::AttributeValueString::theValue**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionXHTML::longName** := **ReqIF.Name**	**SpecRelation::AttributeValueXHTML::theValue**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionString::longName := ReqIF.ForeignCreatedBy**	**SpecRelation::AttributeValueString::theValue**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionDate::longName := ReqIF.ForeignCreatedOn**	**SpecRelation::AttributeValueDate::theValue**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionString::longName := ReqIF.ForeignModifiedBy**	**SpecRelation::AttributeValueString::theValue**
SpecRelation	TBD	TBD	TBD	Use existing ReqIF concept	**UseSpecRelation::lastChange attribute**	**Fill value in SpecRelation::lastChange**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionString::longName := ReqIF.ForeignID**	**SpecRelation::AttributeValueString::theValue**
SpecRelation	TBD	TBD	TBD	Create ReqIF AttributeDefinition/Value	**SpecRelation::AttributeDefinitionXHTML::longName := ReqIF.Description**	**SpecRelation::AttributeValueXHTML::theValue**

Figure 2: Naming conventions for system attributes 2/2

As an example of how to read that table, consider its first line:

ReqIF Attribute level	IBM DOORS System Attribute Name	PTC Integrity System Attribute Name	Visure IRQA System Attribute Name	ReqIF exporter behaviour first export of specification	ReqIF Attribute Name Representation during export	ReqIF concept to store the Attribute Value during export
SpecObject	*Absolute Number*	*Item ID*	*Code*	Create ReqIF AttributeDefinition/ Value	**SpecObject:: AttributeDefinitionString:: longName** := **ReqIF.ForeignID**	**SpecObject:: AttributeValueString:: theValue**

The green color indicates the representation in ReqIF. The yellow, blue and red colors indicate representations in requirements authoring tools.

The line reads as follows (from left to right):

SpecObject	The convention defined in this line applies to SpecObject attributes (that is: to attributes of requirements).
Absolute Number	In the requirements authoring tool IBM DOORS there is a system attribute called *Absolute Number*, to which the convention in this line applies.
Item ID	In the requirements authoring tool PTC Integrity there is a system attribute called *Item ID*, to which the convention in this line applies.
Code	In the requirements authoring tool Visure IRQA there is a system attribute called *Code*, to which the convention in this line applies.
Create ReqIF AttributeDefinition/Value	In order to map any of the above requirements authoring tool attributes to ReqIF, create an AttributeDefinition instance and an AtttributeValue instance in ReqIF (when exporting a ReqIF file).
SpecObject:: AttributeDefinitionString:: longName:= **ReqIF.ForeignID**	The name of the created ReqIF AttributeDefinitionString instance is **ReqIF.ForeignID**. To do this in ReqIF, set the **longName** of the AttributeDefinitionString instance to the text **ReqIF.ForeignID**.
SpecObject:: AttributeValueString:: theValue	The value of the attribute is stored in **theValue** attribute of the **AttributeValueString** instance.

To summarize the line: the system attributes *Absolute Number, Item ID* and *Code* each represent the same concept, an identifier. Therefore, each of them is mapped to a string attribute with the same name in ReqIF. The name of that ReqIF attribute is **ReqIF.ForeignID**.

2.4 How to handle read-only, automatically set system attributes

2.4.1 Problem statement

It is relatively easy to import and export a system attribute that is **not** read-only in a requirements authoring tool: when importing, the value from the ReqIF XML document is stored in a requirements authoring tool attribute (e.g. the requirement‘s text), when exporting, this attribute’s value is stored in the ReqIF XML document.

However, there are certain system attributes that are read-only and automatically set by the requirements authoring tool when creating a requirement, a typical example is the creator of a requirement.

When importing a ReqIF XML document, these attributes’ values from the ReqIF XML document may get lost. For example, instead of importing the original creator of a requirement from the ReqIF XML document, the name of the importing person is automatically set in the requirements authoring tool.

2.4.2 Rules

The following rules should be applied in case there are automatically-set, read-only system attributes as described above and the resulting problem needs to be avoided.

R1. For each requirements authoring tool attribute, a ReqIF exporting tool should allow the user to choose whether to export the attribute or not, no matter if the attribute is a writable system attribute, a read-only system attribute or a user defined attribute.

R2. If the user chooses to export an automatically set, read-only system attribute, the attribute name in the ReqIF XML document should have the name as defined in clause 2.3, “Naming conventions for system attributes”. For example, the attribute for the creator of a requirement should be called **ReqIF.ForeignCreatedBy** in the XML document.

R3. Names of user defined attributes should always be represented in ReqIF XML documents and requirements authoring tools without a **ReqIF.** prefix.

R4. When importing a system attribute that has been exported according to R2 into a requirements authoring tool for the first time, the ReqIF importing tool should create a new user attribute in the requirements authoring tool. The name of this attribute is obtained by stripping the **ReqIF.** prefix. Example: A **ReqIF.ForeignCreatedBy** attribute in the ReqIF XML document should be called *ForeignCreatedBy* in the requirements authoring tool after import.

R5. When an attribute created due to R4 is exported again, R3 applies. For example: If the *ForeignCreatedBy* attribute in the R4 example is exported again, it has to be called **ForeignCreatedBy** in the ReqIF XML document, NOT **ReqIF.ForeignCreatedBy**. Note that exporting attributes again like this rarely makes sense, especially if only 2 partners exchange requirements, and it may cause conflicts, as described at the end of clause 2.4.3.

2.4.3 Typical exchange scenario

To explain the rules, Figure 3 depicts a typical exchange scenario of one requirement (Requirement X) between two partners, one OEM and one supplier. In the scenario, Requirement X has 2 attributes: *Status* is a user defined attribute and *Created By* is a read-only, automatically set system attribute.

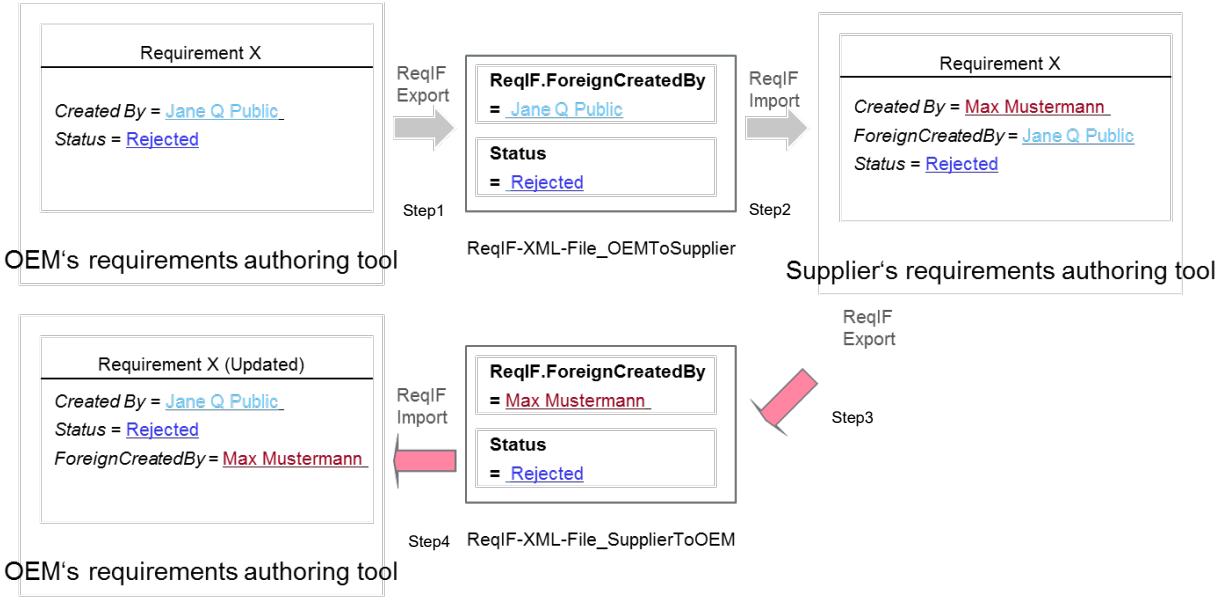

Figure 3: Typical exchange scenario for automatically set system attributes

What follows is an explanation of the 4 steps of the typical exchange scenario. At the end of the sentence, you’ll find the applied rule number in brackets.

Step1. Requirement X and its 2 attributes are exported from the OEM's authoring tool (R1). The *Created By* attribute is exported named **ReqIF.ForeignCreatedBy** in the ReqIF XML document (R2). The name of the *Status* attribute stays **Status** in the ReqIF XML document (R3).

Step2. Requirement X and its 2 attributes are imported to the supplier's authoring tool. The ReqIF importing tool creates a user attribute *ForeignCreatedBy* in the authoring tool and stores the value of the **ReqIF.ForeignCreatedBy** attribute (Jane Q. Public) from the XML document in it (R4). The ReqIF importing tool stores the value of the **Status** attribute from the XML document in a newly created *Status* user attribute (R3).

The *Created By* attribute in the supplier's authoring tool is a read-only, automatically set system attribute. Its value is not affected by the import. Instead, the supplier's authoring tool automatically sets its value to the name of the importing person - in the scenario that name is Max Mustermann - during creation of Requirement X.

Step3. Requirement X and two of its attributes are exported from the supplier's authoring tool (R1): the *Created By* attribute and the *Status* attribute. The same rules as in Step1 apply, that is: the *Created By* attribute is exported named **ReqIF.ForeignCreatedBy**, the *Status* attribute is exported named **Status**. Note that the *ForeignCreatedBy* attribute is not exported from the supplier's authoring tool in this scenario.

Step4. Requirement X and its 2 attributes are re-imported into the OEM's authoring tool. This step is analogous to Step2, only the authoring tool, the attribute values and the name of the importing person is different (Jane Q. Public).

Note that each **identifier** of ReqIF **Identifiable** elements MUST be immutable during consecutive exports and imports, as required by the standard. That means: in the **AttributeDefinitionString** elements for the attribute called **ReqIF.ForeignCreatedBy**, the **identifier** MUST be the same after Step1 and Step3, as they represent the same system attribute.

2.4.4 Different exchange scenario

Note also that rule R5 has not been applied in the typical exchange scenario, because in Step3, the *ForeignCreatedBy* attribute has not been exported from the supplier's authoring tool.

In a different scenario, the *ForeignCreatedBy* attribute might have been exported as well, named **ForeignCreatedBy** in the ReqIF XML document (R5). In that different scenario, the ReqIF XML document would contain two attributes for the creator, one called **ReqIF.ForeignCreatedBy** (with value Max Mustermann) as in the typical exchange scenario, one called **ForeignCreatedBy** (with value Jane Q. Public), both having different identifiers. It would be the responsibility of the next ReqIF importing tool in line to resolve the conflict between those 2 values.

2.5 How to interchange DOORS table information

IBM Rational DOORS allows the user to create tables as part of requirements' content. Up to RIF1.2, there has been no defined way how to exchange DOORS tables.

ReqIF has an attribute **isTableInternal** in the **SpecHierarchy** information type. This attribute must be set to true for all SpecHierarchy elements that are related to SpecObjects that are parts of the DOORS table. This includes the root node of the DOORS table.

Additionally, there needs to be an **alternative XHTML representation** of the table's content to allow tools that don't support requirement internal tables to easily process the ReqIF XML document.

The following sub clauses describe a concrete example of the mapping between DOORS tables and ReqIF.

2.5.1 Mapping the root node of the DOORS table

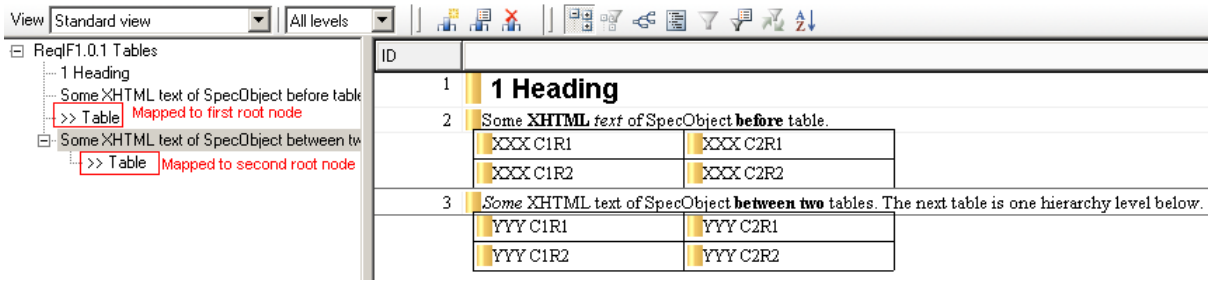

Figure 4: Screenshot of DOORS table (root node)

Each DOORS **>> Table** element is mapped to a **SpecHierarchy** element in ReqIF.

In the example: **2** DOORS tables result in **2 SpecHierarchy** elements for the roots of the tables.

2.5.2 Mapping the rows and colums of the DOORS table

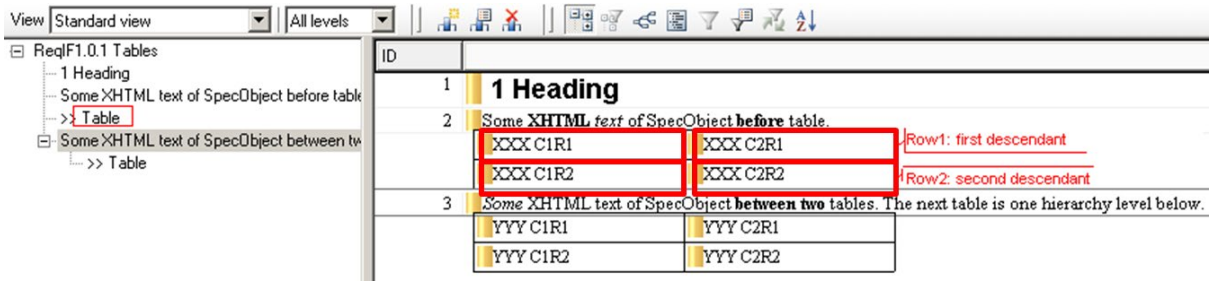

Figure 5: Screenshot of DOORS table (rows and columns)

Each DOORS table row (**Row1** and **Row2** on the right) is mapped to a ReqIF **SpecHierarchy** element. In addition, each DOORS table cell (**bold rectangles** on the right) is mapped to a ReqIF **SpecHierarchy** element.

In the example: The first DOORS table (**red** on the left) has **2** rows with **2** cells each.

Thus, **2 SpecHierarchy** elements need to be exported for the rows and additional **4 SpecHierarchy** elements need to be exported for the cells of this table.

2.5.3 Summary of the example mapping

The following table describes how to map the first DOORS table shown in the previous figures to ReqIF **SpecHierarchy** elements. The ReqIF standard dictates that each **SpecHierarchy** element must reference a **SpecObject** element, so the contents of the objects are shown additionally.

DOORS Structure	**ReqIF representation (including references to SpecObjects)**
>> Table	**SH i→ SpecObject[ReqIF.Text = "<table><tr>..</tr></table>"]**
Row 1	**SH i→ SpecObject**
Cell1	**SH i→ SpecObject[ReqIF.Text="XXX C1R1"]**
Cell2	**SH i→ SpecObject[ReqIF.Text="XXX C2R1"]**
Row 2	**SH i →SpecObject**
Cell1	**SH i→ SpecObject[ReqIF.Text="XXX C1R2"]**
Cell2	**SH i→ SpecObject[ReqIF.Text="XXX C2R2"]**

(**SH** means: **SpecHierarchy**, **i** means: with ‘**isTableInternal** set to true’,

→ means: ‘SpecHierarchy references SpecObject’.

[foo=bar] means: ‘SpecObject has attribute called **foo** with value **bar**’,

indentation explains parent-child-relationships of SpecHierarchy elements)

As you can see, the root **SpecHierarchy** element has a ReqIF attribute called **ReqIF.Text** with an XHTML value. This attribute value is used to store an **alternative XHTML representation** of the whole table. In the example, the contents of this attribute would look like this:

```
<table>
    <tr>
        <td> XXX C1R1 </td>
        <td> XXX C2R1 </td>
    </tr>
    <tr>
        <td> XXX C1R2 </td>
        <td> XXX C2R2 </td>
    </tr>
</table>
```

2.6 How to handle Embedded Objects

The ReqIF standard describes how embedded objects should be handled (in section 10.8.20). The aim of this chapter is to give a more detailed recommendation to simplify tool implementation.

2.6.1 Problem Statement

The ReqIF standard is clear on what elements should be present for embedded objects.

However, it lacks details on:

- The content of those elements
- The presentation of those elements.

The elements are:

- **Object**: A link to the object (typically a relative path)
- **Type**: A MIME-Type
- **Image**: Another embedded object of type image/png
- **Text**: An alternate text, which may be XHTML-Formatted

The following table captures the open issues:

Element	Content issues	Presentation Issues
Object	A tool may not be able to handle the content itself (i.e. embed it), but may hand it to the operating system. Does this count as "handling"?	If the object can be embedded, should image and text be hidden?
Type	The spec does not state what to do when the type is not known.	None
Image	What should the image show?	Should the image always be shown?
Text	There are no guidelines for the content	Should the alternate text always be shown?

Figure 6: Issues concerning embedded objects

2.6.2 General Considerations

1) The information regarding the embedded object must never be misleading or inconsistent. It is preferable to have information loss than to provide inconsistent information. (E.g. it is preferable to replace a screenshot image with a generic icon, if the screenshot does not reflect the object's content.)
2) If an embedded object that originated from a ReqIF import is not modified, all of its related information (mime-type, image, description) and the external file itself should be exported unchanged.
3) If an embedded object that originated from a ReqIF import is modified, its image and description must be updated (The mime-type may change, as described below.)
4) Embedded Objects of type image/png are special and do not need an (additional) image and description.

2.6.3 Rules

1) Embedded Objects must always have a mime type. If the type cannot be determined, it shall be set to application/octet-stream.
2) If the content of an object changes, then the tool must update the alternate text and the image. This will prevent these from containing information that is inconsistent with the new object content.
3) The mime type is allowed to change if (a) the type information is made more precise or (b) the content changes. Scenario (a) could be that the generic application/octet-stream type is changed into something more specific (e.g. application/msword). Scenario (b) could be that the content changes due to the tool chain on the target system. For instance, an OLE-Object could be unpacked, edited and embedded with a specific type (e.g. application/msword).
4) The image can fall in one of three categories: (a) a generic placeholder image (e.g. question mark); (b) an application-specific image (e.g. MS Word); (c) a screenshot of the content. Any of these is allowed, but implementors are encouraged to show as much information as possible.
5) While the placeholder text supports XHTML, this is discouraged. It should be plain text.
6) If a tool can embed the object, then the image and description need not be shown. Otherwise, either the image or the description must be shown (or both). It is recommended to provide the description text in the form of a tool tip that is associated with the image.
7) There are no restrictions to the size of the image. It is recommended for the rendering tool to shrink it if necessary.
8) A tool that exports ReqIF from a requirements authoring tools that supports OLE objects (for example: IBM DOORS) should export OLE objects either in their native MIME type's format (e.g. as Microsoft .doc-file with MIME-type application/msword) OR wrapped in RTF, as shown in the following example:

```
<xhtml:object data="files/powerpoint.rtf" height="96" type="application/rtf"
width="96">
```

```
    <xhtml:object data="files/powerpoint.png" height="96" type="image/png"
     width="96">
       This text is shown if alternative image can't be shown
    </xhtml:object>
  </xhtml:object>
```

2.7 Formatting conventions

This chapter defines conventions that SHOULD be used by ReqIF exporting tools for formatting to maximize interoperability.

If there is more than one solution for representing a specific way of formatting, ReqIF exporting tools SHOULD use the formatting mentioned in the ReqIF standard, or the common way of formatting in XHTML. For those cases, this prefered formatting is called "**Recommended**" in the column of the same name.

ReqIF importing tools MUST be able to deal with all kinds of formattings included in the XHTML modules mentioned in chapter 10.8.20 of the ReqIF1.2 standard. That means: ReqIF importing tools MUST support ALL elements shown below, with the exception of the style attribute with margin-left (for indentation). ReqIF importing tools MUST support any additional elements defined in the XHTML modules mentioned in ReqIF.

The following table shows the formatting conventions.

Kind	ReqIF.Text	Recommended	Description
Bold Text	**Bold**	Recommended	(standard element for bold in XHTML). Represented by using the <b> element.
Bold Text	**Bold**	---	Represented by using the element.
Italic Text	*Italic*	Recommended	Represented by using the <i> element.
Italic Text	*Italic*	---	Represented by using the element.
Italic Text	*Italic*	---	Represented by using the <cite> element.
Underlined Text	Underlined	Recommended	Represented by using the style attribute set to text-decoration:underline.
Underlined Text	Underlined	---	Represented by using the <ins> element.
Strike Through Text	~~Strike Through~~	Recommended	(recommended by ReqIF standard). Represented by using the style attribute set to text-decoration:line-through.
Strike Through Text	~~Strike Through~~	---	Represented by using the <del> element.
Superscript Text	Normal^{Superscript}	Recommended	(standard XHTML). Represented by using the <sup> element.
Subscript Text	Normal$_{Subscript}$	Recommended	(standard XHTML). Represented by using the <sub> element.
Indentation	The second line is indented, the third line is indented even further.	Recommended	Represented by using <blockquote> elements. Simple, straight-forward solution that is compatible with ReqIF. It explicitly uses XHTML elements and therefore can be easily processed by XML parsers.

Kind	ReqIF.Text	Recommended	Description
Indentation	The second line is indented 10 pixels, the third line is indented 20 pixels.	---	Represented by using the style attribute set to margin-left:<indentation>.
Unordered List	• First Bullet Point • Second Bullet Point • Third Bullet Point	Recommended	(standard XHTML). Represented by using the and elements.
Ordered List	1. Number One 2. Number Two 3. Number Three	Recommended	(standard XHTML). Represented by using the and elements.
Colored Text	Red	Recommended	(recommended by ReqIF standard). Represented by using the style attribute set to color:<color>

Additional note: A ReqIF exporting tool SHOULD NOT export XHTML headline elements (<h1> ... <h6>). Rather, the "plain text" of headlines SHOULD be represented by the ReqIF.ChapterName attribute, and the level of the headline SHOULD be represented by appropriately nested SpecHierarchy instances.

Here's the formatting conventions defined as an HTML file:

2.8 How to represent a conversation identifier

The ReqIF standard currently does not allow identifying which "conversation" (a.k.a. exchange process) a ReqIF XML document belongs to.

This could be solved by having an explicit "conversation identifier", that means: each exchange process gets its own, distinct identifier. It is an identifier that is created during the first export of a ReqIF XML document in a chain of exports and imports and is not changed by following imports and exports of ReqIF XML documents. To preserve the conversation identifier, each exported ReqIF file stores the conversation identifier.

As a recommendation on how to store such a conversation identifier, use a section in the ReqIF tool extensions analogous to this example:

```
<REQ-IF-TOOL-EXTENSION xmlns:reqif-common="http://www.prostep.org/reqif">
<reqif-common:EXCHANGE-CONVERSATION>
<reqif-common:IDENTIFIER>_6ced3339-9916-44f7-b9f0</reqif-common:IDENTIFIER>
</reqif-common:EXCHANGE-CONVERSATION>
</REQ-IF-TOOL-EXTENSION>
```

In that example, the conversation identifier is `_6ced3339-9916-44f7-b9f0`. Replace this value by a value created by your tool. If there are several ReqIF exchange files (.reqif-Files) in a ReqIF ZIP archive (.reqifz-File), all conversation identifiers contained in these ReqIF exchange files must be identical.

Note that as this is an extension to the capabilities of the ReqIF standard, it may be raised as an issue against a future version of the ReqIF specification.

2.9 How to represent links from/to external elements

The ReqIF standard contains the SpecRelation concept to represent relations between SpecObjects. Thus, links between requirements are possible. However, there is currently no way to represent links to elements that are external to the requirements authoring tool in ReqIF, like links to JIRA issues or UML modeling tool elements.

The convention described in this clause overcomes this restriction. Note that as this is an extension to the capabilities of the ReqIF standard, it may be raised as an issue against a future version of the ReqIF specification.

The following table shows the properties that are defined for external links by this convention:

Name	Mandatory	Description	Type	Possible Values	Interpretation if missing
SPEC-OBJECT-REF	x	The identifer of the SpecObject that is the source or the target of the external link	Identifier reference, as defined in xs:IDREF	<Any Identifier that is used as identifier of a SpecObject inside this ReqIF XML document>	
LONG-NAME	x	The name of the link.	xs:string	<Any String>	
URI	x	The identifier of the external item. As one example: the URI of a JIRA issue.	URI, as defined in xs:anyURI	<Any URI>	
DESC		Description of the link, as defined by the user of the requirements authoring tool	String, as defined in xs:string	<Any String>	<Empty String>, as user has entered no description
LINK-TYPE		The kind of link, as defined by the user of the requirements authoring tool. As one example: **verifies** for a link between a requirement and a test case.	String, as defined in xs:string For OSLC links, this **MUST** be the URI of the OSLC link type (see examples). For other link types, this may be any text (e.g. verifies).	<Any String>	Nothing specific is said about the link type.
DIRECTION		Out for links starting at a requirement and ending at an external item, In vice versa.	xs:enumeration	INWARDS, OUTWARDS	OUTWARDS is default
LINK-TECHNOLOGY		The link technology used. Currently, only OSLC is supported.		OSLC <Any String>	If missing or <Any String>: Tool gets content from URI through protocol in URI, presents it to user

The following XML example shows how to represent external links with the above properties in XML. Note that the same XML namespace as for the conversation identifier is used (see clause 2.8). Note also that the tool extensions described in this Implementation Guide may appear in any order in the XML document.

```
<REQ-IF-TOOL-EXTENSION xmlns:reqif-common="http://www.prostep.org/reqif">
    <reqif-common:SPEC-EXTENSIONS>
        <reqif-common:SPEC-OBJECT-EXTENSION>
            <reqif-common:SPEC-OBJECT>
                <reqif-common:SPEC-OBJECT-REF>_2ac39519-dfa8-4d9d-8da9-0fdc6107deb4</reqif-common:SPEC-OBJECT-REF>
            </reqif-common:SPEC-OBJECT>
            <reqif-common:EXTERNAL-LINKS>-
                <reqif-common:EXTERNAL-LINK DESC="MyDesc1" LONG-NAME="MyName1">
                    <reqif-common:URI>http://www.bbc.co.uk/news/uk</reqif-common:URI>
                    <reqif-common:LINK-TYPE>http://open-services.net/ns/rm#elaborates</reqif-common:LINK-TYPE>
                    <reqif-common:DIRECTION>OUTWARDS</reqif-common:DIRECTION>
                    <reqif-common:LINK-TECHNOLOGY>OSLC</reqif-common:LINK-TECHNOLOGY>
                </reqif-common:EXTERNAL-LINK>
                <reqif-common:EXTERNAL-LINK DESC="MyDesc2" LONG-NAME="MyName2">
                    <reqif-common:URI>http://www.bbc.co.uk/news/uk</reqif-common:URI>
                    <reqif-common:LINK-TYPE>http://open-services.net/ns/rm#elaborates</reqif-common:LINK-TYPE>
                    <reqif-common:DIRECTION>OUTWARDS</reqif-common:DIRECTION>
                    <reqif-common:LINK-TECHNOLOGY>OSLC</reqif-common:LINK-TECHNOLOGY>
                </reqif-common:EXTERNAL-LINK>
            </reqif-common:EXTERNAL-LINKS>
        </reqif-common:SPEC-OBJECT-EXTENSION>
        <reqif-common:SPEC-OBJECT-EXTENSION>
            <reqif-common:SPEC-OBJECT>
                <reqif-common:SPEC-OBJECT-REF>_7cc392a2-d1d0-4f3e-994c-c62ca4db2bc0</reqif-common:SPEC-OBJECT-REF>
            </reqif-common:SPEC-OBJECT>
            <reqif-common:EXTERNAL-LINKS>
                <reqif-common:EXTERNAL-LINK DESC="MyDesc3" LONG-NAME="MyName3">
                    <reqif-common:URI>http://www.bbc.co.uk/news/uk</reqif-common:URI>
                    <reqif-common:LINK-TYPE>http://open-services.net/ns/rm#elaborates</reqif-common:LINK-TYPE>
                    <reqif-common:DIRECTION>OUTWARDS</reqif-common:DIRECTION>
                    <reqif-common:LINK-TECHNOLOGY>OSLC</reqif-common:LINK-TECHNOLOGY>
                </reqif-common:EXTERNAL-LINK>
                <reqif-common:EXTERNAL-LINK DESC="MyDesc4" LONG-NAME="MyName4">
                    <reqif-common:URI>http://www.bbc.co.uk/news/uk</reqif-common:URI>
                    <reqif-common:LINK-TYPE>http://open-services.net/ns/rm#elaborates</reqif-common:LINK-TYPE>
                    <reqif-common:DIRECTION>OUTWARDS</reqif-common:DIRECTION>
                    <reqif-common:LINK-TECHNOLOGY>OSLC</reqif-common:LINK-TECHNOLOGY>
                </reqif-common:EXTERNAL-LINK>
            </reqif-common:EXTERNAL-LINKS>
        </reqif-common:SPEC-OBJECT-EXTENSION>
    </reqif-common:SPEC-EXTENSIONS>
</REQ-IF-TOOL-EXTENSION>
```

2.10 How to deal with RelationGroup elements

The ReqIF standards allows to export SpecRelation instances that are not contained in any RelationGroup instance.

For those “free floating” SpecRelation instances, it is unclear in which Specification the source and target SpecObject are contained in (as this would be defined by a RelationGroup).

In order to easily identify and locate the source and target object of a SpecRelation, the recommendation is:

It is mandatory to put a SpecRelation in a RelationGroup at least when the SpecRelation is going across the boundary of one Specification AND both the source and the target SpecObject are part of a Specification.

2.11 How to deal with RelationGroupType elements

Due to a bug in the current ReqIF XML schema, it is possible to assign a **RelationGroupType** (including **AttributeDefinition** elements) to a **RelationGroup** element, but it is not possible to assign **AttributeValue** elements to the RelationGroup.

As this may lead to inconsistencies, the current recommendation is not to add any **AttributeDefinition** elements to a **RelationGroupType** instance.

Note that the bug is a potential issue for a future version of the ReqIF specification.

2.12 How to specify a maximum length for string datatypes, when the requirements authoring tool does not limit string length

The `DatatypeDefinitionString` class in the ReqIF standard has a mandatory attribute `maxLength`. In its constraints, the ReqIF standard defines:

„The length of the string value held in any data element defined by DatatypeDefinitionString must not exceed the value of DatatypeDefinitionString::maxLength.“

This constraint holds, even if the requirements authoring tool that is the source of the export supports only string datatypes with unlimited lengths. In that case, ReqIF tools should export a sensible maximum value for `maxLength`, for example a value that makes sense from a technical point of view.

2.13 Attribute handling in database vs. document oriented requirements authoring tools

Requirements authoring tools differ in the way they handle requirement attribute sets.

In database-oriented tools like Siemens Teamcenter, each requirement may have a different set of attributes. In a document-oriented tool like IBM DOORS, all the objects in a formal module have the same attribute set.

This leads to problems when a ReqIF file is exported from a tool that supports multiple attribute sets (leading to multiple SpecObjectTypes in ReqIF), and imported in a tool that supports only one.

To avoid this problem, when a ReqIF importing tool encounters a ReqIF file containing *multiple* SpecObjectTypes, before importing the file into a tool that supports only *one* SpecObjectType, the tool must merge the attributes into one SpecObjectType. Later, during export, the ReqIF tool must restore the original, separate SpecObjectTypes again.

By doing this, information loss and unwanted modifications are prevented.

The recommendation for ReqIF tools to merge/unmerge the attributes is:

- During import, for each SpecObject:
    - The ReqIF tool stores each SpecObject’s identifier, its SpecObjectType and the list of its contained attributes.
    - If the same attribute name reoccurs in more than one SpecObjectType, but the type of the attribute differs, the ReqIF tool must rename the other occurrences of the attribute. (If the name and type are equal, the existing attribute must be reused.)
    - The ReqIF tool must populate an additional enumeration attribute with the names of the original object types and for each object, set it to the original object type (so that the users are aware which type was used in the ReqIF file and, thus, which attributes are available).
- During export, if the option to keep the SpecObjectType is enabled:
    - For each previously imported SpecObject, the ReqIF tool must:
        - Export the SpecObjectType and SpecObjectType identifier stored for it during import
        - Only export attributes that are part of the SpecObject’s stored SpecObjectType.
        - For attributes that are NOT part of the SpecObject’s stored SpecObjectType: if any of them has been changed by the user, warn the user.
        - Use the stored attribute identifier and attribute name on export, even in case it was renamed during import.

- For a new SpecObject (that a user created in the requirements authoring tool between import and export)
  - Assign the SpecObject's type according to the value of the enumeration attribute created during import.
  - The same rules apply as for exporting an existing SpecObject

2.14 How to deal with missing attribute values and default values

The OMG ReqIF standard leaves some room for interpretation when it comes to attribute values that are not present in the ReqIF file (while their attribute definition is).

When a ReqIF tool makes the initial import or an update, if an attribute value for an attribute definition that is part of a ReqIF file is missing for a SpecObject, that means:

- If a default value is defined for that attribute in the ReqIF file, set the attribute value in the target requirements authoring tool to that default value.
- If no default value is defined in the ReqIF file, set the attribute value in the target requirements authoring tool to "Nothing" (i.e. "unset" the attribute).

2.15 How to reimport a previously imported specification when requirements were deleted

In a roundtrip scenario it can happen that a supplier needs to import a specification again.

Maybe the customer deleted some requirements or didn't export them accidentally. These requirements are then missing in the second ReqIF file to be imported (compared to the previously imported ReqIF file).

For the supplier to know which requirements are missing, the importing tool should identify the differences between the ReqIF file and the existing specification, and mark the requirements as deleted in the specification in the target requirements authoring tool.

2.16 How to simplify XHTML-content

Chapter "10.8.20 AttributeValueXHTML", sub chapter "3. Handling information loss" of the ReqIF standard states about simplified attribute values:

"The purpose of the isSimplified attribute is to mark an AttributeValueXHTML element if an importing tool has been unable to interpret the formatted attribute value and thus create the possibility to inform users about it."

However, chapter 10.8.20 does not tell tool vendors how the content of the attribute value can actually be simplified as needed so that the importing requirements authoring tool can render it afterwards.

The following table contains a recommendation for such a transformation on a XHTML level. Tool vendors may adapt it to their needs if necessary.
For an overview of all the XML elements that are used by the ReqIF XHTML schema, see clause 3.3, "XML tags used in XHTML".

Topic	Description
Introduction	
	In general, the rules should be applied in three steps. 1. Apply the "Rules for logical markup" to physically represent logical markup, 2. Apply the "Rules for physical Markup", and

Topic	Description
	3. Apply the “Rules for Styles“ A ReqIF tool only needs to simplify content that can not be rendered in a adequate way in the requirements authoring tool or goes beyond the text formatting capabilities of the requirements authoring tool. Therefore, It may not be necessary to apply all of the following rules for a given step.
Rules for logical markup	
	The simplification algorithm shall convert any simple Datatype (bool, int, real, date) that is not supported in the target tool towards string.
	The simplification algorithm shall ignore the id or title attribute of a tag.
	The simplification algorithm shall convert a code formatted text to a monospaced font text. Therefore it is recommended to replace the starting and ending "Code" tags with the matching "tt" tags.
	The simplification algorithm shall convert a preformatted text to a paragraph. Therefore it is recommended to replace the starting and ending "pre" tags with the matching "p" tags.
	The simplification algorithm shall convert a section header to a paragraph. Therefore it is recommended to replace the starting and ending "h*" tags with the matching "p" tags.
	The simplification algorithm shall convert an unordered list to a paragraph. Therefore it is recommended to replace the starting and ending "ul" tags with the matching "p" tags.
	The simplification algorithm shall convert an ordered list to a paragraph. Therefore it is recommended to replace the starting and ending "ol" tags with the matching "p" tags.
	The simplification algorithm shall convert a definition list to a paragraph. Therefore it is recommended to replace the starting and ending "dl" tags with the matching "p" tags.
	The simplification algorithm shall convert a list item to a string with a following newline. Therefore it is recommended to replace the starting "li" tag with the innerText and the ending "li" tag with a newline (br) tag.
	The simplification algorithm shall convert a blockquote to a quotation mark embedded text. Therefore it is recommended to replace the starting and ending "blockquote" tags with a """ symbol.
	The simplification algorithm shall convert a quote with reference to a quotation mark embedded text. Therefore it is recommended to replace the starting and ending "q" tags with a """ symbol.
	The simplification algorithm shall convert a short quote to a quotation mark embedded text. Therefore it is recommended to replace the starting and ending "cite" tags with a """ symbol.
	The simplification algorithm shall convert a division to a paragraph. Therefore it is recommended to replace the starting and ending "div" tags with the matching "p" tags.
	The simplification algorithm shall convert a strong emphasis to a boldface. Therefore it is recommended to replace the starting and ending "strong" tags with the matching "b" tags.
	The simplification algorithm shall convert an emphasis font to an italic font. Therefore it is recommended to replace the starting and ending "em" tags with the matching "i" tags.

Topic	Description
	The simplification algorithm shall convert an object to a string with its filename. Therefore it is recommended to delete the "object" tags and replace it with the filename of the embedded object.
	The simplification algorithm shall convert an anchor to a string. Therefore it is recommended to replace the "a href" tag with the symbols "(->" and add the target behind it. After the target close the bracket with a ")"
	The simplification algorithm shall ignore an abbreviation. An abbreviation consists of "abbr" tags.
	The simplification algorithm shall ignore a sample text. A sample text consists of "samp" tags.
	The simplification algorithm shall ignore a keyboard text. A keyboard text consists of "kbd" tags.
	The simplification algorithm shall ignore a table head. A table head consists of "thead" tags.
	The simplification algorithm shall ignore a table body. A table body consists of "tbody" tags.
	The simplification algorithm shall ignore a table foot. A table foot consists of "tfoot" tags.
	The simplification algorithm shall ignore a column group. A column group consists of "colgroup" tags.
	The simplification algorithm shall ignore a column. A column consists of "col" tags.
	The simplification algorithm shall ignore a table caption. A table caption consists of "caption" tags.
	The simplification algorithm shall ignore a html body. A html body consists of "body" tags.
	The simplification algorithm shall ignore a html root element. A html root element consists of "html" tags.
	The simplification algorithm shall ignore an address information. An address information consists of "address" tags.
	The simplification algorithm shall ignore inserted content. Inserted content consists of "ins" tags.
	The simplification algorithm shall ignore an acronym. An acronym consists of "acronym" tags.
	The simplification algorithm shall ignore a variable. A variable consists of "var" tags.
	The simplification algorithm shall ignore a definition. A definition consists of "dfn" tags.
	The simplification algorithm shall ignore decreased font size. Decreased font size consists of "small" tags.
	The simplification algorithm shall ignore increased font size. Increased font size consists of "big" tags.
	The simplification algorithm shall ignore a parameter. A parameter consists of "param" tags.
	The simplification algorithm shall ignore a definition of a term in a definition list. A definition of a term consists of "dd" tags.
	The simplification algorithm shall ignore a definition term in a definition list. A definition term consists of "dt" tags.

Topic	Description
Rules for physical Markup	
	The simplification algorithm shall convert subscripted text to a special string. Therefore it is recommended to replace the start "sub" tag with "_(" and the end "sub" tag with ")"
	The simplification algorithm shall convert superscripted text to a special string. Therefore it is recommended to replace the start "sup" tag with "^(" and the end "sup" tag with ")"
	The simplification algorithm shall convert a paragraph to a newline. Therefore it is recommended to replace the ending "p" tag with a newline (br) tag and ignore the starting "p" tag.
	The simplification algorithm shall ignore a boldface font. A boldface font consists of "b" tags.
	The simplification algorithm shall ignore an italic font. An italic font consists of "I" tags.
	The simplification algorithm shall ignore an underlined font. An underlined font consists of "u" tags.
	The simplification algorithm shall ignore an monospaced font. An monospaced font consists of "tt" tags.
	The simplification algorithm shall convert a table to a paragrah. Therefore it is recommended to replace the starting and ending "table" tags with the matching "p" tags.
	The simplification algorithm shall convert a tablerow to a tabstop seperated text row. Therefore it is recommended to replace the ending "tr" tag with a newline (br) tag and the </td><td> tags with tabstopps and ignore the start tag (tr) of a table row.
	The simplification algorithm shall convert a tableheader to a tabstop seperated text row. Therefore it is recommended to replace the ending "th" tag with a newline (br) tag and the </td><td> tags with tabstopps and ignore the start tag (th) of a table header..
	The simplification algorithm shall convert a horizontal rule to a row of 40 hyphens with a following break. Therefore it is recommended to delete the starting and ending "hr" tag and replace it with 40 "-" symbols. After that it is recommended to add a newline (br) tag.
Rules for Styles	
	The simplification algorithm shall convert every unicode font family into a standard font. The tool may use a predefined font or ask the user to select one. Not unicode based fonts may cause wrong symbols. If the font-family is known to be not unicode (e.g. Symbol, Wingdings1, Wingdings2) a warning must be placed in front of the simplified text. It is recommended to place a warning in front of text from unclear font-families too. It is recommended to recognize Arial, Times New Roman, Helvetica and Sans Serif as unicode font-families and allow a user defined list for coporate design font-families.

2.17 How to deal with access policy data for SpecHierarchy elements

The ReqIF standard states in chapter 10.8.36, **SpecHierarchy**, constraint [5]:

„*If the set of editableAtts is empty for a SpecHierarchy element, the following constraint applies:*

If there is a parent SpecHierarchy element, the set of editable attributes is copied from the parent SpecHierarchy element. If there is no parent SpecHierarchy element, all attribute values for the SpecHierarchy are considered read-only"

The word "empty" may be ambiguous in the sentence above, so it is clarified here.

What “empty” means: if the set of editableAtts is **not present** in the ReqIF XML file.
What it does not mean is: if the set of editableAtts is **present, but empty** in the ReqIF XML file.
If the set was present, but empty, there would be no way to distinguish between the case that no attributes are editable and the case that attributes are inherited.
The wording may be changed in a future version of the ReqIF standard.

Furthermore, there is an issue concerning editableAtts inheritance: inheritance only works properly if the parent and the child SpecHierarchy’s SpecObject element have the same SpecObjectType, otherwise, what is inherited is undefined.
In case the parent SpecHierarchy’s SpecObject element has a different SpecType, exporting ReqIF tools should explicitly specifiy a set of editableAtts for the child SpecHierarchy element, in other words: **don’t use the editableAtts inheritance mechanism when the SpecType changes**.

2.18 How to resolve conflicting isEditable values for AttributeDefinition-Enumeration

The ReqIF standard defines this: if an AttributeDefinitionEnumeration has its isEditable flag set to false or the flag is omitted, the user of the requirements authoring tool is not allowed to modify its list of enumeration literals.

Concerning AttributeDefinitionEnumeration, the following conflict may occur as a consequence:

- one AttributeDefinitionEnumeration element (A) is editable
- a different AttributeDefinitionEnumeration element (B) is not editable
- both AttributeDefinitionEnumeration elements refer to the same DatatypeDefinitionEnumeration, which defines the enumeration literals

If that conflict occurs, it may not be clear whether the first AttributeDefinitionEnumeration element (A) should be treated as editable or not, as changes to the list of enumeration literals will be visible in the second AttributeDefinitionEnumeration element (B).

The recommendation to resolve this conflict is:

- AttributeDefinitionEnumeration (A) is editable, and changes to it are visible in AttributeDefinitionEnumeration (B).
- AttributeDefinitionEnumeration (B) cannot be edited itself.
- If this is not the desired behavior, two DatatypeDefinitionEnumeration elements need to be present in the ReqIF exchange file, one for (A), one for (B).

2.19 How to use xsi:schemaLocation in ReqIF XML documents

A ReqIF XML document can give a hint where to find the ReqIF XML schema to be used for validating by using the `xsi:schemaLocation` attribute. If it does so, it is recommended to use the absolute path of the ReqIF XML schema on the internet, as shown in the following example root element of a ReqIF XML document:

```
<REQ-IF xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.omg.org/spec/ReqIF/20110401/reqif.xsd
http://www.omg.org/spec/ReqIF/20110401/reqif.xsd"
xmlns="http://www.omg.org/spec/ReqIF/20110401/reqif.xsd" >
```

Note that is equally valid not to include a `xsi:schemaLocation` attribute at all.

Note as well that a tool that processes a ReqIF XML document is not obliged to use the XML schema at the specified location for validation.

3 Annex

3.1 Links to ReqIF1.2 example files

ReqIF1.2 test files can be found under:
https://hudson.eclipse.org/rmf/job/rmf.develop.luna/lastSuccessfulBuild/artifact/test-data/

The ReqIF example files can be opened with ProR, a tool that is part of the Eclipse project RMF and can be found at the following URL:

http://www.eclipse.org/rmf/

3.2 Differences to prior RIF/ReqIF versions

The following sections highlight the key changes from prior RIF/ReqIF versions. Note that while these sections give an overview, it is necessary for implementers to refer to the ReqIF1.2 specification for details and additional information.

3.2.1 Differences from OMG ReqIF1.0.1 to OMG ReqIF1.2

Only editorial changes have been performed, no changes to the ReqIF XML schema have been performed since ReqIF1.0.1. The issues that have been resolved can be found under:

http://www.omg.org/issues/reqif-rtf.html

3.2.2 Differences from RIF1.2 to ReqIF1.2

Change ID	*Change*	*Change affects*	*Rationale*
ReqIF-Chng-1	Removed <format> sub element from DatatypeDefinitionDate	XML Schema	In RIF1.2, the possibility to specify custom data formats has already been removed from the UML model, but not from the XML schema. As this is considered inconsistent, it is now removed from the schema as well.
ReqIF-Chng-2	Renamed the following XML elements: a) HEADER ------> THE-HEADER b) RIF-HEADER -----> REQ-IF-HEADER, c) RIF-CONTENT -----> REQ-IF-CONTENT d) RIF ------> REQ-IF e) RIF-TOOL-EXTENSION ------> REQ-IF-TOOL-EXTENSION	XML Schema / Specification	a) This is a reaction to the fact that some technologies (like JAXB) had problems compiling the schema. b) to e) Renaming is necessary due to the changed abbreviation "ReqIF"
ReqIF-Chng-3	There are new DatatypeDefinitions, AttributeDefinitions, and AttributeValues for XHTML content (DatatypeDefinitionXHTML, AttributeDefinitionXHTML, AttributeValueXHTML). They replace all other ways of including binary data in RIF1.2 The specification has been made more precise concerning the usage of MIME-Types for external objects and the representation of alternative images etc.	XML Schema / Specification	

Change ID	*Change*	*Change affects*	*Rationale*
ReqIF-Chng-4	Source and target specification of a RelationGroup are now both associated to the RelationGroup, there is no containment relation as it were between SpecGroup and a RelationGroup element any more. Also, the associations between SpecRelations and SpecObjects and the relations between RelationGroups and their source and target specification have been marked <<global>> with a stereotype. This means that SpecRelations can now cross file boundaries, in contrast to RIF1.2 (they use xsd:string as referencing type, instead of xsd:IDREF)	XML Schema / Specification	There has been a request to re-enable cross-file relationships in the new version of the specification.
ReqIF-Chng-5	Removed the attribute "author", introduced "reqIFToolId" and “repositoryId” (in ReqifHeader). This changes the order of XML elements (alphabetically).	XML Schema / Specification	As most RIF files are not created manually, it is more sensible to have an attribute that specifies the exporting RIF tool - separate from the "sourceToolID", which should be used for RM tool identifiers. This has also become necessary as there needs to be a way to track which ReqIF tool has performed a simplification and what repository the requirements originate from.
ReqIF-Chng-6	Added the mandatory attribute "isTableInternal" to SpecHierarchy elements.	XML Schema / Specification	
ReqIF-Chng-7	A new optional attribute has been added to SpecHierarchy (isSimplified) to handle information loss between different RM tools	XML Schema / Specification	
ReqIF-Chng-8	A new RIF information type has been introduced (AlternativeID) to enable migration of RIF1.1a IDs.	XML Schema / Specification	This provides a mechanism to migrate RIF1.1a Ids
ReqIF-Chng-10	The usage of the longName attribute has been specified rigidly, including making it the primary mechanism for specifying the name of enumeration literals, as described in the compliance requirements	Specification	
ReqIF-Chng-11	The its:dir attribute has been removed, as it can be replaced by using the style XHTML attribute	XML Schema / Specification	
ReqIF-Chng-12	The concepts “SpecGroup” and “SpecGroupHierarchy” have been removed. A “Specification” now acts as the root of a “SpecHierarchy”.	XML Schema / Specification	For some time, there has been discussion about the distinction of SpecGroup, SpecGroupHierarchy and SpecHierarchy. This is clarified in OMG ReqIF.
ReqIF-Chng-13	There is now a distinct SpecType and AttributeValueSimple for each simple attribute value.	XML Schema / Specification	This change greatly improves the validation capabilities of the ReqIF XML schema. ReqIF tools can now validate simple attribute values (bool, integer etc.) by using the XML schema instead of using custom validators.

3.2.3 Differences from RIF1.1a to ReqIF1.2

3.2.3.1 How ReqIF1.2 handles IDs and references and how to deal with RIF 1.1a-IDs

RIF1.1a:

In RIF1.1a, the “identifier“ attribute of „Identifiable“ had a „string“ type. References between RIF elements also had a “string” type. This allowed RIF1.1a tools to relate elements within a RIF XML file as well as elements from different files.

RIF1.2:

In RIF1.2, the “identifier“ attribute of „Identifiable“ got a „xsd:ID“ type. References between RIF elements got a “xsd:IDREF” type. This had the benefit that identifier uniqueness and references between elements could now be checked by the XML schema.

However, these new identifiers had the downside that they were not compatible with the RIF1.1a identifiers. Additionally, referencing elements via the xsd:IDREF mechanism became possible only between elements in the same XML document, so cross-XML-file referencing became difficult.

Both of these issues have been addressed in ReqIF1.2.

ReqIF1.2

In ReqIF1.2, the “identifier“ attribute of „Identifiable“ still has a „xsd:ID“ type to enable checking for uniqueness, which is identical to RIF1.2. Most references between ReqIF1.2 elements also use the xsd:IDREF mechanism of RIF1.2.

However, the exceptions are SpecRelation elements and RelationGroups. As links between requirements are the most common case where cross-XML-file dependencies are needed (for example between customer requirements and system requirements), a “global reference” may be specified in the XML file, meaning that the type of the reference is “xsd:string” instead of “xsd:IDREF”.

Note that the content of the reference does not change (it still references an “identifier” with type “xsd:ID”), only the type of the reference has been adapted to enable cross-file references which would otherwise be prevented by the xsd:IDREF mechanism).

To deal with legacy RIF1.1a IDs, ReqIF1.2 has the concept of an optional “AlternativeID”. (See chapter “10.2 Identification of elements” in the specification). An instance of AlternativeID stores an “identifier” attribute whose value must be equal to a RIF1.1a “identifier” attribute value in case of migration. Note that this RIF1.1a ID MUST NOT be referenced from a ReqIF element in any case. The “identifier” attribute of the “Identifiable” instance must be set to a RIF1.2 style identifier which may be different.

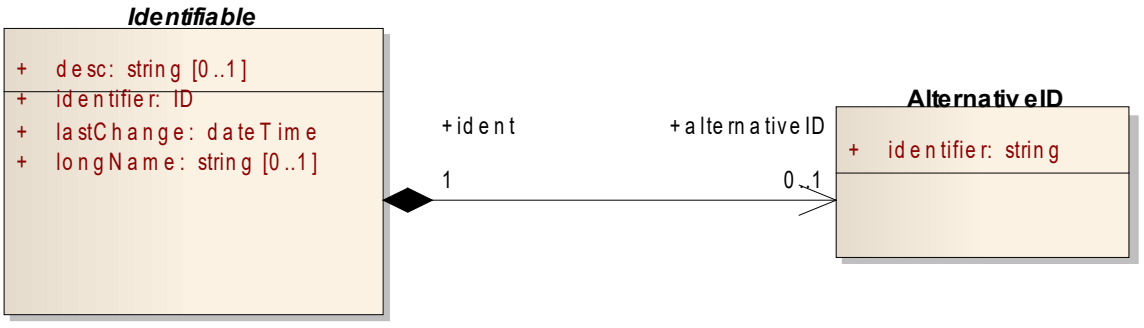

Figure 7: RIF1.1a “identifier” may be stored in "identifier" attribute of “AlternativeID” instance

3.2.3.2 Other differences from RIF1.1a to ReqIF1.2

Change ID	*Change*	*Change affects*	*Rationale*
ReqIF-Chng-1	Removed <format> sub element from DatatypeDefinitionDate	XML Schema	In RIF1.1a, there has been the possibility to specify a custom date format for the type DatatypeDefinitionDate. In ReqIF1.2, this possibility no longer exists.
ReqIF-Chng-2	Content split in three parts	XML Schema / Specification	In RIF1.1a, the RIF element acted as a container for all the other elements. In ReqIF1.2, the content is split in three parts: a header (THE-HEADER), the core content part (REQ-IF-CONTENT), and a tool specific part (REQ-IF-TOOL-EXTENSION)
ReqIF-Chng-3	There are new DatatypeDefinitions, AttributeDefinitions, and AttributeValues for XHTML content (DatatypeDefinitionXHTML, AttributeDefinitionXHTML, AttributeValueXHTML). They replace all other ways of including binary data in RIF1.1a The specification has been made more precise concerning the usage of MIME-Types for external objects and the representation of alternative images etc.	XML Schema / Specification	
ReqIF-Chng-4	Source and target specification of a RelationGroup are now both associated to the RelationGroup, there is no containment relation as it were between SpecGroup and a RelationGroup element any more. Also, the associations between SpecRelations and SpecObjects and the relations between RelationGroups and their source and target specification have been marked <<global>> with a stereotype. Only <<global>> relations can cross XML file boundaries.	XML Schema / Specification	In RIF1.1a, any kind of reference between RIF elements could cross XML file boundaries. In ReqIF1.2, this has been limited: only relations between requirements or specifications can cross XML file boundaries, while all other relations (for example: between a requirement and its type) are limited to the same XML file. This allows for validation of non-<<global>> relations using the XML ID/IDREF mechanism.
ReqIF-Chng-5	Removed the attribute "author", introduced "reqIFToolId" and “repositoryId” (in ReqifHeader). This changes the order of XML elements (alphabetically).	XML Schema / Specification	As most RIF files are not created manually, it is more sensible to have an attribute that specifies the exporting RIF tool - separate from the "sourceToolID", which should be used for RM tool identifiers. This has also become necessary as there needs to be a way to track which ReqIF tool has performed a simplification and what repository the requirements originate from.

Change ID	*Change*	*Change affects*	*Rationale*
ReqIF-Chng-6	Added the mandatory attribute "isTableInternal" to SpecHierarchy elements.	XML Schema / Specification	
ReqIF-Chng-7	A new optional attribute has been added to SpecHierarchy (isSimplified) to handle information loss between different RM tools	XML Schema / Specification	
ReqIF-Chng-8	A new RIF information type has been introduced (AlternativeID) to enable migration of RIF1.1a IDs.	XML Schema / Specification	
ReqIF-Chng-10	The usage of the longName attribute has been specified rigidly, including making it the primary mechanism for specifying the name of enumeration literals, as described in the compliance requirements	Specification	
ReqIF-Chng-11	The its:dir attribute has been removed, as it can be replaced by using the style XHTML attribute	XML Schema / Specification	
ReqIF-Chng-12	The concepts “SpecGroup” and “SpecGroupHierarchy” have been removed. A “Specification” now acts as the root of a “SpecHierarchy”.	XML Schema / Specification	For some time, there has been discussion about the distinction of SpecGroup, SpecGroupHierarchy and SpecHierarchy. This is more clear in ReqIF1.2.
ReqIF-Chng-13	There is now a distinct SpecType and AttributeValueSimple for each simple attribute value.	XML Schema / Specification	This change greatly improves the validation capabilities of the ReqIF XML schema. ReqIF tools can now validate simple attribute values (bool, integer etc.) by using the XML schema instead of using custom validators.

3.3 XML tags used in XHTML

The following table provides an overview of the XML tags that are used in the ReqIF XHTML schema. For more details on these topics, see the W3C page http://www.w3.org/TR/xhtml-modularization/.

XML Tag	Comments	in XHTML Module
 	Forced line break	Text Module
	Generic language/style container	Text Module
<object>	Generic embedded object	Object Module
<tt>	Teletype or monospaced text style	Presentation Module
<i>	Italic text style	Presentation Module
<b>	Bold text style	Presentation Module
<big>	Large text style	Presentation Module
<small>	Small text style	Presentation Module
	Indicates emphasis	Text Module
	Indicates stronger emphasis	Text Module
<dfn>	Indicates that this is the defining instance of the enclosed term	Text Module
<code>	Designates a fragment of computer code	Text Module
<q>	Short inline quotation	Text Module
<samp>	Designates sample output from programs, scripts, etc.	Text Module
<kbd>	Indicates text to be entered by the user	Text Module
<var>	Indicates an instance of a variable or program argument	Text Module
<cite>	Contains a citation or a reference to other sources	Text Module
<abbr>	Indicates an abbreviated form	Text Module
<acronym>	Indicates an acronym	Text Module
<sub>	Subscript	Presentation Module
<sup>	Superscript	Presentation Module
<ins>	Inserted text	Edit Module
<del>	Deleted text	Edit Module
<a>	Anchor	Hypertext Module
<h1>	Heading	Text Module
<h2>	Heading	Text Module
<h3>	Heading	Text Module
<h4>	Heading	Text Module
<h5>	Heading	Text Module
<h6>	Heading	Text Module
	Unordered list	List Module
	Ordered list	List Module
<dl>	Definition list	List Module
<pre>	Preformatted text	Text Module
<hr>	Horizontal rule	Presentation Module
<blockquote>	Long quotation	Text Module
<address>	Information on author	Text Module
<p>	Paragraph	Text Module
	Generic language/style container	Text Module
<table>	Used to describe tables	Basic Tables Module

XML Tag	Comments	in XHTML Module
	List item	List Module
<dt>	Definition term	List Module
<dd>	Definition description	List Module
<param>	Named property value	Object Module
<caption>	Table caption	Basic Tables Module
<tr>	Table row	Basic Tables Module
<th>	Table header cell	Basic Tables Module
<td>	Table data cell	Basic Tables Module
	Reading direction	Bi-directional Text Module