Requirements Interchange Format (ReqIF)

Version 1.1

OMG Document Number: formal/2013-10-01
Standard document URL: http://www.omg.org/spec/ReqIF/1.1
Machine Consumable Files:
  Normative: http://www.omg.org/spec/ReqIF/20101201/reqif.cmoF
  http://www.omg.org/spec/ReqIF/20110401/reqif.xsd
  http://www.omg.org/spec/ReqIF/20110402/driver.xsd/
USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.
PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™, Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, MOF™, OMG Interface Definition Language (IDL)™, and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.
COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials. Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.
OMG’s Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).
Table of Contents

Preface........................................................................................................................................... v

1 Scope .............................................................................................................................................. 1
   1.1 Who should read this document? ......................................................................................... 1
   1.2 Objectives of the Requirements Interchange Format .......................................................... 1

2 Conformance .................................................................................................................................. 2

3 Normative References .................................................................................................................. 3
   3.1 Normative References .......................................................................................................... 3
   3.2 Non-normative references .................................................................................................... 4

4 Terms and Definitions ................................................................................................................... 4

5 Symbols ......................................................................................................................................... 5

6 Additional Information .................................................................................................................. 6
   6.1 How to read this specification ............................................................................................... 6
   6.2 Acknowledgements ............................................................................................................... 6
     6.2.1 Submitting Organizations .............................................................................................. 6
     6.2.2 Supporting Organizations ............................................................................................. 7

7 Concept Overview and Use Cases .............................................................................................. 9
   7.1 Preface: How requirements authoring tools handle information ......................................... 9
   7.2 How the Requirements Interchange Format handles information from requirement authoring tools .................................................................................................................. 11
   7.3 How the Requirements Interchange Format copes with different tool capabilities ................................................................................................................................. 12
   7.4 Exchange Scenarios ............................................................................................................. 13
     7.4.1 Role descriptions ........................................................................................................... 14
     7.4.2 First exchange scenario (“One-Way”) .......................................................................... 14
     7.4.3 Second exchange scenario (“Roundtrip”) .................................................................... 15
   7.5 Detailed Use Cases ............................................................................................................... 17
     7.5.1 Use Case Overview ...................................................................................................... 17
     7.5.2 Use Case Specifications ............................................................................................... 17

8 Abstract Architecture .................................................................................................................... 21
9 Exchange Document Structure ................................................................. 25
  9.1 Class Descriptions ............................................................................. 25
    9.1.1 ReqIF ....................................................................................... 25
    9.1.2 ReqIFContent ........................................................................ 26
    9.1.3 ReqIFHeader .......................................................................... 26
    9.1.4 ReqIFToolExtension ............................................................... 28

10 Exchange Document Content ............................................................... 29
  10.1 Overview ...................................................................................... 29
  10.2 Identification of Elements ............................................................. 29
  10.3 Specifications, Requirements, and Attributes ..................................... 30
  10.4 Hierarchical Structuring of Requirements in a Specification
      and Requirement Relations .......................................................... 32
  10.5 Representing Attribute Data Types .................................................. 34
    10.5.1 Representing Data Types ....................................................... 34
    10.5.2 Relating Attributes to Data Types .......................................... 34
  10.6 Concrete Data Types ...................................................................... 35
    10.6.1 Simple Data Types ................................................................. 35
    10.6.2 Enumeration Data Type ......................................................... 36
    10.6.3 Data Type for XHTML Content ............................................. 37
  10.7 Access Restrictions ......................................................................... 37
  10.8 Class Descriptions ........................................................................ 38
    10.8.1 AccessControlledElement ..................................................... 38
    10.8.2 AlternativeID ......................................................................... 39
    10.8.3 AttributeDefinition ................................................................. 40
    10.8.4 AttributeDefinitionBoolean .................................................. 41
    10.8.5 AttributeDefinitionDate ......................................................... 42
    10.8.6 AttributeDefinitionEnumeration .......................................... 43
    10.8.7 AttributeDefinitionInteger .................................................... 44
    10.8.8 AttributeDefinitionReal ......................................................... 45
    10.8.9 AttributeDefinitionSimple .................................................... 46
    10.8.10 AttributeDefinitionString .................................................... 47
    10.8.11 AttributeDefinitionXHTML .................................................. 48
    10.8.12 AttributeValue ................................................................. 48
    10.8.13 AttributeValueBoolean ....................................................... 49
    10.8.14 AttributeValueDate ............................................................. 50
    10.8.15 AttributeValueEnumeration ................................................. 51
    10.8.16 AttributeValueInteger ........................................................ 52
    10.8.17 AttributeValueReal ............................................................. 53
    10.8.18 AttributeValueSimple ........................................................ 53
    10.8.19 AttributeValueString ........................................................ 54
    10.8.20 AttributeValueXHTML ....................................................... 55
10.8.21 DatatypeDefinition ............................................................. 58
10.8.22 DatatypeDefinitionBoolean ............................................... 59
10.8.23 DatatypeDefinitionDate ...................................................... 60
10.8.24 DatatypeDefinitionEnumeration ....................................... 60
10.8.25 DatatypeDefinitionInteger ............................................... 61
10.8.26 DatatypeDefinitionReal .................................................... 62
10.8.27 DatatypeDefinitionSimple ............................................... 63
10.8.28 DatatypeDefinitionString ............................................... 64
10.8.29 DatatypeDefinitionXHTML ........................................... 65
10.8.30 EmbeddedValue .............................................................. 66
10.8.31 EnumValue ................................................................. 66
10.8.32 Identifiable ................................................................. 67
10.8.33 RelationGroup ............................................................. 68
10.8.34 RelationGroupType ....................................................... 69
10.8.35 ReqIFContent ............................................................. 70
10.8.36 SpecHierarchy ............................................................. 71
10.8.37 Specification .............................................................. 73
10.8.38 SpecificationType ........................................................ 73
10.8.39 SpecObject ............................................................... 74
10.8.40 SpecObjectType ........................................................... 75
10.8.41 SpecRelation ............................................................... 76
10.8.42 SpecRelationType ........................................................ 77
10.8.43 SpecType ................................................................. 78
10.8.44 XhtmlContent ............................................................. 78

11 Production Rules of ReqIF XML Schema ................................... 81

11.1 Purpose ................................................................. 81
11.2 Notation for EBNF .......................................................... 81
11.3 Tags ................................................................. 81
11.4 EBNF ............................................................... 84

Annex A: Implementation Guide ................................................. 95
Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG’s specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG’s specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL:

http://www.omg.org/spec

Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

• CORBA/IIOP
• Data Distribution Services
• Specialized CORBA

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

• UML, MOF, CWM, XMI
• UML Profile

Modernization Specifications
Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

- CORBAServices
- CORBAFacilities

OMG Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

All of OMG’s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to http://www.omg.org/report_issue.htm.
1 Scope

1.1 Who should read this document?

This document is created to inform:

- Persons interested in exchanging requirements data between organizations that do not have a possibility to share the same repository (See Clause 4 for a definition of “repository”).
- Requirements authoring tool vendors who want to support the Requirements Interchange Format (ReqIF) with export and import interfaces for their requirements authoring tools. See Clause 4 for a definition of “requirements authoring tool.”
- Tool vendors other than requirements authoring tool vendors who wish to interchange requirements for documentation or other purposes.
- Anyone interested in defining, interchanging, storing, etc., requirements in a standard interchange format.

1.2 Objectives of the Requirements Interchange Format

Requirements management has been an integral part of the development process in various industries (especially in the military, aeronautical, or the medical device industry) for years. Other industries have been adopting requirements management recently.

The automotive industry for example introduced requirements management around 1999. As requirements management spread in the automotive industry over the years, more and more car manufacturers and suppliers have been applying requirements management and making use of dedicated requirements authoring tools. Large improvements have been made in these organizations and requirements management has been established as a key discipline in this collaborative engineering environment. Now with this established discipline in place, manufacturers and suppliers strive for collaborative requirements management where requirements management does not stop at company borders.

For technical and organizational reasons, two companies in the manufacturing industry are rarely able to work on the same requirements repository and sometimes do not work with the same requirements authoring tools. A generic, non-proprietary format for requirements information is required to cross the chasm and to satisfy the urgent industry need for exchanging requirement information between different companies without losing the advantage of requirements management at the organizations’ borders.

With the help of a dedicated interchange format for requirements specifications, it is possible to bridge the gap:

- The collaboration between partner companies is improved by the benefits of applying requirements management methods across company borders.
- The partner companies do not have to use the same requirements authoring tool and suppliers do not need to have multiple requirements authoring tools to fulfill the need of their customers with regards to compatibility.
- Within a company, requirement information can be exchanged even if various tools are used to author requirements.

The Requirements Interchange Format (ReqIF) described in this specification defines such an open, non-proprietary exchange format. Requirement information is exchanged by transferring XML documents that comply to the ReqIF format.

See the following figure for an example scenario between two partners who are exchanging a Customer Requirements Specification and the corresponding System Requirements Specification.
Figure 1.1 - Example ReqIF exchange scenario

Figure 1.1 represents a common scenario how requirements specifications are exchanged between partners. Both partners in the scenario use different requirements management (RM) tools to create, manage, and evolve their requirements specifications. The process is usually initiated by Partner 1. Customer requirements that are relevant for Partner 2 are consolidated in a snapshot document. The Partner 2 specific CRS snapshot is exported out of the RM-Tool A by means of the ReqIF-Exporter and transferred asynchronously to Partner 2 via existing data transfer mechanisms. The result of the export is a ReqIF compliant XML document representing the specific CRS snapshot. The data transfer mechanism is out of scope of ReqIF. Having received the exported CRS snapshot Partners 2 imports the information into RM-Tool B in order to analyze the customer requirements imposed by Partner 1. For traceability reasons Partner 2 links the received customer requirements with the corresponding system requirements. As an answer to the customer requirements Partner 2 creates a consolidated SRS snapshot that contains the system requirements realizing the imposed customer requirements of Partner 1. The SRS snapshot is fed back to Partner 1 as an exported ReqIF compliant XML document. Having imported the SRS snapshot Partner 1 can analyze within RM-Tool A how the customer requirements are fulfilled by the system requirements specified by Partner 2. As specifications evolve over time the exchange via ReqIF is an event driven, asynchronous data exchange.

2 Conformance

A technology targeting the seamless information exchange between a wide variety of tool implementations may tolerate only a very limited variability in the definition of the information exchange format.

Therefore, a compliant implementation of the Requirements Interchange Format (ReqIF) must implement all elements described in Clauses 9, 10, and 11. Further, a compliant implementation must also recognize and support the high-level exchange protocol and associated exchange document states defined in Clause 8.
As a compliance variation point, compliant implementations may use an alternative element identification mechanism in parallel to the primary identification mechanism. Further, implementations may be unable to interpret or handle certain forms of formatted attributes. In this case, implementations are allowed to substitute the offending representation with a simplified form, as long as the attribute is marked as simplified, a reference to an original form of the attribute is preserved, and the simplified attribute is excluded from any further alterations.

3 Normative References

3.1 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification:

URI

  http://www.ietf.org/rfc/rfc2396.txt

XHTML 1.1 Modularization

- XHTML™ Modularization 1.1, Daniel Austin et al., eds., W3C, 8 October 2008
  http://www.w3.org/TR/xhtml-modularization/

XML 1.0 (Second Edition)

  http://www.w3.org/TR/REC-xml

XML-Namespace

- Namespaces in XML, Tim Bray et al., eds., W3C, 14 January 1999
  http://www.w3.org/TR/REC-xml-names

XML-Schema

The authoritative description of the Requirements Interchange Format exchange document structure is provided as an XML Schema. XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents.

  http://www.w3.org/TRA/xmlschema-1/
  http://www.w3.org/TRA/xmlschema-2/
3.2 Non-normative references

MIME Media Types
http://www.iana.org/assignments/media-types/

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editable</td>
<td>The characteristic of an object that it is possible to define, alter, adapt, or refine the object.</td>
</tr>
<tr>
<td>Exchange XML Document</td>
<td>An XML document with specification content that is exchanged between two partners.</td>
</tr>
<tr>
<td>Exporting ReqIF tool</td>
<td>A ReqIF Tool that is used to export requirements information from a requirements authoring tool into an exchange XML document.</td>
</tr>
<tr>
<td>&lt;Name of ReqIF Information Type&gt; instance</td>
<td>An instance of the ReqIF information type, or an instance of a direct or indirect subclass of the ReqIF information type.</td>
</tr>
<tr>
<td>Importing ReqIF tool</td>
<td>A ReqIF Tool that is used to import an exchange XML document into a requirements authoring tool.</td>
</tr>
<tr>
<td>Information Element</td>
<td>An information element is an atomic unit of information, e.g., a requirements text, an attribute value in a requirements authoring tool, a relation that links two requirements.</td>
</tr>
<tr>
<td>Information Type</td>
<td>An information type is a category of information elements with the same properties in terms of e.g., element attributes or relationships to other information elements.</td>
</tr>
<tr>
<td>Links</td>
<td>References between requirements or between requirement and solution.</td>
</tr>
<tr>
<td>MIME type</td>
<td>Multipurpose Internet Mail Extensions type. MIME-Types are a common mechanism to specify kinds of textual or binary objects.</td>
</tr>
<tr>
<td>Repository</td>
<td>Container for RE&amp;M data that is managed by a requirements authoring tool.</td>
</tr>
<tr>
<td>Requirement</td>
<td>A requirement specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve.</td>
</tr>
<tr>
<td>ReqIF</td>
<td>Requirements Interchange Format: The format specified in this standard.</td>
</tr>
<tr>
<td>ReqIF model</td>
<td>The ReqIF-model is located on the same abstraction level as information types. It specifies and describes the different kind of information types as well as their relationships to each other.</td>
</tr>
<tr>
<td>ReqIF tool</td>
<td>A tool that exports ReqIF compliant XML documents from a source requirements authoring tool and/or imports them in a target requirements authoring tool.</td>
</tr>
</tbody>
</table>
For the purposes of this specification, the following acronyms and abbreviations apply.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDATA</td>
<td>Character Data</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HIS</td>
<td>Hersteller Initiative Software (<a href="http://www.automotive-his.de/">http://www.automotive-his.de/</a>). The Hersteller Initiative Software is a consortium of the vehicle manufacturers Audi, BMW, Daimler, Porsche, and Volkswagen. The objective of this consortium is to bundle their activities for standard software modules, process maturity levels, software test, software tools, and programming of control units. The common goal is to achieve and use joint standards.</td>
</tr>
<tr>
<td>IT</td>
<td>Information Type</td>
</tr>
<tr>
<td>MIME type</td>
<td>Multipurpose Internet Mail Extensions type.</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group:</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements Engineering</td>
</tr>
<tr>
<td>RE&amp;M</td>
<td>Requirements Engineering &amp; Management</td>
</tr>
<tr>
<td>ReqIF</td>
<td>Requirements Interchange Format.</td>
</tr>
<tr>
<td>RM</td>
<td>Requirements Management</td>
</tr>
<tr>
<td>UC</td>
<td>Use Case</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
</tbody>
</table>
6  Additional Information

6.1  How to read this specification

Clauses 1 to 6 contain background and basics for reading this specification. Clause 1 describes the objectives of this specification and the intended readership. Clause 2 defines conformance. Clause 3 lists other specifications and documents containing provisions which, through reference in this text, constitute provisions of this specification. Clause 4 and 5 contain definitions of terms and abbreviations used in this document. Clause 6 provides additional information to this specification.

Clauses 7 to 11 include the technical part of this specification. Clause 7 gives an introduction to the Requirements Interchange Format and describes relevant exchange scenarios. Clause 8 describes the abstract architecture of the ReqIF information model. Clause 9 defines the general structure of exchange XML documents. Clause 10 defines the details of the exchange XML documents. Clause 11 contains the production rules for theReqIF XML Schema.

6.2  Acknowledgements

The following companies submitted and/or supported parts of this specification:

6.2.1 Submitting Organizations

The following companies are formal submitting members of OMG:

- Atego
- ProSTEP iViP Association

6.2.2 Supporting Organizations

The following organizations support this specification, but are not formal submitters.

<table>
<thead>
<tr>
<th>Name of company</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>88solutions Corporation</td>
<td><a href="http://88solutions.com/">http://88solutions.com/</a></td>
</tr>
<tr>
<td>Audi AG</td>
<td><a href="http://www.audi.de">http://www.audi.de</a></td>
</tr>
<tr>
<td>BMW AG</td>
<td><a href="http://www.bmw.de">http://www.bmw.de</a></td>
</tr>
</tbody>
</table>
The initial work on ReqIF was done by the members of the HIS group and additional partners that were associated for this project. The HIS group is the panel of the vehicle manufacturers Audi AG, BMW Group, Daimler AG, Porsche AG, and Volkswagen AG to bundle their activities for standard software modules, process maturity levels, software test, software tools, and programming of control units. The common goal is to achieve and facilitate joint standards. The group that is working on the initial release of ReqIF consists of the ProSTEP iViP Association, Atego Systems GmbH, Audi AG, BMW AG, Continental AG, Daimler AG, HOOD GmbH, International Business Machines, MKS GmbH, PROSTEP AG, Robert Bosch GmbH, and Volkswagen AG.

Before the submission of the Requirements Interchange Format (ReqIF) to the OMG, the Requirements Interchange Format has been a specification proposed by the HIS and in its latest version, a recommendation of ProSTEP iViP. For these versions, the abbreviation “RIF” has been applied. The HIS released the Requirements Interchange Format as RIF 1.0, RIF1.0a, RIF 1.1; RIF1.1a and the ProSTEP iViP released the recommendation RIF 1.2.

As the acronym RIF has an ambiguous meaning within the OMG, the acronym ReqIF has been introduced to separate it from the W3C’s Rule Interchange Format. ReqIF 1.0 is the direct successor of the ProSTEP iViP recommendation RIF 1.2.
7 Concept Overview and Use Cases

7.1 Preface: How requirements authoring tools handle information

Most modern requirement authoring tools emulate word processors, but offer additional features. This allows authors of requirement specifications who have been using word processors to continue working in a similar manner, but enjoy the benefits of a tool specialized for authoring requirements.

Figure 7.1 shows an example for the transition from creating a textual document using a word processor to authoring a specification in a modern requirements authoring tool.

Figure 7.1 - Transition from word processors to requirement authoring tools
The word processing features of requirement authoring tools include the following.

<table>
<thead>
<tr>
<th>Feature</th>
<th>How word processors handle it</th>
<th>How requirement authoring tools handle it</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Structure specifications hierarchically</td>
<td>A user of a word processor structures documents by creating a hierarchy of clauses and sub clauses. The word-processor supports this task by automating the numbering of headlines and the creation of an outline.</td>
<td>Requirement authoring tools support the creation of hierarchically structured specifications. Users can create tree structures of requirements.</td>
</tr>
<tr>
<td>2. Use formatted text in the specifications</td>
<td>Word processors support, among other things, the bold, underlined, italic, and strikethrough text, bullet points and numbering in the documents.</td>
<td>Requirement authoring tools support the use of bold, underlined, italic, and strikethrough text, bullet points, and numbering in attribute values of requirements.</td>
</tr>
<tr>
<td>3. Reference binary files</td>
<td>Word processors support referencing binary files, for example spreadsheets, presentation slides, etc. from within a document.</td>
<td>Requirement authoring tools support the referencing of binary files from within attribute values of requirements.</td>
</tr>
</tbody>
</table>

The features that are specific to requirement authoring tools include the following.

<table>
<thead>
<tr>
<th>Feature</th>
<th>How requirement authoring tools handle it</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Uniquely identify requirements</td>
<td>Requirement authoring tools allow to distinguish individual requirements and to automatically create a unique identifier for each requirement.</td>
<td>A user of a requirement authoring tool defines the attributes “id,” “description,” “priority,” “status,” and “department” as mandatory for a specification. The “priority” attribute has an integer data type, the “status” and “department” attributes have an enumeration data type, and the “description” attribute has a string data type. Each requirement may have a different value for each of these attributes.</td>
</tr>
<tr>
<td>5. Associate attributes with the requirements</td>
<td>A user of a requirement authoring tool can define arbitrary attributes and attach them to requirements. Typically, a set of requirements shares the same attributes. However, these attributes may have different values for each requirement, and the values may have different underlying data types.</td>
<td>Example purposes of relations: a) to establish traceability b) to connect non-functional to functional requirements.</td>
</tr>
<tr>
<td>6. Establish relations between requirements</td>
<td>A user of a requirement authoring tool can define relations between requirements.</td>
<td></td>
</tr>
</tbody>
</table>
### 7. Group relations

Some requirement authoring tools allow the user to define new types of relations and to group relations by their type.

A requirement authoring tool may allow its users to define the new type “contradicts” for relations between two requirements that contradict each other, and then allow the users to create a group of “contradicts” relations. Such a group of relations – together with the requirements that are related by it – may support the users when reviewing and consolidating specifications.

### 8. Restrict user access to certain information

Requirement authoring tools offer the feature to restrict access to certain information.

During an exchange of specifications, the partner company that receives a Customer Requirement Specification is not allowed to edit the “priority” attribute of the requirements.

### 7.2 How the Requirements Interchange Format handles information from requirement authoring tools

The Requirements Interchange Format has been set up with the goal to exchange specifications between modern requirement authoring tools. Therefore, the requirements interchange format must be able to represent the information described in the previous clause. The following table shows how the features described in the previous clause are represented in the format. In the third column of the table, references to the abstract syntax of the format and the descriptions of the elements can be found.

<table>
<thead>
<tr>
<th>Feature</th>
<th>How the Requirements Interchange Format handles it</th>
<th>References to abstract syntax of the format and the description of the elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Structure specifications hierarchically</td>
<td>ReqIF provides the concept of a specification that contains a hierarchical structure of requirements.</td>
<td>See sub clause 10.3 for the basics of requirement specifications. See sub clause 10.4 for the abstract syntax of hierarchies. See sub clause 10.8.38 for the class description of a specification (class). See sub clause 10.8.40 for the class description of a requirement. See sub clause 10.8.37 for the class description of a hierarchical structure.</td>
</tr>
<tr>
<td>2. Use formatted text in the specifications</td>
<td>As representing formatted text is a potentially complex topic, ReqIF re-uses a subset of W3C’s XHTML modules to ease implementation of the format. XHTML content MAY be used in specific attribute values. See sub clause 10.6.3 on the datatype for formatted content. See sub clause 10.8.11, 10.8.29, and 10.8.20 for the class descriptions.</td>
<td></td>
</tr>
<tr>
<td>3. Reference binary files</td>
<td>To allow referencing binary files from within an attribute value that contains formatted text, XHTML is used as a mechanism as well. See sub clause 10.8.20 for the class description.</td>
<td></td>
</tr>
<tr>
<td>4. Uniquely identify requirements</td>
<td>ReqIF provides an identification mechanism for requirements.</td>
<td>See sub clause 10.2 for the abstract syntax. See sub clause 10.8.32 for the class descriptions.</td>
</tr>
</tbody>
</table>
7.3 How the Requirements Interchange Format copes with different tool capabilities

Modern requirement authoring tools vary concerning the features they support. There is no “unified language” for requirements that all requirement authoring tools support, and therefore, there is also no meta-model shared between requirement authoring tools.

The Requirements Interchange Format deals with the tools that are on the market nowadays. Some typical situations and resulting consequences are outlined in the following table.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Example</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement authoring tools use different terminology for the same concept.</td>
<td>What is called an “object” in one authoring tool may be called a “requirement” in another authoring tool</td>
<td>ReqIF includes only a limited collection of concepts, but provides an (informal) mapping to various requirement authoring tools on the market. See Annex A for a reference to an implementation guide that provides such a mapping.</td>
</tr>
</tbody>
</table>
7.4 Exchange Scenarios

The Requirements Interchange Format (ReqIF) described in this specification defines a non-proprietary, open exchange format. Instead of exchanging textual requirement specification documents, requirement specifications are exchanged by transferring XML documents that comply to the ReqIF format, making them processable by tools.

One of the basic ideas of ReqIF is to offer the opportunity to exchange information between different installations of the same requirements authoring tool with a standardized form and that the same format can be used to exchange information between different requirements authoring tools.

This clause explains two exchange scenarios:

- In the first exchange scenario (“one-way”), requirement specifications of one exchange partner are provided to a second exchange partner, for example to inform the second partner about the requested requirements.

- In the second exchange scenario (“roundtrip”), requirement specifications of one exchange partner are provided to a second exchange partner as well. After that, however, the second exchange partner makes modifications to the requirements, for example to comment them concerning the feasibility. The second exchange partner transmits the modified requirement specifications back to the first exchange partner.

The two exchange partners mentioned above may for example be two different companies or two departments within one company. In any case, there needs to be at least one installation of a requirements authoring tool per exchange partner, which is used to author the requirements. There also needs to be a user for each requirements authoring tool who exports, imports, or updates the requirement specifications in the requirements authoring tools.

Clause 7.4.1 describes the relevant roles in the scenarios. The sub clauses 7.4.2 and 7.4.3 outline the two exchange scenarios. The steps of the exchange scenarios that need further detailing are described in sub clause 7.5. Note that ReqIF tools MAY support additional scenarios. For example, exchanges with more than two partners MAY be supported, or there MAY be other purposes for using ReqIF than exchange, for example document generation.
7.4.1 Role descriptions

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Role description</th>
<th>Role type</th>
</tr>
</thead>
</table>
| RequirementsAuthoringToolUser | • Party that is responsible for starting the export of requirements specifications from a requirements authoring tool to ReqIF exchange XML documents, or  
                              | • Party that is responsible for starting the import of ReqIF exchange XML documents to a requirements authoring tool. (An update of the specification in the requirements authoring tool may become necessary.) | Person    |
| RequirementsAuthoringTool     | See clause “Terms and Definitions” for a definition.                                                                                                                                                                | System    |

7.4.2 First exchange scenario (“One-Way”)

Figure 7.2 shows a one-way exchange of requirement specifications between two requirements authoring tools.

![Diagram of one-way exchange of requirements](image)

Figure 7.2 - One-Way exchange of requirements between two requirements authoring tools using ReqIF
Scenario steps

<table>
<thead>
<tr>
<th>Step ID</th>
<th>Step description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0: Export Requirement Specifications()</td>
<td>A user of a requirements authoring tool (User1) starts the export of requirement specifications from a requirements authoring tool (Tool1). See Use Case “UC1: Export Requirement Specifications” for details on this step.</td>
</tr>
<tr>
<td>1.1: Export Requirement Specifications() : Exchange XML Documents</td>
<td>The requirement specifications chosen by User1 are exported into one or more ReqIF compliant XML documents. See Use Case “UC1: Export Requirement Specifications” for details on this step.</td>
</tr>
<tr>
<td>2.0: Transmit(Exchange XML Documents)</td>
<td>NOTE: This step is not in the scope of ReqIF. The exchange XML documents are transmitted by the sender (User1) to the receiver (User2) using traditional file transfer tools (e.g., email or ftp).</td>
</tr>
<tr>
<td>3.0: Import Requirement Specifications(Exchange XML Documents)</td>
<td>User2 imports the exchange XML documents into his requirements authoring tool (Tool2). For the case that, during this step, requirement specifications are newly created in Tool2, see “UC2: Import New Requirement Specifications” for details. For the case that, during this step, existing requirement specifications are updated in Tool2, see “UC3: Update Requirement Specifications” for details.</td>
</tr>
</tbody>
</table>

7.4.3 Second exchange scenario ("Roundtrip")

Figure 7.3 shows a roundtrip exchange of requirement specifications between two requirements authoring tools.

![Diagram of Roundtrip exchange of requirements between two requirements authoring tools using ReqIF]

Figure 7.3 - Roundtrip exchange of requirements between two requirements authoring tools using ReqIF
### Scenario steps

<table>
<thead>
<tr>
<th>Step ID</th>
<th>Step description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0: Export Requirement Specifications()</td>
<td>A user of a requirements authoring tool (User1) starts the export of requirement specifications from a requirements authoring tool (Tool1). See Use Case “UC1: Export Requirement Specifications” for details on this step.</td>
</tr>
<tr>
<td>1.1: Export Requirement Specifications(Exchange XML Documents)</td>
<td>The requirement specifications chosen by User1 are exported into one or more ReqIF compliant XML documents. See Use Case “UC1: Export Requirement Specifications” for details on this step.</td>
</tr>
<tr>
<td>2.0: Transmit(Exchange XML Documents)</td>
<td>NOTE: This step is not in the scope of ReqIF. The exchange XML documents are transmitted by the sender (User1) to the receiver (User2) using traditional file transfer tools (e.g., email or ftp).</td>
</tr>
<tr>
<td>3.0: Import Requirement Specifications (Exchange XML Documents)</td>
<td>User2 imports the exchange XML documents into his requirements authoring tool (Tool2). For the case that, during this step, requirement specifications are newly created in Tool2, see “UC2: Import New Requirement Specifications” for details. For the case that, during this step, existing requirement specifications are updated in Tool2, see “UC3: Update Requirement Specifications” for details.</td>
</tr>
<tr>
<td>4.0: Modify Requirement Specifications</td>
<td>User2 modifies the requirement specifications in Tool2. He MAY add or delete individual requirements or requirement specifications and change the contents of requirements or the structure of requirement specifications.</td>
</tr>
<tr>
<td>5.0: Export Requirement Specifications</td>
<td>User2 starts the export of the requirement specifications.</td>
</tr>
<tr>
<td>5.1: Export Requirement Specifications(Exchange XML Documents)</td>
<td>The requirement specifications chosen by User2 are exported into one or more ReqIF compliant XML documents.</td>
</tr>
<tr>
<td>6.0: Transmit(Exchange XML Documents)</td>
<td>NOTE: This step is not in the scope of ReqIF. The exchange XML documents are transmitted by the sender (User2) to the receiver (User1) using traditional file transfer tools (e.g., email or ftp).</td>
</tr>
<tr>
<td>7.0: Import Requirement Specifications (Exchange XML Documents)</td>
<td>User1 imports the exchange XML documents into Tool1. For the case that, during this step, requirement specifications are newly created in Tool1, see “UC2: Import New Requirement Specifications” for details. For the case that, during this step, existing requirement specifications are updated in Tool1, see “UC3: Update Requirement Specifications” for details.</td>
</tr>
</tbody>
</table>
7.5 Detailed Use Cases

7.5.1 Use Case Overview

Figure 7.4 - Use Cases for the Requirements Interchange Format

7.5.2 Use Case Specifications

The template used for specifying the succeeding uses cases document is loosely based on a Use Case template by Alistair Cockburn.
UC1: Export Requirement Specifications

<table>
<thead>
<tr>
<th>ID</th>
<th>UC-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Export Requirement Specifications</td>
</tr>
</tbody>
</table>

**CHARACTERISTIC INFORMATION**

**Goal in Context**
A user of a requirements authoring tool wants to export requirement specifications and relations between them from the requirements authoring tool to an exchange XML document.

**Preconditions**
The user has a requirements authoring tool installed. The user has a ReqIF tool installed that is capable of exporting requirement specifications from this requirements authoring tool. The requirement specifications the user wants to export are available in the requirements authoring tool and their contents are accessible by the user.

**Success End Condition**
The requirement specifications the user wanted to be exported have successfully been exported from the requirements authoring tool to an exchange XML document.

**Failed End Condition**
The requirement specifications the user wanted to be exported have not successfully been exported from the requirements authoring tool to an exchange XML document.

**Primary Actor**
The user of a requirements authoring tool.

**MAIN SUCCESS SCENARIO**

**Step 1**
The user uses the ReqIF tool to specify the requirements specifications he wants to export and to request the export of the requirements specifications.

**Step 2**
The ReqIF tool exports each specification to one or several exchange XML documents. The exported exchange XML documents include information about requirements, types, attributes, and (optionally) access policies relations between requirements; the relations may be grouped. The structure of the specifications.

**ALTERNATIVE SCENARIOS**

**Alternative B: Export Parts of a Specification (Step 1 + Step 2)**
Instead of exporting complete requirement specifications, a ReqIF tool MAY additionally have the feature to export only parts of a specification.

UC2: Import New Requirement Specifications

<table>
<thead>
<tr>
<th>ID</th>
<th>UC-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Import New Requirement Specifications</td>
</tr>
</tbody>
</table>

**CHARACTERISTIC INFORMATION**

**Goal in Context**
A user of a requirements authoring tool wants to import requirement specifications and relations between them contained in exchange XML documents into a requirements authoring tool.

**Preconditions**
The user has a requirements authoring tool installed. The user has a ReqIF tool installed that is capable of importing requirement specifications from an exchange XML document into this requirements authoring tool. The exchange XML documents to be imported are available to the user. The user has the appropriate access rights in the requirements authoring tool to create new specifications, their contents and relations between requirements. The exchange XML documents have not been imported to the above requirements authoring tool so far.

**Success End Condition**
The requirement specifications the user wanted to be imported have successfully been imported from the exchange XML documents to the requirements authoring tool.
Failed End Condition | The requirement specifications the user wanted to be imported have not successfully been imported from the exchange XML documents to the requirements authoring tool.
--- | ---
Primary Actor | The user of a requirements authoring tool

**MAIN SUCCESS SCENARIO**

**Step 1** | The user specifies the following information using the ReqIF tool:
- the exchange XML documents he wants to import
- the target location of elements to be created in the requirements authoring tool
After that, the user requests the import of the exchange XML documents using the ReqIF tool.

**Step 2** | The ReqIF tool imports the exchange XML document into the requirements authoring tool.

**ALTERNATIVE SCENARIOS**

---

**UC3: Update Requirement Specifications**

<table>
<thead>
<tr>
<th>ID</th>
<th>UC-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Update Requirement Specifications</td>
</tr>
</tbody>
</table>

**CHARACTERISTIC INFORMATION**

**Goal in Context** | A user of a requirements authoring tool wants to import requirement specifications and relations between them contained in exchange XML documents into a requirements authoring tool.

**Preconditions** | The user has a requirements authoring tool installed. The user has a ReqIF tool installed that is capable of importing requirement specifications from an exchange XML document into this requirements authoring tool. The exchange XML documents are available to the user. The user has the appropriate access rights in the requirements authoring tool to update specifications, their contents and relations between requirements. The exchange XML documents have previously been imported to the above requirements authoring tool.

**Success End Condition** | The requirement specifications in the requirements authoring tool that correspond to the specifications contained in the exchange XML document have successfully been updated.

**Failed End Condition** | The requirement specifications in the requirements authoring tool that correspond to the specifications contained in the exchange XML document have successfully been updated.

**Primary Actor** | The user of a requirements authoring tool

**MAIN SUCCESS SCENARIO**

**Step 1** | The user specifies the following information using the ReqIF tool:
- The exchange XML documents he wants to use as a source for the update.
- The specifications in the requirements authoring tool he wants to update.
After that, the user requests the update.

**Step 2** | The ReqIF tool merges the existing requirement specifications in the requirements authoring tool with the information from the exchange XML documents.

**ALTERNATIVE SCENARIOS**

---
8 Abstract Architecture

Figure 8.1 shows the requirements exchange process, with a particular emphasis on attribute handling.

Figure 8.1 - The requirements exchange process

The four activities in the upper left corner, following the start symbol, representing the creation and initial population of the exchange document through export from a requirements authoring tool capable enough to hold the complete requirements specification and all its attributes in original form.
After the initial population of the exchange document, the exchange process becomes a chain of requirements exchanges. This could be a linear chain of partner organizations, or a “roundtrip” exchange terminating at the originating organization, the process is always the same.

The following table describes the activities inside the attribute import loop. One loop iteration is performed for each attribute value encountered in the import stream.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Activity Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has isSimplified = true</td>
<td>If isSimplified is true for a formatted attribute value, it signals that the previous tool in the exchange chain was unable to handle the attribute value in its original (formatted) form and created a simplified representation instead.</td>
</tr>
<tr>
<td>Formatted Attribute?</td>
<td>Interpretation deficiencies are only expected and tolerated for formatted attribute values (AttributeValueXHTML elements), therefore formatted attributes are singled out. Other attributes bypass all the following steps.</td>
</tr>
<tr>
<td>Understood Formatted Attribute Value</td>
<td>An attempt is made to interpret the current formatted attribute value. If the tool is unable to handle the formatted attribute value in its original form for any reason, special processing, as described in the following three rows, is required. If the formatted attribute value is understood, the following three rows are skipped.</td>
</tr>
<tr>
<td>Remember Original Attribute Value</td>
<td>The formatted attribute value contained in the exchange document needs to be preserved, as information may get lost during the import of the attribute value. This is done by copying the Value to the Original Value.</td>
</tr>
<tr>
<td>Create Simplified Attribute Representation</td>
<td>Dependent on the formatted attribute value and on the experienced capability mismatch, the importing tool must create a suitable simplified attribute representation that can be presented to the tool user.</td>
</tr>
</tbody>
</table>
After the importing requirements authoring tool finished its processing on the imported requirements specification, the exchange chain may end by simple termination without further action on the exchange document, or the requirements specification in its processed form may be re-exported. The following table describes this export process.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Action Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remember Attribute Value as Simplified</strong></td>
<td>If simplified representations of attribute values are imported or newly created, they need to be marked so that they get exported with isSimplified set to true. This signals to the following importer in the exchange chain that the tool was unable to interpret the formatted attribute value in its original form.</td>
</tr>
<tr>
<td><strong>Notify User or Tool [send event]</strong></td>
<td>If the previous tool in the exchange chain was unable to handle the attribute value in its original (formatted) form, this event alerts the user or tool about the potential information loss that previously occurred.</td>
</tr>
<tr>
<td><strong>Import Simplified Attribute Value ?</strong></td>
<td>A decision is made whether attribute values are imported in their simplified form. INFORMATIVE NOTE: This decision may for example be based on a configuration setting. Importing the simplified attribute representations is especially interesting for point-to-point exchanges between two human users of requirement authoring tools, where the sender needs to know what has actually been received. For automatic exchanges between tools, keeping the original attribute value may be the better option.</td>
</tr>
<tr>
<td><strong>Restore Original Attribute Value</strong></td>
<td>This restores the original attribute value from step Remember Original Attribute Value. This is done by copying theOriginalValue to theValue.</td>
</tr>
<tr>
<td><strong>Remember Attribute Value as Not Simplified</strong></td>
<td>This is the alternate path for all attributes other than formatted attributes, and for formatted attributes understood in their original form.</td>
</tr>
<tr>
<td><strong>Provide Attribute to Importing Tool</strong></td>
<td>This is the final step in the attribute interpretation loop.</td>
</tr>
</tbody>
</table>
9 Exchange Document Structure

9.1 General

This clause defines the top-level structure of a ReqIF Exchange Document, consisting of a header, the core content, and optionally of one or more tool-specific content extensions. These document elements are enclosed by the ReqIF root element. See Clause 1 for detailed definition of the content elements.

![Diagram of ReqIF Document Structure]

Figure 9.1 - ReqIF Document Structure

9.2 Class Descriptions

9.2.1 ReqIF

Package: ReqIF
isAbstract: No
Generalization: none

Description

This class constitutes the root element of the Exchange Document.

Attributes

- lang: xml:lang [0..1]
  Default language encoding of the Exchange XML Document content. The format is defined by the standard for specifying languages in XML documents proposed by the W3C. See [http://www.w3.org/TR/xml11/#sec-lang-tag](http://www.w3.org/TR/xml11/#sec-lang-tag)
Associations

- coreContent : ReqIFContent [1] {composite}
  This composition links the mandatory Exchange Document content.
- theHeader : ReqIFHeader [1] {composite}
  This composition links the mandatory Exchange Document header, which contains metadata relevant for this exchange.
- toolExtensions : ReqIFToolExtension [0..*] {composite}
  This composition links optional Exchange Document content based on tool extensions, if such extensions and content are present.

Operations

No operations.

Constraints


Tags

org.omg.reqif.global_element True
org.omg.reqif.ordered True

Semantics

Element ReqIF is the document root element, which encapsulates the whole Exchange Document.

Additional Information

No additional information.

9.2.2 ReqIFContent

Description

This class represents the mandatory content of the Exchange Document. Please refer to sub clause 10.8.35 for the complete class description.

9.2.3 ReqIFHeader

Package: ReqIF
isAbstract: No
Generalization: none

Description

This class holds metadata relevant to the Exchange Document content.
Attributes

- **comment**: string [0..1]
  
  Optional comment associated with the Exchange Document as a whole.

- **creationTime** : xsd:dateTime
  
  Time of creation of the exchange XML document in the format of the XML Schema data type “dateTime” which specifies the time format as *CCYY-MM-DDThh:mm:ss* with optional time zone indicator as a suffix ±hh:mm. Example: 2005-03-04T10:24:18+01:00 (MET time zone).

- **identifier** : xsd::ID
  
  Unique identifier for whole exchange XML document. The value of the identifier is of the XML Schema data type “xsd::ID”

- **repositoryId** : string [0..1]
  
  Optional unique identifier of the repository containing the requirements that have been exported. Examples for repositoryId: databaseId, URL.

- **reqIFToolId** : string
  
  Identifier of the exporting ReqIF tool.

- **reqIFVersion** : string
  
  ReqIF interchange format and protocol version.

- **sourceToolId** : string
  
  Identifier of the exporting requirements management tool.

- **title** : string
  
  Title of the Exchange Document.

Associations

- **documentRoot** : ReqIF [1]
  
  Linking back to the Exchange Document root element.

Operations

No operations

Constraints

[1] The value of attribute reqIFVersion must be “1.0.”

Tags

```
org.omg.reqif.order               1
org.omg.reqif.xsd_element         “comment,” “creationTime,” “repositoryId,” “reqIFToolId,” “reqIFVersion,” “sourceToolId,” “title”
org.omg.reqif.fixed               “reqIFVersion”
```

Semantics

Metainformation held in the ReqIFHeader element is applicable to the Exchange Document as a whole.
Additional Information

No additional information.

9.2.4 ReqIFToolExtension

Package: ReqIF
isAbstract: No
Generalization: none

Description

This class allows the optional inclusion of tool-specific information into the Exchange Document.

Attributes

No attributes.

Associations

- documentRoot : ReqIF [1]
  Linking back to the Exchange Document root element.

Operations

No operations.

Constraints

No constraints.

Tags

org.omg.reqif.order 3
org.omg.reqif.processContents lax

Semantics

ReqIFToolExtension elements may be used to exchange requirements authoring tool specific concepts for which no ReqIF information types are applicable.

As an example, a ReqIFToolExtension element can be used to represent instances of the View type found in requirements authoring tools, as there is no ReqIF information type defined for the concept of a View.

Additional Information

As format, type, and content of information transferred in ReqIFToolExtension is not specified, preservation and/or correct interpretation of this information cannot be guaranteed if:

- different ReqIF tools are used for export and import, or
- different requirements authoring tools are used as source and target for the exchange.
10 Exchange Document Content

10.1 Overview

Figure 10.1 provides an overview of the information types aggregated by a ReqIFContent element.

10.2 Identification of Elements

Information elements in an Exchange Document are distinguished through global unique identifiers (Identifiable elements), which are assigned during the creation of the information element. After assignment, these identifiers must not be altered during the lifetime of the information element, nor reused for any different information element. These identifiers allow the unique identification of information elements, even across several exchange documents.

Using these identifiers, elements of the specification that have been modified in a requirements authoring tool of an exchange partner can be updated in the requirements authoring tool where they had originally been created. In cases where a tool is unable to handle the original element identifiers, the original identifier may be complemented with a tool-specific alternative identifier (AlternativeID element).
10.3 Specifications, Requirements, and Attributes

A key concept of ReqIF is the specification (Specification element), which acts as a container for the individual requirements (SpecObject elements). SpecObject elements constitute individually identifiable requirements. Apart from the information inherited from Identifiable, an instance of SpecObject is “empty” by itself and therefore contains no data.

Requirements can have attributes to represent requirement related information kept in the requirement authoring tool. Typically, a set of requirements shares the same attributes. For example: all requirements in a certain set have a “priority”-attribute and a “status”-attribute. What is actually shared among the requirements is the requirement attribute definitions (the number of attributes, the names of the attributes, the default values for the attributes, and the datatypes of the attributes.) In contrast to that, the value of a certain attribute may vary among the requirements in the set.

Therefore, ReqIF differs between the attribute definitions (AttributeDefinition elements) and the attribute values (AttributeValue elements) of a requirement. Several attribute definitions can be attached to a requirement by using a type (SpecType element).

In ReqIF, the concept of having attributes also expands to relations between requirements (SpecRelation elements), to requirement specifications (Specification elements), and to groups of relations (RelationGroup elements).

Figure 10.3 - Specification (Specificification), requirement (SpecObject), requirement relation (SpecRelation), relation group (RelationGroup) and associated attributes (AttributeDefinition, AttributeValue)

The information type AttributeDefinition is an abstract super-class for attribute definitions.
The information type AttributeValue is an abstract superclass for attribute values. There is one concrete AttributeValue information type for each concrete (direct or indirect) subclass of AttributeDefinition.
10.4 Hierarchical Structuring of Requirements in a Specification and Requirement Relations

Two requirements may have a relation to each other, for example to establish traceability between a Customer Requirements Specification and a System Requirements Specification. Having a relation is represented by an association of one SpecRelation element to two SpecObject elements, one being the source, one the target of the relation.

The two specifications that are related to each other (in the above example: a Customer Requirements Specification and a System Requirements Specification) are referred to by the sourceSpecification and targetSpecification association of a RelationGroup instance.

The hierarchical structure of a requirement specification is represented by SpecHierarchy elements.
Figure 10.6 - Requirements, requirement relations and how requirements are structured hierarchically in a specification

Figure 10.7 - Grouping relations by the source and target specification they relate
10.5 Representing Attribute Data Types

10.5.1 Representing Data Types

In ReqIF, there are three kinds of data types:

1. Simple data types (i.e., Integer, Date, Real, Boolean, String)
2. A data type for enumeration values.
3. A data type for formatted content. This data type can also be used to reference external objects (for example, pictures from within formatted content).

The abstract super-class for the three kinds of data types is DatatypeDefinition. The classes of data types are displayed in Figure 10.8. Concrete information types for simple data type definitions inherit from the information type DatatypeDefinitionSimple.

![DatatypeDefinition class hierarchy](image)

Figure 10.8 - DatatypeDefinition class hierarchy

10.5.2 Relating Attributes to Data Types

Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. For example: the value of a “priority”-attribute may need to be an integer number, while the value for a “status”-attribute may need to be picked from a list of choices.

In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

A concrete AttributeDefinition element MAY contain a default value that represents the value that is used if no attribute value is supplied by the user of the requirements authoring tool. For example, a user of a requirements authoring tool may specify that the value “TBD” is used for the “status”-attribute of all requirements that have not been assigned a “status” so far.
10.6 Concrete Data Types

10.6.1 Simple Data Types

The following diagram shows the primitive data types that are supported by ReqIF.
10.6.2 Enumeration Data Type

The following diagram shows the enumeration data type that is supported by ReqIF.

Figure 10.10 - Simple data types

Figure 10.11 - Enumeration data types
10.6.3 Data Type for XHTML Content

There are two main functionalities of ReqIF that are realized through XHTML:

1. Storing of formatted text.
   Requirement authoring tools support (among other things) the use of bold, italic, underlined, and strikethrough text, bullet points, and numbering in attribute values of requirements. Reusing XHTML is a pragmatic approach to represent this formatted text in exchange documents.

2. Inclusion of objects that are external to the exchange XML document in the requirements authoring tool. The objects may have binary content.

Furthermore, as requirements authoring tools and ReqIF tools have different capabilities, information may be lost during the exchange process (3.)

Please note that instances of AttributeValueXHTML are in principle wrappers for an XHTML document that is embedded into the exchange XML document.

The embedded XHTML document is modeled as a ReqIF information element XhtmlContent, as shown in Figure 10.12. XhtmlContent switches the XML namespace to the standard XHTML namespace http://www.w3.org/1999/xhtml

Separating the XML namespaces allows validating against different XML-Schemas (i.e., against the ReqIF-Schema and against an XHTML-Schema) resulting in more independency between the different XML-Schemas.

![Figure 10.12 - Data types for XHTML content](image)

10.7 Access Restrictions

For certain information elements, ReqIF allows to specify whether they are editable or read-only by the user of the requirements authoring tool. Having such access restrictions in place supports exchange processes where partners have different rights to modify information.

There are three cases for which access may be restricted:

1. Making subtrees of a specification hierarchy editable or read-only.
   A subtree of a specification hierarchy that is editable allows the user to add or remove requirements from/to the subtree. Subtrees of the subtree may override the access settings.
2. Making requirement attributes editable or read-only in subtrees of a specification hierarchy.

A set of attributes is editable in a subtree means: the values of the attributes in the set can be edited in that subtree. As a consequence, all attributes that are not in the set are not editable in that subtree. Subtrees of the subtree may override the access settings.

3. Making the attribute definition of an attribute editable or read-only

For example: it shall be possible to make the “status”-attribute definition read-only, meaning that no additional enumeration literal can be added to the “status” attribute’s set of enumeration literals (like “accepted,” “rejected,” etc.)

For the classes affected by the access restriction concept, see the class descriptions for details.

---

**Figure 10.13 - Access Restrictions**

**10.8 Class Descriptions**

**10.8.1 AccessControlledElement**

**Package:** ReqIF  
**isAbstract:** Yes  
**Generalization:** Identifiable  

**Description**

Base class for classes that may restrict user access to their information.
Attributes

- isEditable: Boolean[0..1]
  True means that the element's contents may be modified by the user of a tool containing the element.
  False or leaving isEditable out means that the element is read-only to the user of a tool containing the element.

Associations

Operations

No operations

Constraints

No constraints

Tags

No tags

Semantics

For certain information elements, ReqIF allows to specify whether they are editable or read-only by the user of the tool containing them. Having such access restrictions in place supports exchange processes where partners have different rights to modify information.

Sub classes of AccessControlledElement may detail the semantics of “being editable” in their context.

Additional Information

No additional information

10.8.2 AlternativeID

Package: ReqIF
isAbstract: No
Generalization: none

Description

Used to provide an alternative, tool-specific identification.

Attributes

- identifier: string[1]
  An optional alternative identifier, which may be a requirements management tool identifier or ReqIF tool identifier.

Associations

- ident : Identifiable
  Back linkage to the owning Identifiable.

Operations

No operations
Constraints


Tags

No tags

Semantics

In cases where Identifiable::identifier cannot be handled by a requirements authoring tool or ReqIF tool for any reason, an AlternativeID may be associated to provide a tool consumable alternative identification.

Additional Information

No additional information

10.8.3 AttributeDefinition

Package: ReqIF
isAbstract: Yes
Generalization: AccessControlledElement

Description

Base class for attribute definitions.

Attributes

No attributes

Associations

• specType : SpecType [1]
  Back linkage to the owning SpecType.

Operations

No operations

Constraints

[1] The attribute longName inherited from Identifiable is mandatory for all sub classes of AttributeDefinition.

Tags

No tags

Semantics

Base class for Exchange Document content attributes, must be specialized for concrete attributes.

Additional Information

No additional information.
10.8.4 AttributeDefinitionBoolean

Package: ReqIF
isAbstract: No
Generalization: AttributeDefinitionSimple

Description
Definition of a boolean attribute.

Attributes
No attributes

Associations
- defaultValue: AttributeValueBoolean [0..1] {composite}
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.
- type: DatatypeDefinitionBoolean[1]
  Reference to the data type

Operations
No new operations

Constraints
[1] If the inherited isEditable attribute is set to false or left out, no modification of the default value by tool users is allowed.

Tags
No tags

Semantics
Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

An AttributeDefinitionBoolean element therefore relates an AttributeValueBoolean element to a DatatypeDefinitionBoolean element via its type attribute.

An AttributeDefinitionBoolean element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.
10.8.5 AttributeDefinitionDate

Package: ReqIF
isAbstract: No
Generalization: AttributeDefinitionSimple

Description
Definition of a date and time attribute.

Attributes
No attributes

Associations
- defaultValue : AttributeValueDate [0..1] {composite}
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.
- type : DatatypeDefinitionDate[1]
  Reference to the data type.

Operations
No new operations

Constraints
[1] If the inherited isEditable attribute is set to false or left out, no modification of the default value by tool users is allowed.

Tags
No tags

Semantics
Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

An AttributeDefinitionDate element therefore relates an AttributeValueDate element to a DatatypeDefinitionDate element via its type attribute.

An AttributeDefinitionDate element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.
10.8.6 AttributeDefinitionEnumeration

**Package:** ReqIF

**isAbstract:** No

**Generalization:** AttributeDefinition

**Description**

Definition of an enumeration attribute.

**Attributes**

multiValued : Boolean

- If set to true, this means that the user of a requirements authoring tool can pick one or more than one of the values in the set of specified values as an enumeration attribute value.
- If set to false, this means that the user of a requirements authoring tool can pick exactly one of the values in the set of specified values as an enumeration attribute value.

**Associations**

- defaultValue : AttributeValueEnumeration [0..1] {composite}
  
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.

- type : DatatypeDefinitionEnumeration [1]
  
  Reference to the data type for enumerations.

**Operations**

No operations

**Constraints**

[1] If the inherited isEditable attribute is set to false or left out, all of the following constraints apply:

- No modification of the default value by tool users is allowed.
- No adding, deleting, or modification of enumeration literals by tool users is allowed.

**Tags**

No tags

**Semantics**

Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

An AttributeDefinitionEnumeration element therefore relates an AttributeValueEnumeration element to a DatatypeDefinitionEnumeration element via its type attribute.

An AttributeDefinitionEnumeration element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.
There are basically two kinds of enumerations: “single-choice” and “multiple-choice” enumerations. “Single-choice” enumerations allow the user of a requirements authoring tool to pick exactly one value out of a set of specified values. “Multiple-choice” enumerations allow the user of a requirements authoring tool to pick several values out of a set of specified values. For “multiple-choice” enumerations, the multiValued attribute needs to be set to true, for “single-choice” enumerations it needs to be set to false.

The set of specified values is defined by the `DatatypeDefinitionEnumeration` element that is linked via the type association.

**Additional Information**

No additional information.

### 10.8.7 AttributeDefinitionInteger

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** AttributeDefinitionSimple

**Description**

Definition of an integer attribute.

**Attributes**

No attributes

**Associations**

- `defaultValue : AttributeValueInteger [0..1] {composite}`  
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.

- `type : DatatypeDefinitionInteger[1]`  
  Reference to the data type

**Operations**

No new operations

**Constraints**

[1] If the inherited isEditable attribute is set to `false` or left out, no modification of the default value by tool users is allowed.

**Tags**

No tags
Semantics

Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

An AttributeDefinitionInteger element therefore relates an AttributeValueInteger element to a DatatypeDefinitionInteger element via its type attribute.

An AttributeDefinitionInteger element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.

10.8.8 AttributeDefinitionReal

Package: ReqIF
isAbstract: No
Generalization: AttributeDefinitionSimple

Description
Definition of an attribute with Real data type.

Attributes
No attributes

Associations
• defaultValue : AttributeValueReal [0..1] {composite}
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.

• type : DatatypeDefinitionReal[1]
  Reference to the data type.

Operations
No new operations

Constraints
[1] If the inherited isEditable attribute is set to false or left out, no modification of the default value by tool users is allowed.

Tags
No tags

Semantics
Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).
An `AttributeDefinitionReal` element therefore relates an `AttributeValueReal` element to a `DatatypeDefinitionReal` element via its `type` attribute.

An `AttributeDefinitionReal` element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.

### 10.8.9 AttributeDefinitionSimple

**Package:** ReqIF  
**isAbstract:** Yes  
**Generalization:** AttributeDefinition

**Description**  
Abstract base class of simple type attributes.

**Attributes**  
No attributes

**Associations**  
No associations

**Operations**  
No new operations

**Constraints**  
No constraints

**Tags**  
No tags

**Semantics**  
Abstract base class of simple type attributes.

**Additional Information**  
No additional information

### 10.8.10 AttributeDefinitionString

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** AttributeDefinitionSimple
Description
Definition of an attribute with string data type.

Attributes
No attributes

Associations
- `defaultValue : AttributeValueString [0..1] {composite}`
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.
- `type : DatatypeDefinitionString[1]`
  Reference to the data type.

Operations
No new operations

Constraints
[1] If the inherited `isEditable` attribute is set to `false` or left out, no modification of the default value by tool users is allowed.

Tags
No tags

Semantics
Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (`AttributeValue` element) is related to its data type (`DatatypeDefinition` element) via an attribute definition (`AttributeDefinition` element).

An `AttributeDefinitionString` element therefore relates an `AttributeValueString` element to a `DatatypeDefinitionString` element via its `type` attribute.

An `AttributeDefinitionString` element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.

10.8.11 AttributeDefinitionXHTML

Package: ReqIF
isAbstract: No
Generalization: AttributeDefinition

Description
Definition of an XHTML attribute.
Attributes
No attributes

Associations
- defaultValue : AttributeValueXHTML [0..1] {composite}
  Linkage of the owned default value that is used if no attribute value is supplied by the user of the requirements authoring tool.
- type : DatatypeDefinitionXHTML [1]
  Reference to the data type.

Operations
No operations

Constraints
[1] If the inherited isEditable attribute is set to false or left out, no modification of the default value by tool users is allowed.

Tags
No tags

Semantics
Each concrete attribute value that is created in a requirements authoring tool needs to be valid against its related data type. In ReqIF, each attribute value (AttributeValue element) is related to its data type (DatatypeDefinition element) via an attribute definition (AttributeDefinition element).

An AttributeDefinitionXHTML element therefore relates an AttributeValueXHTML element to a DatatypeDefinitionXHTML element via its type attribute.

An AttributeDefinitionXHTML element MAY contain a default value that represents the value that is used as an attribute value if no attribute value is supplied by the user of the requirements authoring tool.

Additional Information
No additional information.

10.8.12 AttributeValue

Package: ReqIF
isAbstract: Yes
Generalization: None

Description
Base class for concrete attribute values.
Attributes
No attributes

Associations
• specElAt : SpecElementWithAttributes [1]
  The linkage between AttributeValue and the owning class SpecElementWithAttributes

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
This is the base class for all concrete classes that represent attribute values of requirements authoring tools.

Additional Information
No additional information

10.8.13 AttributeValueBoolean

Package: ReqIF
isAbstract: No
Generalization: AttributeValueSimple

Description
A boolean attribute value.

Attributes
• theValue : Boolean
  The attribute value.

Associations
• definition : AttributeDefinitionBoolean [1]
  Reference to the value definition.
• owningDefinition : AttributeDefinitionBoolean [1]
  Back linkage of the owning attribute definition.
**Operations**
No new operations

**Constraints**
No constraints

**Tags**
No tags

**Semantics**
Contains a boolean attribute value.

**Additional Information**
No additional information

### 10.8.14 AttributeValueDate

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** AttributeValueSimple

**Description**
A date/time attribute value.

**Attributes**
- **theValue : xsd::dateTime**  
  The attribute value.

**Associations**
- **definition : AttributeDefinitionDate [1]**  
  Reference to the value definition.
- **owningDefinition : AttributeDefinitionDate [1]**  
  Back linkage of the owning attribute definition.

**Operations**
No new operations

**Constraints**
No constraints

**Tags**
No tags
**Semantics**
Contains a date/time attribute value.

**Additional Information**
No additional information

### 10.8.15 AttributeValueEnumeration

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** AttributeValue

**Description**
Definition of an enumeration attribute value.

**Attributes**

**Associations**

- definition : AttributeDefinitionEnumeration [1]  
  Reference to the attribute definition that relates the value to its data type.

- owningDefinition : AttributeDefinitionEnumeration [1]  
  Back linkage of the owning attribute definition.

- values : EnumValue [*]  
  Reference to the enumeration values that are chosen from a set of specified values.

**Operations**
No operations

**Constraints**

[1] If the multiValued attribute of the AttributeValueEnumeration element referenced by the definition association is set to false, the values set must contain at most one value.

[2] Each value referenced by the values association must be contained in the specifiedValues set of the related DatatypeDefinitionEnumeration element.

NOTE: The definition association references an AttributeDefinitionEnumeration element that in turn references the DatatypeDefinitionEnumeration element mentioned above.

**Tags**
No tags

**Semantics**
Provides a link to the concrete literals of an enumeration.
Additional Information

No additional information

10.8.16 AttributeValueInteger

Package: ReqIF

isAbstract: No

Generalization: AttributeValueSimple

Description

An integer attribute value.

Attributes

• theValue : integer
  The attribute value.

Associations

• definition : AttributeDefinitionInteger [1]
  Reference to the value definition.

• owningDefinition : AttributeDefinitionInteger [1]
  Back linkage of the owning attribute definition.

Operations

No new operations

Constraints

No constraints

Tags

No tags

Semantics

Contains an integer attribute value.

Additional Information

No additional information

10.8.17 AttributeValueReal

Package: ReqIF

isAbstract: No

Generalization: AttributeValueSimple
**Description**
A Real attribute value.

**Attributes**
- **theValue**: float
  The attribute value.

**Associations**
- **definition**: AttributeDefinitionReal [1]
  Reference to the value definition.
- **owningDefinition**: AttributeDefinitionReal [1]
  Back linkage of the owning attribute definition.

**Operations**
No new operations

**Constraints**
No constraints

**Tags**
No tags

**Semantics**
Contains a Real attribute value.

**Additional Information**
No additional information

### 10.8.18 AttributeValueSimple

**Package**: ReqIF
**isAbstract**: Yes
**Generalization**: AttributeValue

**Description**
Abstract base class for simple attribute values.
Attributes
No attributes

Associations
No associations

Operations
No new operations

Constraints
No constraints

Tags
No tags

Semantics
Abstract base class for simple attribute values.

Additional Information
No additional information

10.8.19 AttributeValueString

Package: ReqIF
isAbstract: No
Generalization: AttributeValueSimple

Description
A string attribute value.

Attributes
• theValue : string
  The attribute value.

Associations
• definition : AttributeDefinitionString [1]
  Reference to the value definition.
• owningDefinition : AttributeDefinitionString [1]
  Back linkage of the owning attribute definition.

Operations
No new operations
**Constraints**
No constraints

**Tags**
No tags

**Semantics**
Contains an string attribute value.

**Additional Information**
No additional information

### 10.8.20 AttributeValueXHTML

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** AttributeValue

**Description**
An attribute value with XHTML contents.

**Attributes**
- **isSimplified** : Boolean[0..1]  
  Set to true if the attribute value is a simplified representation of the original value.

**Associations**
- **definition** : AttributeDefinitionXHTML [1]  
  Reference to the value definition.
- **owningDefinition** : AttributeDefinitionXHTML [1]  
  Back linkage of the owning attribute definition.
- **theValue** : XhtmlContent [1] {composite}  
  Linkage to the owned XhtmlContent.
- **theOriginalValue** : XhtmlContent [0..1] {composite}  
  Linkage to the original attribute value that has been saved if isSimplified is true.

**Operations**
No operations

**Constraints**
[1] The value of isSimplified is considered false if it is left out.
Semantics

There are two main functionalities of ReqIF that are realized through XHTML:

1. Storing of formatted text.
   Requirement authoring tools support – among other things – the use of bold, italic, underlined, and strikethrough text, bullet points, and numbering in attribute values of requirements. Reusing XHTML is a pragmatic approach to represent this formatted text in exchange documents.

2. Inclusion of objects that are external to the exchange XML document in the requirements authoring tool. The objects may have binary content. Furthermore, as requirements authoring tools and ReqIF tools have different capabilities, information may be lost during the exchange process (3.)

1. Storing of formatted text

ReqIF reuses XML elements for formatting that are defined by XHTML 1.0. These XML elements, which are in the XHTML namespace are embedded into the exchange XML document by using an XHTML schema driver, as defined by the XHTML Modularization 1.1.

NOTE: Formatted content from a requirements authoring tool’s attribute values MUST always be stored as XHTML attribute values in the exchange XML documents. It is, for example, not allowed to store formatted content as RTF (Rich Text Format) or another format for formatted text, as this would decrease the interoperability between different ReqIF tools.

The XML elements of the following XHTML modules SHOULD be expected as contents of `AttributeValueXHTML` instances during an import of an exchange XML document:

1. `Text` Module
2. `List` Module
3. `Hypertext` Module
4. `Edit` Module
5. `Presentation` Module
6. `Basic Tables` Module
7. `Object` Module
8. `Style Attribute` Module

The contents of these modules are defined in the XHTML Modularization (http://www.w3.org/TR/xhtml-modularization/).

Concerning the XML attributes of the above XHTML elements, there are the following constraints:

- The `class` attribute of the XHTML Core Attribute Collection MUST NOT be used.
- Only the following values for the `style` attribute from the `Style Attribute` Module need to be considered during import:
Apart from these constraints, all XML attributes of the XHTML XML elements SHOULD be processed during import. If any of XHTML’s XML elements or XML attributes can’t be processed, information may be lost. See line “3. Handling information loss” on how to handle information loss.

2. Inclusion of objects that are external to the exchange XML document in the requirements authoring tool

External objects are referenced binary objects that are usually not edited with the requirements authoring tool itself, but by accessing an external application (e.g., a Visio drawing or an Excel sheet). External objects can be referenced from within a formatted text (as described in line “1. Storing of formatted text”).

External objects are referenced using the XHTML object element from the XHTML Object Module. The specification for the XTHML object element defines several XML attributes. For ReqIF, only a subset of these attributes is relevant and used. These attributes are shown together with their purposes in the following table.

<table>
<thead>
<tr>
<th>XHTML XML-Element</th>
<th>XML Attributes</th>
<th>Attribute types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>data</td>
<td>URI</td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>MIME-Type</td>
</tr>
<tr>
<td></td>
<td>width</td>
<td>Length</td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>Length</td>
</tr>
</tbody>
</table>

To maximize interoperability between ReqIF tools, the following rules MUST be obeyed:

- If there is a specific MIME-type for the application that handles the external object, it MUST be stored in the type attribute and no attribute in addition to the four attributes for the object element (data, type, width, height) MUST be used in that case.

- For XHTML object elements that refer to an external object that is not an image with the MIME-Type image/png, an alternative image AND an alternative text MUST be provided analogous to the following example.

```
<object data="http://www.example.com/bar.mp3" type="audio/mpeg">
  <!-- Else, try the image -->
  <object data="baz.png" type="image/png">
    <!-- Else process the alternate text -->
    The <strong>Earth</strong> as seen from space.
  </object>
</object>
```

An exporting ReqIF tool MUST only export alternative images with MIME-Type image/png.

- The location of an external object MUST be specified via the data attribute. The data attribute MUST either contain:
  a) a URL relative to the location of the exchange XML document, or
b) an absolute URL.

Case a) MUST be supported, case b) SHOULD be supported.

3. Handling information loss

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.

If AttributeValueXHTML elements are marked that way, importing ReqIF tools SHOULD still display a simplified version of the attribute value using an external HTML processor, allowing the user to at least read the information. Tool vendors are strongly encouraged to implement this feature.

The following rules MUST be obeyed during the import of each AttributeValueXHTML element:
If either

- the requirements authoring tool is not capable of displaying its XHTML contents adequately, or
- its contents can’t be translated to the requirements authoring tool adequately, the isSimplified flag must be set to true.

NOTE: The guideline for what is adequate is the default style sheet proposed by the W3C which maps HTML elements to CSS (http://www.w3.org/TR/CSS2/sample.html) and the CSS2.1 specification.

For the details on setting the isSimplified flag during the exchange process, see Clause 8.

Additional Information

10.8.21 DatatypeDefinition

**Package:** ReqIF  
**isAbstract:** Yes  
**Generalization:** Identifiable

**Description**

Abstract base class for all data types.

**Attributes**

No attributes

**Associations**

- coreContent : ReqIFContent [1]
  
  The back linkage to the owning ReqIFContent element.

**Operations**

No operations

**Constraints**

No constraints
Tags
No tags

Semantics
This is the abstract base class for all data types available to the Exchange Document.

Additional Information
No additional information

10.8.22 DatatypeDefinitionBoolean

Package: ReqIF
isAbstract: No
Generalization: DatatypeDefinitionSimple

Description
This class defines the primitive Boolean data type.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
This element defines a data type for the representation of Boolean data values in the Exchange Document. The representation of data values shall comply with the definitions in http://www.w3.org/TR/xmlschema-2/#boolean.

Additional Information
No additional information
10.8.23 DatatypeDefinitionDate

Package: ReqIF  
isAbstract: No  
Generalization: DatatypeDefinitionSimple

Description
This class defines the Date and Time data type.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
This element defines a data type for the representation of Date and Time data values in the Exchange Document. The representation of data values shall comply with the definitions in http://www.w3.org/TR/xmlschema-2/#isoformats.

Additional Information
No additional information

10.8.24 DatatypeDefinitionEnumeration

Package: ReqIF  
isAbstract: No  
Generalization: DatatypeDefinition

Description
Data type definition for enumeration types.

Attributes
No attributes
Associations

- specifiedValues : EnumValue [*] {composite, ordered}
  The linkage to the owned enumeration literals.

Operations

No operations

Constraints

No constraints

Tags

No tags

Semantics

Data type definition for enumeration types. The set of enumeration values referenced by specifiedValues constrains the possible choices for enumeration attribute values, as described in sub clause 10.8.15.

Additional Information

No additional information

10.8.25 DatatypeDefinitionInteger

Package: ReqIF
isAbstract: No
Generalization: DatatypeDefinitionSimple

Description

This class defines the primitive Integer data type.

Attributes

- max : integer
  Denotes the largest positive data value representable by this data type.
- min : integer
  Denotes the largest negative data value representable by this data type.

Associations

No associations

Operations

No operations
Constraints

[1] The value of the integer value held in any data element defined by DatatypeDefinitionInteger must be less than or equal to the value of DatatypeDefinitionInteger::max, and greater than or equal to the value of DatatypeDefinitionInteger::min.

Tags

No tags

Semantics

This element defines a data type for the representation of Integer data values in the Exchange Document. The representation of data values shall comply with the definitions in http://www.w3.org/TR/xmlschema-2/#integer.

Additional Information

No additional information

10.8.26 DatatypeDefinitionReal

Package: ReqIF
isAbstract: No
Generalization: DatatypeDefinitionSimple

Description

This class defines the primitive Real data type.

Attributes

- accuracy : integer
  Denotes the supported maximum precision of real numbers represented by this data type.
- max : float
  Denotes the largest positive data value representable by this data type.
- min : float
  Denotes the largest negative data value representable by this data type.

Associations

No associations

Operations

No operations

Constraints

[1] The value of the real value held in any data element defined by DatatypeDefinitionReal must be less than or equal to the value of DatatypeDefinitionReal::max, and greater than or equal to the value of DatatypeDefinitionReal::min.
Tags
No tags

Semantics
This element defines a data type for the representation of Real data values in the Exchange Document. The representation of data values shall comply with the definitions in http://www.w3.org/TR/xmlschema-2/#double. The precision of represented values is limited to the precision denoted by `DatatypeDefinitionReal::accuracy`.

Additional Information
No additional information

10.8.27 DatatypeDefinitionSimple

Package: ReqIF
isAbstract: Yes
Generalization: DatatypeDefinition

Description
Abstract base class for all primitive data types.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
DatatypeDefinitionSimple is the abstract base class from which all primitive data types, except enumeration, are derived.

Additional Information
No additional information
10.8.28 DatatypeDefinitionString

Package: ReqIF
isAbstract: No
Generalization: DatatypeDefinitionSimple

Description
This class defines the primitive String data type.

Attributes
• maxLength : integer
  The maximum permissible string length.

Associations
No associations

Operations
No operations

Constraints
[1] The length of the string value held in any data element defined by DatatypeDefinitionString must not exceed the value of DatatypeDefinitionString::maxLength.

Tags
No tags

Semantics
This element defines a data type for the representation of String data values in the Exchange Document. The representation of data values shall comply with the definitions in http://www.w3.org/TR/xmlschema-2/#string.

Additional Information
No additional information

10.8.29 DatatypeDefinitionXHTML

Package: ReqIF
isAbstract: No
Generalization: DatatypeDefinition

Description
Data type definition for XHTML formatted data.
Attributes
No attributes

Associations
No associations

Operations
No new operations

Constraints
No new constraints

Tags
No tags

Semantics
Data type definition for XHTML formatted data.

Additional Information
No additional information

10.8.30 EmbeddedValue

Package: ReqIF
isAbstract: No
Generalization:

Description
Class representing additional information related to enumeration literals.

Attributes

- key : integer
  The numerical value corresponding to the enumeration literal.

- otherContent : string
  Arbitrary additional information related to the enumeration literal (for example, a color.)

Associations

- enumValue : EnumValue [1]
  Back linkage to the owning EnumValue class.

Operations
No operations
**Constraints**

No constraints

**Tags**

No tags

**Semantics**

This class represents additional information related to enumeration literals.

**Additional Information**

No additional information

**10.8.31 EnumValue**

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** Identifiable

**Description**

Class representing enumeration literals.

**Attributes**

No attributes

**Associations**

- dataTpeDefEnum : DataTypeDefinitionEnumeration [1]
  
  Back linkage to the owning DatatypeDefinitionEnumeration class.

- Properties : EmbeddedValue [1] {composite}
  
  Link to owned EmbeddedValue.

**Operations**

No operations

**Constraints**

[1] The attribute longName inherited from Identifiable is mandatory for EnumValue.

**Tags**

No tags

**Semantics**

This class represents the enumeration literals.


**Additional Information**

No additional information

---

### 10.8.32 Identifiable

**Package:** ReqIF  
**isAbstract:** Yes  
**Generalization:** none

**Description**

Abstract base class providing an identification concept for ReqIF elements.

**Attributes**

- `desc : string [0..1]`
  Optional additional description for the information element.

- `identifier : string`
  The lifetime immutable identifier for an instance of a ReqIF information type. The value of the identifier must be a well-formed xsd:ID.

- `lastChange : xsd::dateTime`
  The date and time of the last change of the information element. This includes the creation of the information element. lastChange is of the XML Schema data type “dateTime” that specifies the time format as `CCYY-MM-DDThh:mm:ss` with optional time zone indicator as a suffix `±hh:mm`.  
  Example: 2005-03-04T10:24:18+01:00 (MET time zone).

- `longName : string [0..1]`
  The human-readable name for the information element.

**Associations**

- `alternativeID : Class [0..1] {composite}`
  The linkage to the optional alternative identification element.

**Operations**

No operations

**Constraints**

[1] The value of Identifiable::identifier must be globally unique.

**Tags**

No tags

**Semantics**

The Identifiable element provides globally unique and lifetime immutable identity to ReqIF elements. In addition, Identifiable provides change tracking for the derived ReqIF element, and provides for an optional human-readable name and an optional textual description for the derived ReqIF element.
Additional Information

While the longName attribute is optional from the viewpoint of Identifiable, some ReqIF elements make this long name mandatory. This fact will be stated in the class description of the affected elements.

10.8.33 RelationGroup

Package: ReqIF
isAbstract: No
Generalization: SpecElementWithAttributes

Description

Represents a group of relations.

Attributes

No attributes

Associations

• coreContent : ReqIFContent [1]
  The back linkage to the owning ReqIFContent element.

• specRelations : SpecRelation [*]
  Points to the grouped SpecRelations.

• type : RelationGroup Type [1]
  Linkage to the concrete SpecType instance.

• sourceSpecification: Specification [1]
  Reference to the specification that contains SpecObject instances that are source objects of the relations
  (referred to by the specRelations association).

• targetSpecification: Specification [1]
  Reference to the specification that contains SpecObject instances that are target objects of the relations
  (referred to by the specRelations association).

Operations

No new operations

Constraints

[1] The attribute longName inherited from Identifiable is mandatory for RelationGroup.

[2] For each SpecObject instance that is referred to by any SpecRelation instance in the set of specRelations
  (via the relation's source or target association) : the SpecObject instance must either be contained in the
  sourceSpecification or in the targetSpecification.

Tags

org.omg.reqif.order 6
Semantics
Represents a group of relations between a source specification and a target specification. For example, a RelationGroup instance may represent a set of relations between a customer requirements specification and a system requirements specification.

Additional Information
No additional information

10.8.34 RelationGroupType

Package: ReqIF
isAbstract: No
Generalization: SpecType

Description
Contains a set of attribute definitions for a RelationGroup element.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
Inherits a set of attribute definitions from SpecType. By using RelationGroupType elements, RelationGroup elements can be associated with attribute names, default values, data types, etc.

Additional Information
No additional information

10.8.35 ReqIFContent

Package: ReqIF
isAbstract: No
Generalization: none

Description
Core content root

Attributes
No attributes

Associations
• datatypes: DataTypeDefinition [0..*] {composite}
  Linkage to the DataTypeDefinition content elements.
• documentRoot: ReqIF [1]
  Linking back to the Exchange Document root element.
• specifications: Specification [0..*] {composite}
  Linkage to the Specification content elements.
• specObjects: SpecObject [0..*] {composite}
  Linkage to the SpecObject content elements.
• specRelationGroups: RelationGroup [0..*] {composite}
  Linkage to the RelationGroup content elements.
• specRelations: SpecRelation [0..*] {composite}
  Linkage to the SpecRelation content elements.
• specTypes: SpecType [0..*] {composite}
  Linkage to the SpecType content elements.

Operations
No operations

Constraints
No constraints

Tags
org.omg.reqif.order 2
org.omg.reqif.ordered true

Semantics
This element represents the root of the Exchange Document core content.

Additional Information
No additional information
10.8.36 SpecElementWithAttributes

Package: ReqIF
isAbstract: Yes
Generalization: Identifiable

Description
An abstract super class for elements that can own attributes.

Attributes
No attributes

Associations
• values : AttributeValue [0..*] {composite}
  The values of the attributes owned by the element.

Operations
No operations

Constraints
No constraints

Tags
None

Semantics
Any element that can own attributes, like a requirement, a specification, or a relation between requirements needs to be an instance of a concrete subclass of this abstract class.

While this class aggregates the values of the attributes, the association to the attributes’ types that define the acceptable values for the attributes is realized by concrete sub classes of this class.

Additional Information
No additional information

10.8.37 SpecHierarchy

Package: ReqIF
isAbstract: No
Generalization: AccessControlledElement
Description

Represents a node in a hierarchically structured requirements specification.

Attributes

- isTableInternal : Boolean[0..1]
  Some requirements authoring tools enable the user to use tables as part of a requirement’s content, where parts of
  the table represent requirements as well. If that is the case, this attribute needs to be set to true for the root node of
  the table hierarchy and all descendant SpecHierarchy nodes.

NOTE: The root node of the table hierarchy is related to the SpecObject element that is the root of the table by the object
association.

Associations

- children : SpecHierarchy [*] {composite, ordered}
  Down links to next level of owned SpecHierarchy.

- editableAtts : AttributeDefinition [*]
  The attributes whose values are editable for the SpecHierarchy by a tool user.

- parent : SpecHierarchy [0..1]
  Up link to previous level of SpecHierarchy (which owns this level).

- root : Specification [0..1]
  Up link to specification hierarchy root (which may own this level).

- object : SpecObject [1]
  Pointer to the associated SpecObject.

Operations

No operations

Constraints

[1] The value of isTableInternal is considered false if it is left out.

[2] If the inherited isEditable attribute is left out, the following constraint applies:

- If there is a parent SpecHierarchy element, the value of isEditable is copied from the parent SpecHierarchy element.
- If there is no parent SpecHierarchy element, the value of isEditable is false.

[3] If isEditable is false, the user of the requirements authoring must not replace the associated object with another
object.

[4] If isEditable is false, the user of the requirements authoring must not add or delete any direct children to/from the
SpecHierarchy element.

[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.
Tags
No tags

Semantics
Represents a node in a hierarchically structured requirements specification.

Additional Information
In most cases, the isTableInternal attribute may be set to false or left out. However, if at least one isTableInternal flag is set to true in an exchange document, a representation of each whole table must be exported as AttributeValueXHTML element to allow tools that can’t process table internal structures to represent them as formatted content.

10.8.38 Specification
Package: ReqIF
isAbstract: No
Generalization: SpecElementWithAttributes

Description
Represents a hierarchically structured requirements specification.

Attributes
No attributes

Associations
• children : SpecHierarchy [*] {composite, ordered}
  Links to next level of owned SpecHierarchy.
• coreContent : ReqIFContent [1]
  The back linkage to the owning ReqIFContent element.
• type : Specification Type [1]
  Linkage to the concrete SpecType instance.

Operations
No operations

Constraints
No constraints

Tags
org.omg.reqif.order      5

Semantics
Represents a hierarchically structured requirements specification.
It is the root node of the tree that hierarchically structures SpecObject instances.

Additional Information
No additional information

10.8.39 SpecificationType

Package: ReqIF
isAbstract: No
Generalization: SpecType

Description
Contains a set of attribute definitions for a Specification element.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
Inherits a set of attribute definitions from SpecType. By using SpecificationType elements, multiple specifications can be associated with the same set of attribute definitions (attribute names, default values, data types, etc.).

Additional Information
No additional information

10.8.40 SpecObject

Package: ReqIF
isAbstract: No
Generalization: SpecElementWithAttributes
**Description**
Constitutes an identifiable requirements object.

**Attributes**
No attributes

**Associations**
- `coreContent : ReqIFContent [1]`
  The back linkage to the owning ReqIFContent element.
- `type : SpecObject Type [1]`
  Linkage to the concrete SpecType instance.

**Operations**
No operations

**Constraints**
No constraints

**Tags**
org.omg.reqif.order 3

**Semantics**
Constitutes an identifiable requirements object that can be associated with various attributes. This is the smallest granularity by which requirements are referenced.

The `SpecObject` instance itself does not carry the requirements text or any other user defined content. This data is stored in `AttributeValue` instances that are associated to the `SpecObject` instance.

**Additional Information**
No additional information

### 10.8.41 SpecObjectType

**Package:** ReqIF  
**isAbstract:** No  
**Generalization:** SpecType

**Description**
Contains a set of attribute definitions for a SpecObject element.

**Attributes**
No attributes
Assessments
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags

Semantics
Inherits a set of attribute definitions from SpecType. By using SpecObjectType elements, multiple requirements can be associated with the same set of attribute definitions (attribute names, default values, data types, etc.).

Additional Information
No additional information

10.8.42 SpecRelation

Package: ReqIF
isAbstract: No
Generalization: SpecElementWithAttributes

Description
Defines relations (links) between two SpecObject instances.

Attributes
No attributes

Associations
- coreContent : ReqIFContent [1]
  The back linkage to the owning ReqIFContent element.
- source : SpecObject [1]
  Source object of the relationship.
- target : SpecObject [1]
  Target object of the relationship.
- type : SpecRelation Type [1]
  Linkage to the concrete SpecType instance.
Operations
No operations

Constraints
No constraints

Tags
org.omg.reqif.order 4
org.omg.reqif.reference.global “source,” “target”

Semantics
Defines relations (links) between two SpecObject instances.

Additional Information
No additional information

10.8.43 SpecRelationType

Package: ReqIF
isAbstract: No
Generalization: SpecType

Description
Contains a set of attribute definitions for a SpecRelation element.

Attributes
No attributes

Associations
No associations

Operations
No operations

Constraints
No constraints

Tags
No tags
Semantics
Inherits a set of attribute definitions from SpecType. By using SpecRelationType elements, multiple relations can be associated with the same set of attribute definitions (attribute names, default values, data types, etc.).

As an example, a requirement authoring tool may allow its users to define the new type “contradicts” for relations between two requirements that contradict each other, and associate a comment attribute with each relation that explains the contradiction.

Additional Information
No additional information

10.8.44 SpecType
Package: ReqIF
isAbstract: Yes
Generalization: Identifiable

Description
Contains a set of attribute definitions.

Attributes
No attributes

Associations
- coreContent : ReqIFContent [1]
  The back linkage to the owning ReqIFContent element.
- specAttributes: AttributeDefinition [0..*] {composite}
  The set of attribute definitions.

Operations
No operations

Constraints
No constraints

Tags
org.omg.reqif.order 2

Semantics
Contains a set of attribute definitions. By using an instance of a subclass of SpecType, multiple elements can be associated with the same set of attribute definitions (attribute names, default values, data types, etc.).
10.8.45 XhtmlContent

**Package:** ReqIF

**isAbstract:** No

**Generalization:**

**Description**

Class representing XHTML content.

**Attributes**

No attributes

**Associations**

- `attributeValue : AttributeValueXHTML [1]`
  Back linkage to the owning AttributeValueXHTML class.

**Operations**

No operations

**Constraints**

No constraints

**Tags**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.omg.reqif.datatype</td>
<td>true</td>
</tr>
<tr>
<td>org.omg.reqif.max</td>
<td>1</td>
</tr>
<tr>
<td>org.omg.reqif.min</td>
<td>1</td>
</tr>
<tr>
<td>org.omg.reqif.nsURI</td>
<td><a href="http://www.w3.org/1999/xhtml">http://www.w3.org/1999/xhtml</a></td>
</tr>
<tr>
<td>org.omg.reqif.processContents</td>
<td>strict</td>
</tr>
</tbody>
</table>

**Semantics**

This class represents XHTML formatted content.

**Additional Information**

No additional information
11 Production Rules of ReqIF XML Schema

11.1 Purpose

This clause describes the rules for creating a schema from the reqif metamodel.

11.2 Notation for EBNF

The rule sets are stated in EBNF notation. Each rule is numbered for reference. Rules are written as rule number, rule name, for example 1a. SchemaStart. Text within quotation marks are literal values, for example “<xsd:element>.”

Text enclosed in double slashes represents a placeholder to be filled in with the appropriate external value, for example /Name of Attribute/. Literals should be enclosed in single or double quotation marks when used as the values for XML attributes in XML documents. The suffix “*” is used to indicate repetition of an item 0 or more times. The suffix “?” is used to indicate repetition of an item 0 or 1 times. The suffix “+” is used to indicate repetition of an item 1 or more times.

The vertical bar “|” indicates a choice between two items. Parentheses “( )” are used for grouping items together. EBNF ignores white space; hence these rules do not specify white space treatment. However, since white space in XML is significant, the actual schema generation process must insert white space at the appropriate points.

11.3 Tags

Some defined tags control the production rules.

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.global_element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>true</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Marks the class as root element.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>There should be exactly one class with value true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.xsd_element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>Collection of attribute names.</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Defines how the attribute from the metamodel is represented in the schema.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.xsd_attribute_reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>Collection of attribute names.</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Defines that the attributes in the collection are represented as xsd:attribute with ref attribute.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
<tr>
<td>Tag id:</td>
<td>org.omg.reqif.ordered</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Values:</td>
<td>true</td>
</tr>
<tr>
<td>Meaning:</td>
<td>The composite properties of the class have a defined order.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>one Integer</td>
</tr>
<tr>
<td>Meaning:</td>
<td>The position the class in its parent class.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.reference.global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>Collection of target property names.</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Indicates if a reference can point to an element in an external document.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>only if property is composite</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>Collection of attribute names.</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Add a fixed attribute to the XML attribute element.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>true</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Marks the class as datatype if value is set to true.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.nsURI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>one String</td>
</tr>
<tr>
<td>Tag id:</td>
<td>org.omg.reqif.processContents</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Values:</td>
<td>&quot;skip&quot;</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Add an processContents attribute to the XML element.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Ignored if org.omg.reqif.datatype not equals true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>one Integer</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Specifies a minimal value.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Ignored if org.omg.reqif.datatype not equals true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>one Integer</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Specifies a maximal value.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Ignored if org.omg.reqif.datatype not equals true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag id:</th>
<th>org.omg.reqif.xhtml_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>true</td>
</tr>
<tr>
<td>Meaning:</td>
<td>Marks the class as xhtml type.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Ignored if org.omg.reqif.datatype not equals true.</td>
</tr>
</tbody>
</table>

### 11.4 EBNF

The EBNF for ReqIF schemas is listed below with rule description between sections.

1. Schema ::= 1a:SchemaStart
    id:XHTMLImports
    2:PackageSchema
    1e:SchemaEnd
1. A schema consists of a schema XML element that contains import statements and declarations for the contents of the packages in the metamodel.

1a. The schema XML element consists of the schema namespace attribute, namespace attributes for the other namespaces used in the schema.

1b. The name of the Reqif namespace.

1c. The URI of the Reqif namespace.

1d. Fixed driver import declaration for xhtml module schemas.

1e. The end of the schema XML element.

2. The schema contribution from a package consists of the declarations am global element, fixed reference types, classes, enumerations, and type definitions.

3. GlobalElement::="<xsd:element

name='" 3a:GlobalElementName "'
type='" 1b:Namespace " : " 3b:GlobalElementType"'>
</xsd:element>"
3a. GlobalElementName ::= // Name of Global Element //
3b. GlobalElementType ::= // Name of Global Element Type //

3. If the tag org.omg.reqif.global_element is true, the rule describes the declaration of a global element in the metamodel as an element. In the package there should be exactly one element with tag org.omg.reqif.global_element set to true.
3a. The name of the global element.
3b. The type of the global element.

4. FixedRefTypes::="<xsd:simpleType name='LOCAL-REF'>
    <xsd:restriction base='xsd:IDREF'/>
</xsd:simpleType>
<xsd:simpleType name='GLOBAL-REF'>
    <xsd:restriction base='xsd:string'/>
</xsd:simpleType>"

4. This rule declares two simple types to be used as type in non-containment associations. LOCAL-REF wraps xsd:IDREF type to point to an identifier inside the same document. GLOBAL-REF can point to an identifier in an arbitrary document.

5. ClassTypeDef ::="<xsd:complexType name='"//Name of Class// "]">
    ( "<xsd:sequence" | "<xsd:all>" )
    ( 5a:ClassElementAttribute )*
    ( 5b:ClassReferences )*
    ( 5c:ClassCompositions ) *
    ( "</xsd:sequence" | "</xsd:all>" )
    ( 5d:ClassAttribute )*
    ( 5e:ClassAttributeRef)*
</xsd:complexType>"

5a. ClassElementAttribute ::="<xsd:element
    name='"//Name of Attribute// "]'
    minOccurs='"// Minimum // "]'
    maxOccurs='"// Maximum // "]'
    5g:FixedAttribute
    "type='"//Name of Attribute Type// "]"/>

5b. ClassReferences ::= "<xsd:element name='"// Name of Target Property // "]'
    minOccurs="0' | 1'"
    maxOccurs='1'>
</xsd:complexType>"
5. These rules describe the declaration of a class in the metamodel as an XML complex type with XML attributes and content elements. If the tag org.omg.reqif.ordered is true, the contents of the class are put in a sequence, otherwise they are put in an XML all element. Content classes that put in the sequence should be tagged by org.omg.reqif.order tag, which defines the position by an integer value. Classes in the metamodel with a tag org.omg.reqif.xsd_*use the rules 5a, 5d, or 5e.

5a. XML elements for the attributes of the class if the name is contained in the values of the tag org.omg.reqif.xsd_element or the target class is the data type XhtmlContent. Inherited attributes are also included.
5b. The XML element for each reference of the class that is no composite reference. The name is the name of target property. The attribute minOccurs is set to 0 if the multiplicity lower equal 0 else to 1, maxOccurs is always set to 1. The element is defined by a complex type. The included choice element represents the multiplicities of the reference. The minOccurs attribute shows the lower value of the reference target property, maxOccurs the upper value. The choice element contains one of more elements. The name of the element is the name of the association target class or if this class is abstract, the name of the non-abstract sub class, decorated with -REF. The type of the element is one of the reference types defined in 4. If the association target role name is contained in the value of the tag org.omg.reqif.reference.global, GLOBAL-REF will be appended, else LOCAL-REF. Global means that the reference can point to an Element outside this document. Inherited references are also included.

5c. This rule applies to references that are composite. It differs from rule 5b only in the definition of the element type of the association target class.

5d. Attributes of the class in the metamodel which names are not values of the tag org.omg.reqif.xsd_element are declared as XML attributes with name, type, and use attributes.

5e. Attributes of the class in the metamodel which names are values of the tag org.omg.reqif.xsd_attribute_reference are declared as XML attributes with a ref attribute which refers to the referenced element and a use attribute.

5f. The attribute use controls the use of the containing element. The value is derived from the multiplicity of the attribute from the metamodel. If the upper value is 0, then prohibited is used, else if the lower value is 0, then use is set to optional, else if the lower value is greater than 0, required is used.

5g. If the name of the attribute is contained in the values of the tag org.omg.reqif.fixed, this rule is applied.

6. EnumSchema ::= "<xsd:simpleType name='' // Name of Enumeration Class // '' >
    <xsd:restriction base='xsd:string'>
        { 6a:EnumLiteral }*
    </xsd:restriction>
</xsd:simpleType>"

6a. EnumLiteral ::= "<xsd:enumeration
    value='' // Name of Literal from Enumeration Class // '' />

6. The enumeration schema contribution consists of a simple type derived from string whose legal values are the enumeration literals.

6a. Each enumeration literal is put in the value XML attribute of an enumeration XML element.

7. TypeSchema ::= ( 7a:DatatypeSchema | 7b:XhtmlType)
7a. DatatypeSchema ::= "<xsd:complexType
    name='' // Name of Datatype // '' >
    <xsd:sequence>
NOTE: The names of the XML elements are constructed by converting the information type’s name into uppercase letters with additional hyphens (“-”) indicating word separations that have originally been indicated by uppercase letters of by a numeric character inside the name. Thus, by definition, a “word” is one of the following:

- First letter is upper-case followed by lower-case letters.
- All letters are upper-case.
- Contiguous numeric characters.

For example, the name TestECUClass12ADC is converted into an XML element with name TEST-ECU-CLASS-12-ADC.
<xsd:include schemaLocation='xhtml-hypertext-1.xsd'/>
<xsd:include schemaLocation='xhtml-image-1.xsd'/>
<xsd:include schemaLocation='xhtml-object-1.xsd'/>
<xsd:include schemaLocation='xhtml-text-1.xsd'/>
<xsd:include schemaLocation='xhtml-list-1.xsd'/>
<xsd:include schemaLocation='xhtml-table-1.xsd'/>
<xsd:include schemaLocation='xhtml-edit-1.xsd'/>
<xsd:include schemaLocation='xhtml-pres-1.xsd'/>
<xsd:include schemaLocation='xhtml-form-1.xsd'/>
<xsd:include schemaLocation='xhtml-meta-1.xsd'/>
<xsd:include schemaLocation='xhtml-link-1.xsd'/>
<xsd:include schemaLocation='xhtml-struct-1.xsd'/>
</xsd:include>
Annex A: Implementation Guide

(informative)

Additional informative technical descriptions that help to implement a ReqIF capable tool can be found in the ReqIF implementation guide at the following location:

http://www.prostep.org/de/downloads/guidelines-use-cases.html