W3C Working Draft

RIF Combination with XML data

W3C Working Draft 22 June 2010

This version:
http://www.w3.org/TR/2010/WD-rif-xml-data-20100622/
Latest version:
http://www.w3.org/TR/rif-xml-data/
Previous version:
http://www.w3.org/TR/2010/WD-rif-xml-data-20100511/ (color-coded diff)
Editors:
Christian de Sainte Marie, IBM

This document is also available in these non-normative formats: PDF version.

Abstract

This document, developed by the Rule Interchange Format (RIF) Working Group, specifies how a RIF document can be combined with XML data.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Set of Documents

This document is being published as one of a set of 11 documents:

W3C Working Draft

1. RIF Overview
2. RIF Core Dialect
3. RIF Basic Logic Dialect
4. RIF Production Rule Dialect
5. RIF Framework for Logic Dialects
6. RIF Datatypes and Built-Ins 1.0
7. RIF RDF and OWL Compatibility
8. OWL 2 RL in RIF
9. RIF Combination with XML data (this document)
10. RIF In RDF
11. RIF Test Cases

XML Schema Datatypes Dependency

RIF is defined to use datatypes defined in the XML Schema Definition Language (XSD). As of this writing, the latest W3C Recommendation for XSD is version 1.0, with version 1.1 progressing toward Recommendation. RIF has been designed to take advantage of the new datatypes and clearer explanations available in XSD 1.1, but for now those advantages are being partially put on hold. Specifically, until XSD 1.1 becomes a W3C Recommendation, the elements of RIF which are based on it should be considered *optional*, as detailed in Datatypes and Builtins, section 2.3. Upon the publication of XSD 1.1 as a W3C Recommendation, those elements will cease to be optional and are to be considered required as otherwise specified.

We suggest that for now developers and users follow the XSD 1.1 Last Call Working Draft. Based on discussions between the Schema, RIF and OWL Working Groups, we do not expect any implementation changes will be necessary as XSD 1.1 advances to Recommendation.

Summary of Changes

The semantics were completely reworked.

Please Comment By 20 July 2010

The Rule Interchange Format (RIF) Working Group seeks public feedback on this Working Draft. Please send your comments to public-rif-comments@w3.org (public archive). If possible, please offer specific changes to the text that would address your concern. You may also wish to check the Wiki Version of this document and see if the relevant text has already been updated.

W3C Working Draft

No Endorsement

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

- 1 Overview
- 2 Importing XML documents and schemas in RIF
- 3 A simple data model for XML documents
  - 3.1 Definitions
  - 3.2 Element information items
  - 3.3 Attribute information items
  - 3.4 Character information items
  - 3.5 Resolution of references
  - 3.6 Example of a data model instance
- 4 RIF combination with XML data
  - 4.1 Model-theoretic semantics of RIF BLD+XML data combinations
    - 4.1.1 Combined interpretation of RIF BLD non-document formulas and schemaless XML data
    - 4.1.2 Combined interpretation of RIF BLD non-document formulas and schema valid XML data
    - 4.1.3 Combined interpretation of RIF BLD documents and XML data
  - 4.2 Operational semantics of RIF PRD+XML data combinations
  - 4.3 Semantics of RIF Core+XML data combinations
- 5 Conformance
- 6 The special case of RDF and OWL data sources

- 7 References
- 8 Appendix A: Glossary (non-normative)
- 9 Appendix B: Extract from the XQuery 1.0 and XPath 2.0 Data Model (non-normative)
  - 9.1 Element and attribute node type names (from [XDM])
  - 9.2 Typed value determination (from [XDM])
- 10 Appendix C: Embedding imported data sources as RIF facts (non-normative)
- 11 Appendix D: Examples using the normative RIF/XML syntax
- 12 Appendix E: Change Log (non-normative)

1 Overview

The Rule Interchange Format (RIF) is a format for interchanging rules over the Web. Rules that are exchanged using RIF may refer to external data sources and may be based on data models that are represented using a language different from RIF. This document specifies how combinations of RIF documents and XML data, possibly associated with XML schemas, are interpreted.

Extensible Markup Language (XML) is a simple, flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. The XML Schema Definition Language offers facilities for describing the structure and constraining the contents of XML documents. The schema language, which is itself represented in an XML vocabulary, provides a way to describe and to share the data model that is associated with the data in an XML document.

This document specifies a standard semantics for combinations of RIF documents and XML data. It uses a data model for XML documents, that is a simplified version of the XQuery 1.0 and XPath 2.0 Data Model [XDM].

The XQuery 1.0 and XPath 2.0 Data Model (XDM) specifies what information is accessible in a collection of XML documents, but it does not specify the language used to represent or access the data: this document specifies an implementation, using the RIF condition language, of a simplified version of the XDM. This makes the RIF condition language comparable to other implementations of the XDM, such as [XPath 2.0] and [XQuery 1.0].

Like the XQuery 1.0 and XPath 2.0 Data Model, the simplified version used in this document supports the following classes of XML documents:

- Well-formed documents conforming to [Namespaces in XML] or [Namespaces in XML 1.1].
- DTD-valid documents conforming to [Namespaces in XML] or [Namespaces in XML 1.1], and
- W3C XML Schema-validated documents.

Accordingly, this document specifies how a RIF document is combined with well-formed, and, where an XML schema is specified, schema-valid XML documents. The semantics is independent on the provenance of the XML data in a combination: it can be imported explicitly in a RIF document, or combined on the consumer-side, or a combination of both. However, only XML schemas that are explicitly imported in the RIF document are taken into account for the interpretation of the combination. This provides a way to communicate the data model that is intended, in a RIF document, for the data source, without specifying an actual data source.

Section 2 specifies how the `rif:Import` directive is used to import an XML document and an XML schema are import in a RIF document.

The RIF data model for XML documents is described in section 3.

Section 4 specifies a standard semantics for RIF combinations with XML data: first, a model-theoretics semantics is given to RIF BLD combination with XML data, with and without associated XML schema (section 4.1); the operational semantics of RIF PRD combinations with XML data is, then, defined, based on the definition of a RIF BLD+XML data combined interpretation (section 4.2); finally, the semantics of RIF Core combinations with XML data is defined with respect to the model-theoretic semantics of the combination of RIF BLD and XML data and the operational semantics of the combination of RIF PRD and XML data (section 4.3).

Editor's Note: This section will be completed in a future draft.

2 Importing XML documents and schemas in RIF

In RIF, the `Import` directive is used to communicate the location of an external document to be combined with the RIF document that contains the directive and, optionally, a profile that governs the combination.

In [RIF-Core], [RIF-PRD] and [RIF-BLD], the use of the `Import` directive is limited to identifying an imported RIF document, or an RDF graph or an OWL ontology to be combined with a RIF document. An optional profile that governs the combination of a RIF document with an RDF graph or an OWL ontology can also be provided, as specified in [RIF-RDF-OWL].

This specification extends the `Import` directive in four ways:

1. The IRI of any XML document is allowed as a value of the `location` sub-element, that contains the IRI that identifies the imported data document;
2. The new value: `http://www.w3.org/2007/rif-import-profile#xml-data`, is defined for the profile of an import, indicating that the imported document is an XML document with no associated XML schema;
3. In addition, any IRI that identifies an XML schema document is allowed as a profile, identifying the XML schema that is associated with the imported XML data;
4. Finally, a new value: `http://www.w3.org/2007/rif-import-location#no-data`, is allowed for the location of an import, indicating that the directive does not require the import of a data document

Editor's Note: The extended syntax is still under discussion. Other approaches include: (i) to make the location sub-element optional as well; (ii) to reserve new keywords in the `http://www.w3.org/2007/rif-import-profile` namespace, such as: `xml-schema`, `xml-schema-valid-data`, etc, and to add a specific construct for the schema locator.

The following constraints must be satisfied:

- If an `Import` directive does not require the import of a data document, the `profile` must be specified;
- The two special values `http://www.w3.org/2007/rif-import-location#no-data`, for the location, and `http://www.w3.org/2007/rif-import-profile#xml-data`, for the profile, are not allowed in the same `Import` directive;
- If the `profile` of an `Import` directive identifies an XML Schema and if the data document provides a `schema-location`, the `profile` must identify the same XML schema as the `schema-location` IRI;
- If the profile is `rif:xml-data`, the `location` must identify an XML document.

This specification does not prescribe the behaviour of a conformant implementation when one of the above constraints is not satisfied.

This specification does not prescribe the behaviour of a conformant implementation when an `Import` directive contains a `profile` that is neither `rif:xml-data` nor an IRI that identifies an XML schema.

Example 2.1. The first three import directives, below, are valid; the fourth is not:

1. `Import(http://example.org/customertable.xml http://www.w3.org/2007/rif-import-profile#xml-data)`
2. `Import(http://example.org/customertable.xml http://example.org/customertable.xsd)`
3. `Import(http://www.w3.org/2007/rif-import-location#no-data http://example.org/customertable.xsd)`
4. `Import(http://www.w3.org/2007/rif-import-location#no-data http://www.w3.org/2007/rif-import-profile#xml-data)`

The first directive says that the rules in the importing RIF document are to be combined with the data in the XML document identified by the IRI: `http://example.org/customertable.xml` and that there is no data model associated with the imported data in the form of an XML schema.

The second directive says that the rules in the importing RIF document are to be combined with the data in the XML document identified by the IRI: `http://example.org/customertable.xml` and that there is a data model associated with the imported data, in the form of the XML schema that is identified by the IRI: `http://example.org/customertable.xsd`.

The third directive says the data that is combined with the rules is expected to be an instance of the data model that is imported as the XML schema identified by the IRI: `http://example.org/customertable.xsd`; but the directive does not say what data is to be combined with the rules.

The fourth directive violates one of the constraint: therefore, it is out of the scope of this specification.

Notice that none of the three valid directives is incompatible with the other two, but that combining the first two is confusing and error-prone, and should be avoided; and that the third one is redundant with the second, and that combining the two is useless and confusing, and that it should be avoided.

3 A simple data model for XML documents

This section defines the RIF data model for XML documents (henceforth "the data model"). The data model serves two purposes: it specifies the information that is accessible to a RIF consumer in an XML document, possibly in combination with an XML schema; and it is used to specify the interpretation of the combination of RIF with XML data.

The data model is a simplified version of the XQuery 1.0 and XPath 2.0 Data Model [XDM]. As a consequence, the RIF condition language can be considered a partial implementation of the XQuery 1.0 and XPath 2.0 Data Model, and the interpretation of the RIF condition language with respect to XML documents can be specified in terms of the XQuery 1.0

and XPath 2.0 Data Model or its implementations, such as XPath 2.0 [XPath 2.0]. This document will reuse definitions from the XQuery 1.0 and XPath 2.0 Data Model and the XPath 2.0 specifications, as appropriate, and otherwise provide pointers and examples where relevant.

3.1 Definitions

The data model specifies the *information items* from the XML infoset [Infoset] and from the post-schema validation infoset (PSVI), or derived from the infoset and the PSVI, that are required to interpret some RIF constructs with respect to an XML document.

Definition (Information item). *(from [Infoset])* An ***information item*** is an abstract description of some part of an XML document: each information item has a set of associated named properties. □

In this document, an information item is said to be *constructed from an infoset*, if the data item that it describes is contained in a data source that is not associated with an XML schema when it is imported in RIF. If the data source is associated with an XML schema, all the information items used to describe the content of that data source are said to be *constructed from a PSVI*.

If an information item is constructed from an infoset, all general and external parsed entities must be fully expanded before the data model is constructed.

In this specification, the property names are shown in square brackets, **[thus]**.

The data model relies on three types of information items:

- *Element information items*, that describe the elements in an XML document. The properties associated with each element information item are: [namespace name], [local name], [children], [root], [attributes], [type name], [string value], [typed value], [is-id], [is-idrefs];
- *Attribute information items*, that describe the attributes of elements. The properties associated with the attribute information item are: [namespace name], [local name], [attribute type], [owner element], [type name], [string value], [typed value], [is-id], [is-idrefs];
- *Character information items*, that describe the data characters that appear in the XML document. The character information item has two properties: [character code] and [element content whitespace].

Given a data source, the relevant set of information items may be created by methods other than parsing and/or schema-validating an XML document. This specification does not describe or prescribe any method for retrieving the required information from a data source, possibly combined with an XML schema.

This specification distinguishes between the data model as a general concept and specific items (information items or atomic values) that are concrete examples of the data model. For the purpose of this specification, the term *instance of the data model* will be used exclusively to denote such concrete examples of the data model that are sequences of element information items in document order. **Definition (Instance of the data model).** An ***instance of the data model*** is a sequence of element information items, in document order. In particular, given an XML document *D*, the *instance of the data model that describes D* is the sequence of all the element information items that describe an element contained in *D*, in document order. ☐

When there is no ambiguity with respect to *D*, the instance of the data model that describes *D* will be called, simply: *the instance of the data model*.

Definition (Atomic value). An ***atomic value*** is a value in the value space of an atomic type and is labeled with the name of that atomic type. ☐

Definition (Atomic type). An ***atomic type*** is one of the 20 primitive simple types defined in Section 3.3 Primitive Datatypes of [XSD 1.1 Part 2] or a type derived by restriction from another atomic type. ☐

Types derived by list or union are not atomic.

Definition (Sequence). A ***sequence*** is an ordered collection of zero or more information items. ☐

A sequence cannot be a member of a sequence. An important characteristic of the data model is that there is no distinction between an item (a information item or an *atomic value*) and a singleton sequence containing that item. An item is equivalent to a singleton sequence containing that item and vice versa.

Except when specified otherwise, sequences are ordered according to the *document order*.

Definition (Document order). A ***document order*** is defined among all the element information items that describe a given XML document. Document order is a total ordering. Informally, document order is the order in which nodes appear in the XML serialization of a document. ☐

Within a tree, *document order* satisfies the following constraints:

1. The root node is the first node.
2. Every node occurs before all of its children and descendants.
3. The relative order of siblings is the order in which they occur in the [children] property of their parent node.
4. Children and descendants occur before following siblings.

XML element, attribute and type names are usually represented as XML qualified names, or QNames. However, `xs:QName` is not a RIF-Core built-in datatype. In the data model, all qualified names, including atomic values, are represented as *expanded QNames*.

Definition (Expanded QName). An ***expanded QName*** is a set of three values consisting of a possibly empty prefix, a possibly empty namespace IRI and a local name. □

Notice that the prefix is never used in this document, and expanded QNames will be dealt with, in all but definition, as consisting of a possibly empty namespace IRI and a local name

3.2 Element information items

There is an *element information item* for each element appearing in the XML document. One of the element information items corresponds to the root of the element tree, and all other element information items are accessible by recursively following its [children] property.

An element information item has the following properties:

1. [**namespace name**] The namespace name, if any, of the element. If the element does not belong to a namespace, this property has no value;
2. [**local name**] The local part of the element name. This does not include any namespace prefix or following colon;
3. [**children**] An ordered list of child information items, in document order. This list contains only *element information items* and *character information items*, one for each element and data character appearing immediately within the current element. It does not contain other kinds of information items, such as *attribute information items*, even if the XML element described by the information item has attributes. If the element is empty, this list has no members;
4. [**root**] The element information item that describes the root element in the XML document. The [root] property of all the element information items that describe elements contained in the same XML document point to the same root element information item, including that root element information item itself. If the [root] property of an element information item points

to that element information item itself, the [root] properties of all the element information items that are accessible by following its [children] property, recursively, must point to that same root element information item;

5. [**attributes**] An unordered set of attribute information items, one for each of the attributes of this element. This includes all of the "special" attributes (`xml:lang`, `xml:space`, `xsi:type`, etc.) but does not include namespace declarations (because they are not attributes). Default and fixed attributes, e.g. provided by XML Schema processing, are added to the [attributes]. If the element has no attributes, this set has no members;
6. [**type name**] The [type name] property of an element information item is empty if, and only if, the element information item is constructed from an infoset. If the element information item is constructed from a PSVI, the type name is represented by an expanded QName. It is determined as described in Section 3.3.1.1. Element and Attribute Node Type Names from [XDM] (reproduced in appendix 9.1, Element and attribute node type names, for the reader's convenience);
7. [**string value**] The normalised representation of the content of the element as a string. The string value is calculated as follows:
    - If the element is empty or if it does not have any character information item children nor descendants, its string value is the zero length string;
    - Else, if the [type name] property of the element information item is empty, or if the element has a complex type with element-only content, or a complex type with mixed content, its string value is the string comprised of characters that correspond to the [character code] properties of each of the character information item children of the element and all its descendants, in document order. If the resulting string consists entirely of whitespace and the [element content whitespace] property of the character information items used to construct it are true, the string value is the zero-length string;
    - Else, if the element has a simple type or a complex type with simple content: its string value is the schema normalized value of the element;
    - Note that, if the element has a typed value, any valid lexical representation of the typed value can be used to determine the [string value] property;
8. [**typed value**] The typed-value is calculated as follows:
    - If the element is empty or if it has element-only children, its typed value is the element itself; more precisely, if an element information item has no character information items in its [children] property, the value of its [typed value] property is the element information item itself. This is a deviation from the XQuery 1.0 and XPath 2.0 Data Model: the latter document specifies the [typed

value] as being undefined, in that case, which is of no use in this specification; whereas this specification requires a handle to an element information item, to access its [children] and [attributes] properties from its [typed value] (see the definitions of combined interpretations in Section 4, RIF combination with XML data);
    - Else, if the [type name] property of the element information item is empty, or if the element has a complex type with mixed content (including xs:anyType), its typed value is the same as its string value;
    - Otherwise, the element must have a simple type or a complex type with simple content. Its typed value is computed as described in Section 3.3.1.2 Typed Value Determination, in [XDM] (reproduced in appendix 9.2, Typed value determination, for the reader's convenience). The result is a sequence of zero or more atomic values. The relationship between the values of the [type name], [typed value], and [string value] properties of an element information item is consistent with XML Schema validation. Note that in the case of `xs:QNames` and `xs:NOTATIONs`, the prefix is not preserved, and the typed values are represented as expanded QNames. This is a minor deviation from the XQuery 1.0 and XPath 2.0 Data Model, but this specification does not use the prefix;
9. [is-id] If the [type name] property of an element information item is empty, or if the element has a complex type with element-only content, the [is-id] property is false. Else, if the typed value of the element consists of exactly one atomic value, that value is of type `xs:ID`, or a type derived from `xs:ID`, the [is-id] property is true; otherwise it is false;
10. [is-idrefs] If the [type name] property of an element information item is empty, or if the element has a complex type with element-only content, the [is-idrefs] property is false. Else, if any of the atomic values in the typed-value of the element is of type `xs:IDREF` or `xs:IDREFS`, or a type derived from one of those types, the [is-idrefs] property is true; otherwise it is false.

Editor's Note: Although minor, the deviations from the XQuery 1.0 and XPath 2.0 Data Model (XDM) may preclude using code developed for XDM. They are still under discussion. The working group is seeking feedback on the issue (ISSUE-103).

3.3 Attribute information items

There is an attribute information item for each attribute (specified or defaulted) of each element in the document, excluding those which are namespace declarations (because they are not attributes).

Attributes declared in the DTD with no default value and not specified in the element's start tag are not represented by attribute information items.

An attribute information item has the following properties:

1. [**namespace name**] The namespace name, if any, of the attribute. Otherwise, this property has no value;
2. [**local name**] The local part of the attribute name. This does not include any namespace prefix or following colon;
3. [**attribute type**] An indication of the type declared for this attribute in the DTD. The only values that are relevant to this specification are `ID`, `IDREF` and `IDREFS`. If there is no declaration for the attribute, or if no declaration has been read, this property has no value. The value of this property is not affected by the validity of the attribute value;
4. [**owner element**] The element information item which contains this information item in its [attributes] property;
5. [**type name**] Empty if the attribute information item is constructed from an infoset. If the attribute information item is constructed from a PSVI, the type name is represented as an expanded QName, and it is determined as described in Section 3.3.1.1. Element and Attribute Node Type Names from [XDM] (reproduced in Appendix B for the reader's convenience);
6. [**string value**] The schema normalized value PSVI property if that exists; otherwise, the normalized attribute value according to Section 3.3.3 Attribute-Value Normalization in [XML]). Note that, if the attribute has a typed value, any valid lexical representation of the typed value can be used to determine the string value;
7. [**typed value**] The typed-value is calculated as follows:
    - If the [type name] property of an attribute information item is empty, its typed value is the same as its string value;
    - Otherwise, a sequence of zero or more atomic values as described in Section 3.3.1.2 Typed Value Determination, in [XDM] (reproduced in Appendix B for the reader's convenience). The relationship between the values of the [type name], [typed value], and [string value] properties of an attribute information item is consistent with XML Schema validation.
8. [**is-id**] If the attribute is named `xml:id` and its [attribute type] property does not have the value `ID` and its [type name]

property does not have the value `xs:ID`, then [xml:id] processing is performed. This will assure that the value does have the type `ID` or `xs:ID` (if the attribute information item is constructed from an infoset or from a PSVI, respectively) and that it is properly normalized. The [is-id] property is always true for attributes named `xml:id`. Else, if the [attribute type] property has the value `ID`, or if the type name is `xs:ID` or a type derived from `xs:ID`, the [is-id] property is true; otherwise, it is false. This specicification does not prescribe the behaviour of a RIF consumer application if an error is encountered during [xml:id] processing.

9. [**is-idrefs**] True if the value of the [attribute type] property is `IDREF` or `IDREFS`, or if any of the atomic values in the typed-value of the attribute is of type `xs:IDREF` or `xs:IDREFS`, or a type derived from one of those types. Otherwise, false.

3.4 Character information items

There is a character information item for each data character that appears in the document, whether literally, as a character reference, or within a CDATA section.

Each character is a logically separate information item, but applications are free to chunk characters into larger groups as necessary or desirable.

A character information item has the following properties:

1. [**character code**] The ISO 10646 character code (in the range 0 to #x10FFFF, though not every value in this range is a legal XML character code) of the character.
2. [**element content whitespace**] A boolean indicating whether the character is white space appearing within element content (see [XML], Section 2.10. White Space Handling). Note that validating XML processors are required to provide this information. If there is no declaration for the containing element, or if there are multiple declarations, or if no declaration has been read, this property has no value for white space characters. It is always false for characters that are not white space.

3.5 Resolution of references

Definition (Reference information item). Given an information item, *I*, whose [is-id] is true, and given an atomic value, *id*, of type `ID`, `IDREF`, `xs:ID`, or `xs:IDREF`, that matches one of the atomic values in *I*'s [typed value] property, the ***reference information item*** identified by *id* is the element information item, *R*, such that

- its [root] property has the same value as the [root] property of *I*, if *I* is an element information item, or as the [root] property of

the element information item pointed to by the [owner element] property of *I*, if *I* is an attribute information element;
- and
  - either the [is-id] property of *R* is true and its [typed value] property matches *id*;
  - or the [is-id] property of one of the attribute information items in *R*'s [attributes] property is true, and the [typed value] property of that attribute information item matches *id*. □

Note that the reference information item is always an *element*.

Where this specification states, with respect to a set of information items, that the *references are resolved*, the following processing is applied to every information item in the set whose [id-refs] property is true:

- If the information item is an element information item, *E*, and
  - if its typed value is a single atomic value of type `xs:IDREF` or `xs:IDREFS` or a type derived from one of those types, *E* itself is replaced with its reference information item;
  - or, if its typed value is a sequence that contains more than one atomic value, each atomic value that is of type `xs:IDREF` or `xs:IDREFS` or a type derived from one of those types is replaced with the typed value of its reference information item, after the references in that typed value have been resolved, if the [id-refs] property of the reference information item is true;
- otherwise, the information item must be an attribute information item, *A*, and every atomic value in the typed value of *A* is replaced with its reference information item.

No property value is changed in any information item: the references are resolved only on a "need-to-resolve" basis, where the specification of RIF as an implementation of the data model requires them to be resolved.

This specification does not prescribe a behavior if an error is encountered during reference resolution processing (such as `IDREF`s without corresponding `ID`s, invalid or duplicate `ID`s, etc).

3.6 Example of a data model instance

Consider the following XML document, representing data about customers:

```
<CustomerTable xmlns="http://example.org/customertable"
               xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
```

```
    <Name> John </Name>
    <Account> 111 </Account>
  </Customer>
  <Customer xml:lang="fr">
    <Name> Jane </Name>
    <Account> 222 </Account>
    <PIN> 222 </PIN>
  </Customer>
</CustomerTable>
```

Consider, further, the following XML schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xml="http://www.w3.org/XML/1998/namespace"
           targetNamespace="<nowiki>http://example.org/customertable</nowik
           xmlns="<nowiki>http://example.org/customertable</nowiki>">

  <xs:simpleType name="PIN">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="100"/>
      <xs:maxExclusive value="1000"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Account" type="xs:integer"/>
  <xs:element name="PIN" type="PIN"/>

  <xs:element name="Customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Account"/>
        <xs:element ref="PIN" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
    <xs:attribute ref="xml:lang"/>
  </xs:element>

  <xs:element name="CustomerTable">
    <xs:complexType>
      <xs:all>
        <xs:element ref="Customer" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

The instance of the data model that describes the XML document associated with the schema is a sequence that contains the following eight element information items, in the same order as the numbered list below. When relevant to examples further in this document, the values for some of the properties are given for both cases where the instance of the data model had been constructed from the infoset, marked as: *(no schema)*, and where it has been been constructed from the PSVI, marked as: *(schema)*:

1. an element information item representing the `CustomerTable` element, with the following values for its properties
    - [namespace name]: http://example.org/customertable
    - [local name]: CustomersTable
    - [children]: a sequence of two element information items, which are the two `Customer` element information items in the data model instance, in the same order; that is, items 2 and 5 in the data model instance, in that order
    - [root]: the element information item itself
    - [attributes]: empty
    - [type name]:
        (no schema) empty
        (schema) http://example.org/customertable *"some locally unique identifier for the anonymous type of the `CustomerTable` element"*
    - [string value]: "John 111 Jane 222 222"
    - [typed value]: the element information item itself
    - [is-id]: false
    - [is-idrefs]: false
2. an element information item that describes the first `Customer` element in the document, with the following values for its properties
    - [namespace name]: http://example.org/customertable
    - [local name]: Customer
    - [children]: a sequence of two element information items, which are items 3 and 4 in the data model instance, describing the `Name` and `Account` sub-elements, respectively, in that same order
    - [root]: same value as the [root] property of the previous element information item
    - [attributes]: a attribute information item representing the `xml:lang` attribute. The attribute information item has the following values for its attributes:
        - [namespace name]: http://www.w3.org/XML/1998/namespace
        - [local name]: lang
        - [Attribute type]: empty
        - [owner element]: the previous element information item
        - [type name]:

(no schema) empty
(schema) http://www.w3.org/2001/XMLSchema language
- [string value]: "en"
- [typed value]: en
- [is-id]: false
- [is-idrefs]: false

- [type name]:
  (no schema) empty
  (schema) http://example.org/customertable *"some locally unique identifier for the anonymous type `Customer`"*
- [string value]: "John 111"
- [typed value]: the element information item itself
- [is-id]: false
- [is-idrefs]: false

3. an element information item representing the `Name` sub-element of the first `Customer` element in the document, with the following values for its properties
   - [namespace name]: http://example.org/customertable
   - [local name]: Name
   - [children]: a sequence of four character information items, spelling *j o h n*, in that order
   - [root]: same value as the [root] property of the previous element information item
   - [attributes]: empty
   - [type name]:
     (no schema) empty
     (schema) http://www.w3.org/2001/XMLSchema string
   - [string value]: "John"
   - [typed value]: "John"
   - [is-id]: false
   - [is-idrefs]: false
4. an element information item representing the `Account` sub-element of the first `Customer` element in the document, with the following values for its properties
   - [namespace name]: http://example.org/customertable
   - [local name]: Account
   - [children]: a sequence of three character information items, spelling *1 1 1*, in that order
   - [root]: same value as the [root] property of the previous element information item
   - [attributes]: empty
   - [type name]:
     (no schema) empty
     (schema) http://www.w3.org/2001/XMLSchema integer
   - [string value]: "111"
   - [typed value]: 111
   - [is-id]: false

    - [is-idrefs]: false
5. an element information item that describes the second `Customer` element in the document;
6. an element information item representing the `Name` sub-element of the second `Customer` element in the document;
7. an element information item representing the `Account` sub-element of the second `Customer` element in the document;
8. an element information item representing the `PIN` sub-element of the second `Customer` element in the document, with the following values for its properties
    - [namespace name]: http://example.org/customertable
    - [local name]: PIN
    - [children]: a sequence of three character information items, spelling *2 2 2*, in that order
    - [root]: same value as the [root] property of the previous element information item
    - [attributes]: empty
    - [type name]:
        (no schema) empty
        (schema) http://example.org/customertable PIN
    - [string value]: "222"
    - [typed value]: 222
    - [is-id]: false
    - [is-idrefs]: false

4 RIF combination with XML data

This section specifies the semantics of the combination of RIF documents and XML data, for RIF Core, RIF PRD and RIF BLD, where the XML data may be associated with an XML schema.

Definition (RIF+XML data combination). A ***RIF+XML data combination*** is a tuple $<R, D_1, ..., D_n>$, $n \geq 1$, where R is a RIF document and $D_1, ..., D_n$ are XML documents. □

One use case for the combination of RIF and XML data is when a RIF document imports explicitly the data to which the rules are to be applied: this is the case when the RIF document contains one or more `Import` directives where the `location` identifies explicitly an XML document to be combined with the importing RIF document. In that case, the combination is imposed by the producer of the RIF document, as well as the data to be combined with the rules that the RIF document contains. We call that case: *producer-side combination*, and the XML data to be combined with the RIF document: *imported XML data*.

However, a significant use case for RIF is rules being published or shared as a RIF document for consumers to apply them to their own data. In that case, the consumer of a RIF document decides independently of the producer to combine the rules contained in the RIF document with the data of his choice. We refer to that case as: *consumer-side combination*, and to the data that is combined with a RIF document as: *consumer-side data*.

This section specifies a normative semantics for the combination of a RIF document with XML data, without distinguishing between the two cases; that is, independently of whether the XML data to be combined with the RIF document is imported XML data or consumer-side XML data, or a combination of both. However, it provides a means to require that consumer-side XML data be valid with respect to an XML schema that is imposed by the producer of the RIF document.

Editor's Note: ...and thus to interchange, along with the rules, the data model with respect to which they have been designed. And thus, to combine RIF with any data model that can be represented in an XML Schema (including object models). Explanation, use case and example will be added in a future draft.

Definition (Associated XML schema). An XML document, D_i, is ***associated with an XML schema***, *XSD*, in the context of a RIF+XML data combination, $<R, D_1, ..., D_i, ..., D_n>$, $n \geq 1$, if and only if one of the following is true:

- The RIF document, *R*, contains an `Import` directive, where the `location` identifies the imported XML document D_i and the `profile` identifies *XSD*;
- The RIF document, *R*, contains an `Import` directive, where the `location` is missing, the `profile` identifies *XSD*, and D_i is a consumer-side XML document that validates against *XSD*.

□

4.1 Model-theoretic semantics of RIF BLD+XML data combinations

The model-theoretic semantics of RIF BLD defines a semantic structure as a tuple $I = <\boldsymbol{TV}, \boldsymbol{DTS}, \boldsymbol{D}, \boldsymbol{D}_{\text{ind}}, \boldsymbol{D}_{\text{func}}, \boldsymbol{I}_{\text{C}}, \boldsymbol{I}_{\text{V}}, \boldsymbol{I}_{\text{F}}, \boldsymbol{I}_{\text{NF}}, \boldsymbol{I}_{\text{list}}, \boldsymbol{I}_{\text{tail}}, \boldsymbol{I}_{\text{frame}}, \boldsymbol{I}_{\text{sub}}, \boldsymbol{I}_{\text{isa}}, \boldsymbol{I}_{=}, \boldsymbol{I}_{\text{external}}, \boldsymbol{I}_{\text{truth}}>$.

The specification of the model-theoretic semantics of a RIF BLD+XML data combination follows closely the specification of the semantics of a

RIF BLD document, except that the notion of semantic structures is replaced by the notion of *RIF BLD+XML data combined interpretations*: informally, RIF BLD+XML data combined interpretations are RIF BLD semantic structures, with additional conditions on some of the elements of ***I***.

The basic idea is that pieces of imported data are represented, for the purpose of combination with RIF, by information items in instances of the data model. Assertions about the names of XML elements in the imported data, and, where defined, their types, are represented using class membership and subclass formulas, in the importing RIF documents. This document specifies a subset of the lexical spaces of the symbol spaces `rif:iri` and `xs:NCName` and requires that constants whose literals are in that subset be interpreted as classes of element information items, associated to XML element names and types.

In the same way, assertions about the values of attributes and sub-elements of XML elements, in the imported data, are represented using frame formulas, in the importing RIF documents. This document specifies a subset of the lexical spaces of the symbol spaces `rif:iri` and `xs:NCName` and requires that constants whose literals are in that subset be interpreted as [attribute] and [children] properties of element information items.

Example 4.1. In a RIF document that imports the sample XML document from Section 3.6. Example of a data model instance, associated with the corresponding XML schema, the first rule, below, says that an *EarlyCustomer* is a *Customer* whose *Account* number is lower than 1000:

```
Forall ?x (_EarlyCustomer(?x) :-
           And( ?x # <http://example.org/customertable#Customer>
                Exists ?y (And ?x[<http://example.org/customertable#Account
                                  External(pred:numeric-less-or-equal(?y 1000)
```

Notice that, if the XML document were imported without the XML schema, the RIF consumer processing the combination would only have access to the string value of the `Account` sub-element, without an indication of its type. In that case, to guarantee that the rule behave as expected, the producer of the RIF document would have to add the information that the value of ?y must be cast into an integer before being compared as a number.

Another example, below, shows a rule that involves a combination with data that is represented as an attribute in the XML document.:

```
Forall ?x (_EnglishRec(?x) :-
           ?x[<http://www.w3.org/XML/1998/namespace#attribute(lang)> -> "en
```

That rule could be intended to mean that, if an item is represented, in the imported XML data, by an element with an attribute named `lang` in the XML namespace, and the value of that attribute is the `xs:language` constant en, then the information regarding that item is recorded in english.

□

As in the specification of RIF BLD, `Const` denotes the set of all constant symbols and `Var` denotes the set of all variable symbols.

4.1.1 Combined interpretation of RIF BLD non-document formulas and schemaless XML data

The simplest case is when a RIF BLD document is combined with XML data, without an associated XML schema: in that case, the instance of the data model that describes the XML data is built from the infoset, and it does not assign a type to the elements and attributes contained in the XML document.

To make the definition simpler, we will write that:

- a RIF constant, *c*, *string-matches* the [string value], *s*, of an information item, *i*, in an instance of the data model, if and only if c is a constant with type `xs:string` or a type derived from `xs:string` and $c = s$, after white space normalization;
- a RIF list, *l*, *string-matches* the [string value], *s*, of an information item, *i*, in an instance of the data model, if and only if $s = L$, after white space normalization, where *L* is the order-preserving concatenation of the elements of *l*, after flattening *l*, and with a white space added between each element.

Example 4.2.

- The RIF constants `"1"^^xs:string` and `" 1 "^^xs:string` (notice the white spaces) string-match the string value: "1", but the constants `"1"^^xs:integer` and `"1.0"^^xs:string` do not;
- The RIF lists `List("1"^^xs:string "little"^^xs:string "white"^^xs:string "dog"^^xs:string)`, `List("1 little"^^xs:string "white dog"^^xs:string)`, `List("1"^^xs:string List("little"^^xs:string "white"^^xs:string) "dog"^^xs:string)`, all, string-match the string value: "1 little white dog"; but none string-matches any of the string values: "little white dog" or "1.0 little white dog".

Definition (RIF BLD+schemaless XML data combined interpretation). A ***RIF BLD+schemaless XML data combined interpretation*** is a pair ($\{I_{DM}\}$, I), where $\{I_{DM}\}$ is a set of element information items constructed from an infoset and where the references have been resolved, and I is a semantic structure, as defined in the semantics of RIF BLD, that satisfies the following additional conditions:

1. $\{I_{DM}\} \subseteq D_{Ind}$;
2. For all frame formulas `o [ slot -> v ]`, where $I(o) = e \in \{I_{DM}\}$, $I_{truth}(I($`>o [ slot -> v ]`$)) =$ **t** (true) if one of the following is true:
    a. $I($`slot`$) = I($`"`*NAMESPACE*`#attribute(`*NAME*`)"^^rif:iri`), or $I($`slot`$) = I($`"attribute(`*NAME*`)"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the values of the [namespace name] and [local name] properties of an attribute information item, *a*, in the [attributes] property of *e*, and v string-matches the [string value] of *a*;
    b. or $I($`slot`$) = I($`"`*NAMESPACE*`#`*NAME*`"^^rif:iri`), or $I($`slot`$) = I($`"`*NAME*`"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the values of the[namespace name] and [local name] properties of an element information item, *c*, in the [children] property of *e*, and either
        - the [typed value] of *c* is *c* itself and $I(v) = c$;
        - or the [typed value] of *c* is not *c*, and v string-matches the [string value] of *c*;
    c. or $I($`slot`$) = I($`"list(`*NAME*`)"^^rif:iri`), or $I($`slot`$) = I($`"list(`*NAME*`)"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the [namespace name] and [local name] of at least one element information item in the [children] property of *e*, and *v* is a RIF List, and *v* has the same length as the sub-list, *c*, of the [children] of *e*, that contains only the element information items whose [namespace name] and [local name] property match, respectively, *NAMESPACE* and *NAME*, in document order, and for each pair of elements of the same rank, v_i in *v* and c_i in *c*,
        - either the [typed value] of c_i is c_i itself and $I(v_i) = c_i$;
        - or the [typed value] of c_i is not c_i, and v_i string-matches the [string value] of c_i;
3. For all class membership formulas `o # C`, where $I(o) = e \in \{I_{DM}\}$, I_{truth}(`o # C`) = **t** (true) if
    a. $I($`C`$) = I($`"`*NAMESPACE*`#`*NAME*`"^^rif:iri`), or $I($`C`$) = I($`"`*NAME*`"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the values of the [namespace name] and [local name] properties of *e*.

□

Notice that `xs:NCName` constants are included in the definition to allow the interpretation of RIF+XML data combinations where some or all elements or attributes in the XML data are not in a namespace. As defined in [RIF-DTB], the lexical space of `rif:iri` consists of all absolute IRIs as specified in [RFC-3987]; as a consequence, it does not allow the representations of local names without namespaces.

Editor's Note: The definition will be further commented and explained in a future draft.

The truth valuation of a well-formed RIF BLD formula, φ, under a RIF BLD+schemaless XML data combined interpretation ($\{I_{DM}\}$, I) is determined by I_{truth} exactly as specified for the interpretation of RIF BLD non-document formulas (but with the definition of I_{truth} as modified by the combination with XML data).

Example 4.3. With respect to a RIF BLD+XML data combination, $<D, I_{DM}>$, where a RIF BLD document, D, imports the sample XML document from Section 3.6. Example of a data model instance, above, without associating it with any XML schema, and under the semantics for RIF BLD+schemaless XML data combinations as specified above, the following facts must be true in all the interpretations where the specified conditions hold: that is, for each fact, f, $\boldsymbol{I}_{truth}(f) = \mathbf{t}$ in all interpretations, $\boldsymbol{I}$, where the specified conditions hold. The facts may be true in other interpretations as well, but not as a consequence of the combination (NB: the examples use RIF non-normative presentation syntax, and, in particular, the shortcut presentation syntax for RIF constants, as described in RIF Data Types and Builtins, Section 2.2.2. Shortcuts for constants in RIF's presentation syntax; see Appendix D: Examples using the RIF/XML normative syntax):

- `_Customer_John [<http://www.w3.org/XML/1998/namespace#attribute(lang)> -> "en"^^xs:language]` must be true in all interpretations, $\boldsymbol{I}$, where $\boldsymbol{I}_C$ maps `_Customer_John` on the second element in I_{DM} (representing the first element `Customer` in the sample XML data);
- `_Customer_John [<http://www.w3.org/XML/1998/namespace#attribute(lang)> -> "en"])` must also be true in all these interpretations, since the [string value] of the `xml:lang` attribute matches the literal of the `xs:string` constant `"en"`;
- `_Customer_John [ <http://example.org/customertable#Name> -> "John" ]` is required to be true in all interpretations where $\boldsymbol{I}_C$ maps `_Customer_John` on the second element in I_{DM};
- `_Customer_John [<http://example.org/customertable#Account> -> 111 ]` is not required to be true, even in interpretations where $\boldsymbol{I}_C$ maps `_Customer_John` on the

second element in I_{DM}: indeed, the [string value] of the sub-element `Account` of that element is the string: *111*, whereas the slot value, in the example frame, is the `xs:integer` *111*, and only an `xs:string` constant can string-match a string value;

- `_CustomerTable [<http://example.org/customertable#list(Customer)> -> List(_Customer_John, _Customer_Jane) ]` in all interpretations where I_C maps `_CustomerTable` on the root element of I_{DM}, `_Customer_John` on the second element, and `_Customer_Jane` on the fifth element (representing the second element `Customer` in the sample XML data, whose `Name` sub-element contains the sequence of characters: *Jane*);
- `_Customer_John # <http://example.org/customertable#Customer>`, in all the interpretations where I_C maps `_Customer_John` on the second or fifth element in I_{DM};
- `_Name_John # <http://example.org/customertable#Name>`, in all the interpretations where I_C maps `_Name_John` on the third or sixth element in I_{DM};
- `"John" # <http://example.org/customertable#Name>` is not required to be true in any interpretation, because, per the semantics of the `xs:string` data type in RIF, the `xs:string` constant `"John"` cannot be mapped on an element information item in I_{DM};
- `_Customer_John = <myNamespace#ID0001>` in all the interpretations where I_C(`_Customer_John`) = I_C(`<myNamespace#ID0001>`) = $e \in I_{DM}$ (e.g. the second element, representing the `Customer` with `Name`: "John").

Notice that, if our example XML data was not in a namespace, the semantics of RIF+XML data combinations would require that the following facts be true, all other required conditions being satisfied:

- `_Customer_John [ "Name" -> "John" ]`
- `_CustomerTable [ list("Customer") -> List(_Customer_John, _Customer_Jane) ]`
- `_Customer_John # "Customer"`
- `_Name_John # "Name"`

Example 4.4. Consider the following element with mixed-content, and assume that the element is contained in a data source that is imported in a RIF document, without a namespace nor an associated XML schema:

```
<letterBody>
  Thank you for ordering the <item>widget</item>.
  It should arrive by <arrivalDate>09-09-09</arrivalDate>
</letterBody>
```

I_{truth} must, under the semantics specified above, map the fact:
`_myLetter["letterBody" -> "Thank you for ordering the Widget. It should arrive by 09-09-09"],`

to true in any interpretation of the RIF+XML data combination that maps the RIF local constant `_myLetter` to an element information item that describes, in the data model instance, the parent element of the above letterBody element.

4.1.2 Combined interpretation of RIF BLD non-document formulas and schema valid XML data

In the case where a RIF BLD document is combined with XML data that is associated with an XML schema, the instance of the data model that describes the imported XML data is built from the PSVI, and it does ascribe a type to the elements and attributes contained in the XML document.

The semantics of these combinations are, essentially, the same as in the schemaless case, except that values are compared using their typed values instead of their string values, and additional conditions are imposed on I_{truth} to take into account information that is specific to the PSVI.

Accordingly, and to make the definition simpler, we will write that:

- a RIF constant, *c*, *type-matches* the [typed value] of an information item, *i*, in an instance of the data model, if and only if the type of *c*, T_c, and the type that is identified by the expanded QName in the [type name] property of *i*, T_i, have a non-empty intersection, and *c* and the [typed value] of *i* are the same value in the intersection of T_c and T_i;
- a RIF list, *l*, *type-matches* the [typed value] of an information item, *i*, in an instance of the data model, if and only if the type of *i* is a list type and the elements of *l* type-match, one-to-one and in document order, the atomic values in the [typed value] of *i*, after flattening *l*.

In addition we will write, in the following definition, that two strings, a namespace and an NCName, *match, possibly modulo a substitution,* the values of, respectively, the [namespace name] and [local name] of an element information item, *e*, if

- either the namespace and NCName are the values of, respectively, the [namespace name] and [local name] of *e*,
- or *e* describes an instance *E* of an XML element that belongs to a substitution group, occurs in a position where the schema requires the head element, and the namespace and NCName are the values of, respectively, the [namespace name] and [local name] of the head element.

This specification does not prescribe any semantics for a RIF+XML data combination, when the XML data is associated with an XML schema with respect to which it is not valid.

Definition (RIF BLD+schema valid XML data combined interpretation). A ***RIF BLD+schema valid XML data combined interpretation*** is a pair $(\{I_{DM}\}, I)$, where $\{I_{DM}\}$ is a set of element information items constructed from one or more PSVIs, using one or more XML schemas, XSD_i, and where the references have been resolved, and I is a semantic structure, as defined in the semantics of RIF BLD, that satisfies the following additional conditions:

1. $\{I_{DM}\} \subseteq D_{Ind}$;
2. For all frame formulas `o [ slot -> v ]`, where $I(o) = e \in \{I_{DM}\}$, I_{truth}(`o [ slot -> v ]`) = **t** (true) if one of the following is true:
    a. I(`slot`) = I(`"`*NAMESPACE*`#attribute(`*NAME*`)"^^rif:iri`), or I(`slot`) = I(`"attribute(`*NAME*`)"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the values of the [namespace name] and [local name] properties of an attribute information item, *a*, in the [attributes] property of *e*, and v type-matches the [typed value] of *a*;
    b. or I(`slot`) = I(`"`*NAMESPACE*`#`*NAME*`"^^rif:iri`), or I(`slot`) = I(`"`*NAME*`"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* match, possibly modulo a substitution, the values of, respectively, the [namespace name] and [local name] properties of an element information item, *c*, in the [children] property of *e*, and either
        - the [typed value] of *c* is *c* itself and $I(v) = c$;
        - or the [typed value] of *c* is not *c*, and v type-matches the [typed value] of *c*;
    c. or I(`slot`) = I(`"`*NAMESPACE*`#list(`*NAME*`)"^^rif:iri`), or I(`slot`) = I(`"list(`*NAME*`)"^^xs:NCName`) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* match, possibly modulo a substitution, the values of, respectively, the [namespace name] and [local name] properties of at least one element information item in the [children] property of *e*, and *v* is a RIF List, and *v* has the same length as the sub-list, *c*, of the [children] of *e*, that contains only the element information items whose [namespace name] and [local name] properties are matched, possibly modulo a substitution, by, respectively, *NAMESPACE* and *NAME*, in document order, and for each pair of elements of the same rank, v_i in *v* and c_i in *c*,
        - either the [typed value] of c_i is c_i itself and $I(v_i) = c_i$;

- or the [typed value] of c_i is not c_i, and v_i type-matches the [typed value] of c_i;

3. For all class membership formulas o # C, where $I(o) = e \in \{I_{DM}\}$, I_{truth}(o # C) = **t** (true) if one of the following is true
   a. I(C) = I("*NAMESPACE*#*NAME*"^^rif:iri), or I(C) = I("*NAME*"^^xs:NCName) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are the values, respectively, of the [namespace name] and the [local name] properties of *e*;
   b. or I(C) = I("*NAMESPACE*#type(*NAME*)"^^rif:iri), or I(C) = I("type(*NAME*)"^^xs:NCName) and *NAMESPACE* is empty, where *NAMESPACE* and *NAME* are, respectively, the namespace and the local name in *e* 's [type name] property;
4. ***DTS*** is extended to include all the simple types, whether built-in XML schema or defined in one of the XSD_i, that are mentioned in the [type name] property of any of the information items in the data model instance.
5. For all subclass formulas sub ## SUP, where sub and SUP are distinct constants of type rif:iri or xs:string, or a type derived from rif:iri or xs:string, I_{truth}(sub ## SUP) = **t** (true) if one of the following is true:
   a. I(sub) = I("$NAMESPACE_{sub}$#type($NAME_{sub}$)"^^rif:iri), or I(C) = I("type($NAME_{sub}$)"^^xs:NCName) and $NAMESPACE_{sub}$ is empty, where $NAMESPACE_{sub}$ and $NAME_{sub}$ are, respectively, the values of the target namespace and name of an element, e_{sub} defined in *XSD*, one of the XSD_i *and*
      - either I(SUP) = I("$NAMESPACE_{SUP}$#$NAME_{SUP}$"^^rif:iri), or I(C) = I("$NAME_{SUP}$"^^xs:NCName) and $NAMESPACE_{SUP}$ is empty, where $NAMESPACE_{SUP}$ and $NAME_{SUP}$ are, respectively, the values of the target namespace and name of an element, e_{SUP} defined in *XSD*, and e_{sub} is defined in a substitution group where the head element is e_{SUP};
      - or I(SUP) = I("$NAMESPACE_{SUP}$#type($NAME_{SUP}$)"^^rif:iri), or I(C) = I("type($NAME_{SUP}$)"^^xs:NCName) and $NAMESPACE_{SUP}$ is empty, where $NAMESPACE_{SUP}$ and $NAME_{SUP}$ are, respectively, the values of the target namespace and name of an XML schema type, T_{SUP}, defined in *XSD* (not including built-in XML schema data types), and T_{SUP} is the type declared in the definition of e_{sub}, in *XSD*;
   b. or I(sub) = I("$NAMESPACE_{sub}$#type($NAME_{sub}$)"^^rif:iri), or I(C) = I("type($NAME_{sub}$)"^^xs:NCName) and $NAMESPACE_{sub}$ is empty, and I(SUP) = I("$NAMESPACE_{SUP}$#type($NAME_{SUP}$)"^^rif:iri), or I(C) = I("type($NAME_{SUP}$)"^^xs:NCName) and $NAMESPACE_{SUP}$ is

empty, where $NAMESPACE_{sub}$ and $NAME_{sub}$ are, respectively, the values of the target namespace and name of and XML schema type, T_{sub}, defined in XSD (not including built-in XML schema data types), and $NAMESPACE_{SUP}$ and $NAME_{SUP}$ are, respectively, the values of the target namespace and name of an XML schema type, T_{SUP}, defined in *XSD* (not including built-in XML schema data types), and T_{sub} is derived by restriction or by extension from T_{SUP}.

□

Editor's Note: The case defined by the first bullet under 2.b and 2.c could be further refined, to differentiate between the case of an empty element and the case of the null value: in the latter case, the schema must define a nillable content for the element. Shall we leave the interpretation to the implementations or shall we include it in that spec? If the latter, we must include a `rif:NULL` constant in every symbol space...

Editor's Note: Condition 4 will be further refined in a future draft, to account for the problem posed by the inclusion of some data types in a RIF interpretation, e.g. xs:duration.

Editor's Note: The definition will be further refined in a future draft to account for cases such as `xs:anyType` typed elements.

Editor's Note: The definition will be further commented and explained in a future draft.

The truth valuation of a well-formed RIF BLD formula, φ, under a RIF BLD+schema valid XML data combined interpretation $(\{I_{DM}\}, I)$ is determined by I_{truth} exactly as specified for the interpretation of RIF BLD non-document formulas (but with the definition of I_{truth} as modified by the combination with XML data).

Example 4.5. Following up on Example 4.3, above, but assuming that the RIF BLD document is combined with the sample XML data, associated with the XML schema, as in Section 3.6. Example of a data model instance, the following hold with respect to the interpretation of sample facts, under the semantics of the RIF BLD+schema valid XML data combination:

- `_Customer_John [<http://www.w3.org/XML/1998/namespace#attribute(lang)> -> "en"^^xs:language]` must be true in all interpretations, $\boldsymbol{I}$, where $\boldsymbol{I}_C$ maps `_Customer_John` on the second element in I_{DM} (representing the first element `Customer` in the sample XML data);

- but _Customer_John [<http://www.w3.org/XML/1998/namespace#attribute(lang)> -> "en"]) must not, since the xs:string constant "en" does not type-match the xs:language [typed value]: en;
- _Customer_John [<http://example.org/customertable#Account> -> 111] must be true in all interpretations where I_C maps _Customer_John on the second element in I_{DM} (but, e.g. _Customer_John [<http://example.org/customertable#Account> -> "111"] must not);
- _PIN_Jane # <http://example.org/customertable#type(PIN)> must be true in all interpretations where I_C maps _PIN_Jane on the eighth (and last) element in I_{DM}

Notice that _PIN_Jane # <http://www.w3.org/2001/XMLSchema#type(integer)> is never required to be true in any interpretation of the example RIF BLD+schema valid XML data combination, even in interpretations where I_C maps _PIN_Jane on the last element in I_{DM}, although the XML schema type http://example.org/customertable#PIN is derived by restriction from xs:integer. That is because xs:insteger is not defined in the associated XML schema (*"the namespace of* C *and* NAME [must] *match, respectively, the target namespace and the name of an XML schema type defined in XSD (not including built-in XML schema data types)"*, condition 4.b in the definition of RIF BLD+schema valid XML data combined interpretations).

4.1.3 Combined interpretation of RIF BLD documents and XML data

Editor's Note: The definition of the combined interpretation of non-document RIF BLD formulas and XML data, where some of the XML data is associated with an XML schema and some is not, follows directly from the definitions of the schemaless and schema valid cases. It will be developed explicitly in a future draft.

A RIF+XML data combination $<R, D_1, ..., D_n>$, $n \geq 0$, where R is an RIF BLD document, and the D_i, $1 \leq i \leq n$, are all the XML documents that are, either, imported, directly or indirectly, by R, or that contain consumer-side XML data, is interpreted using a RIF BLD+XML data combined interpretation $(\{I_{DM}\}, \hat{I})$, where:

- $\{I_{DM}\} = \cup_{1...n} \{I_{DM_i}\}$, the union of all the sets, $\{I_{DM_i}\}$, $1 \leq i \leq n$, of all the element information items in the instances of the data model that represent, each, one of the XML documents in $(D_1, ..., D_n)$;
- $\hat{I}$ is a semantic multi-structure that is defined exactly as specified for the interpretation of RIF BLD document formulas, except that the I_{truth} component in each semantic structure, I, contained in $\hat{I}$,

is subject to the additional conditions required for the combined interpretation ($\{I_{DM}\}$, I).

The truth valuation of a RIF+XML data combination $<R, D_1, ..., D_n>$, $n \geq 0$, is determined exactly as specified for the truth valuation of a RIF BLD document formulas, using the RIF BLD+XML data combined interpretation ($\{I_{DM}\}$, $\hat{I}$) in place of a semantic multi-structure.

In the same way, the notions of model and of logical entailment are defined for a RIF BLD+XML data combined interpretation, using exactly the same definition as specified for RIF BOLD (document and non-document) formulas, where the semantic multi-structure, $\hat{I}$, is replaced by the RIF BLD+XML data combined interpretation ($\{I_{DM}\}$, $\hat{I}$).

Notice that, in the case where no XML data is imported, that is, if $n = 0$, $\{I_{DM}\}$ is empty, and the interpretation of a RIF BLD document under a RIF BLD+XML data combined interpretation ($\varnothing$, $\hat{I}$) is strictly equivalent to its interpretation under the standard RIF BLD semantics.

4.2 Operational semantics of RIF PRD+XML data combinations

The effect of the combination with XML data, with or without an associated XML schema, on the operational semantics of a RIF PRD document, is that a ground fact, f, must be added to the set of ground facts that is associated with the state of the fact base that is to be combined with the XML data, if it is true under a RIF BLD+XML data combined interpretation, as defined in the previous section (schemaless or schema valid case, depending on whether the XML data is associated with an associated schema or not).

See also the (non-normative) appendix on embedding XML data as RIF facts for an exhaustive description of the facts to be added to the initial state of the fact base that is to be combined with the XML data.

Editor's Note: The definition will be further commented and explained in a future draft.

Example.

Editor's Note: Examples will be added in a future draft.

4.3 Semantics of RIF Core+XML data combinations

RIF Core is a syntactic subset of RIF BLD, and the semantics of RIF Core+XML data combinations is identical to the semantics of RIF BLD+XML data combinations for that subset. RIF Core is also a syntactic

subset of RIF PRD, and the semantics of RIF Core+XML data combinations is also identical to the semantics of RIF PRD+XML data combinations for that subset.

Notice, in particular, that the condition on the interpretation of subclass formulas, condition 5 in the definition of a RIF BLD+schema valid XML data combined interpretation, permits the simplification of the condition on the interpretation of membership formulas (condition 3). Since subclass formulas are not defined in RIF Core, the condition on the interpretation of membership formulas (condition 3), in the definition of a RIF Core+schema valid XML data combined interpretation, must be extended to account explicitly for all the membership formulas whose interpretation follows, in the RIF BLD+schema valid XML data case, from the axioms on the truth valuation of subclass and membership formulas.

5 Conformance

Editor's Note: This section will be completed in a future draft.

6 The special case of RDF and OWL data sources

Editor's Note: RDF and OWL data sources can be imported in RIF documents in two different ways: according to the RIF-RDF-OWL specification, that specifies the combination of RIF documents with RDF and OWL graphs, directly or as an XML document. This section examines how these two ways of importing RDF and OWL documents relate. The section will be completed in a future draft.

7 References

[Infoset]
XML Information Set (Second Edition), John Cowan and Richard Tobin, Editors. World Wide Web Consortium, 04 Feb 2004. This version is http://www.w3.org/TR/2004/REC-xml-infoset-20040204. The latest version is available at http://www.w3.org/TR/xml-infoset.

[RFC 3986]
RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, and L. Masinter. January 2005. Available at: http://www.ietf.org/rfc/rfc3986.txt

[RFC 3987]
RFC 3987: Internationalized Resource Identifiers (IRIs), M. Duerst and M. Suignard. January 2005. Available at: http://www.ietf.org/rfc/rfc3987.txt

[RIF-Core]
RIF Core Dialect Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, Dave Reynolds, eds. W3C Recommendation, 22 June 2010, http://www.w3.org/TR/2010/REC-rif-core-20100622/. Latest version available at http://www.w3.org/TR/rif-core/.

[RIF-BLD]
RIF Basic Logic Dialect Harold Boley, Michael Kifer, eds. W3C Recommendation, 22 June 2010, http://www.w3.org/TR/2010/REC-rif-bld-20100622/. Latest version available at http://www.w3.org/TR/rif-bld/.

[RIF-PRD]
RIF Production Rule Dialect Christian de Sainte Marie, Gary Hallmark, Adrian Paschke, eds. W3C Recommendation, 22 June 2010, http://www.w3.org/TR/2010/REC-rif-prd-20100622/. Latest version available at http://www.w3.org/TR/rif-prd/.

[RIF-RDF-OWL]
RIF RDF and OWL Compatibility Jos de Bruijn, editor. W3C Recommendation, 22 June 2010, http://www.w3.org/TR/2010/REC-rif-rdf-owl-20100622/. Latest version available at http://www.w3.org/TR/rif-rdf-owl/.

[XDM]
XQuery 1.0 and XPath 2.0 Data Model (XDM), M. Fernández, A. Malhotra, J. Marsh, M. Nagy, N. Walsh, Editors. W3C Recommendation 23 January 2007. This version is http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/. The latest version is available at http://www.w3.org/TR/xpath-datamodel/.

[XML]
Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, Editor. W3C Recommendation 26 November 2008. This version is http://www.w3.org/TR/2005/REC-xml-id-20050909/. The latest version is available at http://www.w3.org/TR/REC-xml.

[xml:id]
xml:id Version 1.0, J. Marsh, D. Veillard, N. Walsh, Editors. W3C Recommendation 9 September 2005. This version is http://www.w3.org/TR/2008/REC-xml-20081126/. The latest version is available at http://www.w3.org/TR/xml-id.

[XPath 2.0]
XML Path Language (XPath) 2.0, D. Chamberlin , A. Berglund, S. Boag, et. al., Editors. W3C Recommendation 23 Jan 2007. This version is http://www.w3.org/TR/2007/REC-xpath20-20070123/. The latest version is available at http://www.w3.org/TR/xpath20/.

[XQuery 1.0]
XQuery 1.0: An XML Query Language, D. Chamberlin , A. Berglund, S. Boag, et. al., Editors. W3C Recommendation 23 Jan 2007. This version is http://www.w3.org/TR/2007/REC-xquery-20070123/. The latest version is available at http://www.w3.org/TR/xquery/.

[XSD 1.1 Part 2]
W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, D. Peterson, S. Gao, A. Malhotra, C. M. Sperberg-McQueen, H. S. Thompson, Editors. W3C Candidate Recommendation 30 April 2009. This version is http://www.w3.org/TR/2009/CR-xmlschema11-2-20090430/. The latest version is available at http://www.w3.org/TR/xmlschema11-2/.

8 Appendix A: Glossary (non-normative)

Editor's Note: This section will be completed in a future draft.

Expanded QName
An expanded QName is a set of three values consisting of a possibly empty prefix, a possibly empty namespace URI, and a local name. (http://www.w3.org/TR/xpath-datamodel/#dt-expanded-qname)

QName
A qualified name is a name subject to namespace interpretation. In documents conforming to this specification, element and attribute names appear as qualified names. Syntactically, they are either prefixed names or unprefixed names. An attribute-based declaration syntax is provided to bind prefixes to namespace names and to bind a default namespace that applies to unprefixed element names; these declarations are scoped by the elements on which they appear so that different bindings may apply in different parts of a document. (http://www.w3.org/TR/REC-xml-names/#dt-qualname)

9 Appendix B: Extract from the XQuery 1.0 and XPath 2.0 Data Model (non-normative)

9.1 Element and attribute node type names (from [XDM])

Editor's Note: This section reproduces the text of Section 3.3.1.1. Element and Attribute Node Type Names in [XDM], for the reader's convenience. Notice that, for the purpose of this specification, the type name is considered unknown, and the [type name] property is left empty, when the [validity] property does not exist or is "unknown" and the [validation attempted] property does not exist or is "none".

Notation: Some aspects of type assignment rely on the ability to access properties of the schema components. Such properties are indicated by the style {component property}. Note that this does not mean a lightweight schema processor cannot be used, it only means that the application must have some mechanism to access the necessary properties.

The precise definition of the schema type of an element or attribute information item depends on the properties of the PSVI. In the PSVI, [Schema Part 1] defines a [type definition] property as well as the [type definition namespace], [type definition name] and [type definition anonymous] properties, which are effectively short-cut terms for properties of the type definition. Further, the [element declaration] and [attribute declaration] properties are defined for elements and attributes, respectively. These declarations in turn will identify the [type definition] declared for the element or attribute. To distinguish the [type definition] given in the PSVI for the element or attribute instance from the [type definition] associated with the declaration, the former is referred to below as the actual type and the latter as the declared type of the element or attribute instance in question.

The type depends on the declared type, the actual type, and the [validity] and [validation attempted] properties in the PSVI. If:

- The [validity] and [validation attempted] properties exist and have the values "valid" and "full", respectively, the schema type of an element or attribute information item is represented by an expanded-QName whose namespace and local name correspond to the first applicable items in the following list:
    - If the declared type exists and is a union and the actual type is (not the same as the declared type, and not a type derived from the declared type, but) one of the member types of the union, or derived from one of its member types:
        - If the {name} property of the declared type is present: the {target namespace} and {name} properties of the declared type.
        - If the {name} property of the declared type is absent: the namespace and local name of the anonymous type name supplied for the declared type.
    - If there is no declared type, and the actual type is a union, then:
        - If the {name} property of the actual type is present: the {target namespace} and {name} properties of the actual type.
        - If the {name} property of the actual type is absent: the namespace and local name of the

anonymous type name supplied for the actual type.
    - Otherwise:
        - If [type definition anonymous] is false: the {target namespace} and {name} properties of the actual type.
        - If [type definition anonymous] is true: the namespace and local name of the anonymous type name supplied for the actual type.
- The [validity] property exists and is "invalid", or the [validation attempted] property exists and is "partial", the schema type of an element is xs:anyType and the type of an attribute is xs:anySimpleType.
- The [validity] property exists and is "notKnown", and the [validation attempted] property exists and is "none", the schema type of an element is xs:untyped and the type of an attribute is xs:untypedAtomic.
- The [validity] or [validation attempted] properties do not exist, the schema type of an element is xs:untyped and the type of an attribute is xs:untypedAtomic.

The prefix associated with the type names is implementation-dependent.

9.2 Typed value determination (from [XDM])

This section describes how the typed value of an Element or Attribute Node is computed from an element or attribute PSVI information item, where the information item has either a simple type or a complex type with simple content. [...]

The typed value of Attribute Nodes and some Element Nodes is a sequence of atomic values. The types of the items in the typed value of a node may differ from the type of the node itself. This section describes how the typed value of a node is derived from the properties of an information item in a PSVI.

The types of the items in the typed value of a node are determined as follows. The process begins with T, the schema type of the node itself, as represented in the PSVI. For each primitive or ordinary simple type T, the W3C XML Schema specification defines a function M mapping the lexical representation of a value onto the value itself.

Note. For atomic and list types, the mapping is the “lexical mapping” defined for T in [Schema Part 2]; for union types, the mapping is the lexical mapping defined in [Schema Part 2] modified as appropriate by any applicable rules in [Schema Part 1]. The mapping, so modified, is a function (in the mathematical sense) which maps to a single value even in cases where the lexical mapping properly maps to multiple values.

The typed value is determined as follows:

- If the nilled property of the node in question is true, then the typed value is the empty sequence.
- If T is xs:anySimpleType or xs:anyAtomicType, the typed value is the [schema normalized value] as an instance of xs:untypedAtomic.
- Otherwise, the typed value is the result of applying M to the string value as an instance of the appropriate value type, where the appropriate value type is the [member type definition] if T is a union type, otherwise it is simply T.

The typed value determination process is guaranteed to result in a sequence of atomic values, each having a well-defined atomic type. This sequence of atomic values, in turn, determines the typed-value property of the node in the data model.

10 Appendix C: Embedding imported data sources as RIF facts (non-normative)

Editor's Note: This section will be completed in a future draft.

11 Appendix D: Examples using the normative RIF/XML syntax

Editor's Note: This section will be completed in a future draft

12 Appendix E: Change Log (non-normative)

- Complete rework of the semantics