W3C Editor's Draft

RIF Combination with XML data

W3C Editor's Draft 4 September 2009

This version:
http://www.w3.org/2005/rules/wg/draft/ED-rif-xml-data-20090904/
Latest editor's draft:
http://www.w3.org/2005/rules/wg/draft/rif-xml-data/
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 sources.

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May Be Superseded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 10 documents:

1. RIF Core Dialect
2. RIF Basic Logic Dialect
3. RIF Framework for Logic Dialects
4. RIF RDF and OWL Compatibility

W3C Editor's Draft

5. RIF Datatypes and Built-Ins 1.0
6. RIF Production Rule Dialect
7. RIF Use Cases and Requirements
8. RIF Test Cases
9. RIF Combination with XML data (this document)
10. OWL 2 RL in RIF

First Public Working Draft

@@@UPDATE

@@@ Include-in-this-round?

Please Comment By 23 October 2009

The Rule Interchange Format (RIF) Working Group seeks public feedback on this First Public 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.

No Endorsement

Publication as a Editor's 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. The group does not expect this document to become a W3C Recommendation. 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

Overview

This document specifies how a RIF document can be combined with XML data sources. It specifies a partial implementation of the XQuery 1.0 and XPath 2.0 Data Model [XDM] that uses a subset of the RIF condition language.

The XQuery 1.0 and XPath 2.0 Data Model 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 how a subset of the RIF condition language can be interpreted with respect to an XML data source, and what is the associated semantics, in accordance with the XQuery 1.0 and XPath 2.0 Data Model.

Essentially, this document specifies:

- the representation as classes, in RIF, of sets of elements nodes with given name or type properties, in an XML instance document;
- the representation, in RIF, of the relation between parent element nodes and children element and attribute nodes, when the children nodes have given name and type properties.

The XQuery 1.0 and XPath 2.0 Data Model 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.

Followingly, this document specifies how XML data sources and, possible, the corresponding XML schemas are combined with a RIF document, using the `Import` construct from RIF.

However, an instance of the data model can also be constructed from non-XML sources such as relational tables in a database or object instances. In this case, the relevant parts of the data model are represented in RIF in conformance with the

serialization of the source according to an XML schema that MUST be specified in the RIF document.

An XML schema can also be combined with a RIF document without without the associated data source being specified: in that case, the selection of the data source to be combined with the RIF document is left to the RIF document consumer. 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.

TBC

Typical scenarios etc

Data sources and schemas

Importing data sources 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.

This document specifies a new allowed value and a new set of allowable values for the profile of an import:

- the new value is the URI: `rif:xml`;
- in addition, any URI that identifies an XML schema document is allowed as a profile.

The `Import` directive is further modified to make the IRI of the data document optional as well.

The BNF-style pseudo-schema for the modified syntax is as follows:

```
<Import>
   <location> xs:anyURI </location>?
   <profile> xs:anyURI </profile>?
</Import>
```

Although all the sub-elements are optional, an `Import` directive must contain at least a data document `location` or a `profile`. If an `Import` directive contains no data document `location`, the `profile` must identify an XML Schema.

If location and schema, and XML doc at location provides a schema-location URI, the schema URI in the profile must identify the same schema as the schema-location URI

In addition, if the profile is `rif:xml`, the `location` URI must be present and it must identify an XML instance document.

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

Available documents

This is a mapping of strings onto document nodes. The string represent the absolute URI of a resource. The document node is the root of a tree that represents the resource using using the XQuery 1.0 and XPath 2.0 Data Model.

In RIF, the available documents in the dynamic context, when the rules retrieved from a RIF document are used in combination with data sources, is defined as follows:

- if the RIF document contains `Import` directives where the profile is `rif:xml`, the `location` URI in each of these directives is mapped onto the document node that is the root of the tree that represent the XML instance document identified by that `location` URI using the data model, where
  - the data model is constructed directly if the XML instance document does not declare a `schema-location` URI;
  - or the data model is constructed from the post-schema validation infoset (PSVI), if the XML instance document declares `schema-location` URI;
- if the RIF document contains `Import` directives that locate a data source document and where the profile is an URI that identifies an XML schema, the `location` URI in each of these directives is mapped onto the document node that is the root of the tree that represent the XML instance document identified by that `location` URI using the data model, where the data model is constructed from the post-schema validation infoset (PSVI) created using the XML schema identified by the profile;

Editor's Note: The data model supports incompletely validated documents. Elements and attributes that are not valid are treated as having unkonw types. See [Section 3.3 Construction from a PSVI[XDM]].

Editor's Note: If an imported data source document is not an XML instance document, its representation using the data model is built from the valid XML serialization of the data source according to the XML schema that is identified in the profile.

- in addition, the URI of any data source that is dynamically associated at run time with the rules retrieved from the RIF document, is mapped onto the document node that is the root of the tree that represents the dynamically associated data source using the data model

- if the RIF document contains an `Import` directive where the location of the data source document is missing and the profile is an URI that identifies an XML schema, the data model is constructed from the post-schema validation infoset (PSVI) created using that XML schema;
- else, the data model is constructed directly.

Editor's Note: If a dynamically associated data source is not an XML instance document, its representation using the data model is built from the valid XML serialization of the data source according to the XML schema that is identified in the profile. If the RIF document does not contains an `Import` directive that identifies an XML schema in the profile, without associating it to an imported data source document with a the `location`, the direct construction of the data model is implementation dependent, but stable (that is, the same document produces always the same data model instance). There are no constraint on how an instance of the data model may be constructed directly, save that the resulting instance must satisfy all of the constraints described in the XQuery 1.0 and XPath 2.0 Data Model specification that are relevant to the partial implementation of the specification in RIF.

Available collections

The default available collection in the dynamic context, as returned by a call to `fn:collection` with no argument (see [Section 15.5.8[XFO]]), is the sequence of the nodes of type *element* in the set of the trees that represent the available data sources documents using the XQuery 1.0 and XPath 2.0 Data Model, as specified in the previous section.

This document does not specify the function `fn:collection` further.

RIF as a partial implementation of XDM

This document specifies, for some constructs of the RIF condition language, an interpretation with respect to the available data source documents represented using the XQuery 1.0 and XPath 2.0 Data Model, as described above.

The specification provides, therefore, a mean to combine RIF documents with XML data sources, with or without an associated XML schema, as well as a mechanism to associate with a RIF document, the data model intended for the data sources as an XML schema.

This makes RIF an implementation of the XQuery 1.0 and XPath 2.0 Data Model. However, RIF is only a very partial implementation of the data model: it makes use only of some of the properties of *element* and *attribute* nodes, as defined in the data model.

Variables

For the purpose of interpreting RIF constructs with respect to the available data source documents represented using the XQuery 1.0 and XPath 2.0 Data Model, the domain of the variables must include all the nodes in the default available collection.

Class identifiers

Class identifiers in RIF class membership and subclass relationship formulas, represented by the `Member` and the `Subclass` construct, respectively, can be interpreted with respect to the available data source documents represented using the XQuery 1.0 and XPath 2.0 Data Model, as identifying subsets of the *element* nodes in the default available collection:

- a class identifier that is represented by a RIF constant of type `rif:iri` identifies the set of all the *element* nodes in the default available collection such that:
  - if the constant's value is an IRI where the local name has the form `type(NAME)` where `NAME` is either an IRI or an XML name from which an IRI can be obtained by casting into a QName and expanding the QName,
    - and, if the identifer is a relative IRI that with only the local part: the set of all the nodes in the default available collection whose **type-name** matches the `NAME` IRI (or the IRI obtained from `NAME`), if it is absolute, or the absolute IRI built from the node's default namespace, if there is one, and the `NAME` (or the IRI obtained from `NAME`), if it is not;
    - or, if the identifier is an absolute IRI: the subset of the above, such that the namespace part of the identifer matches the namespace part in the node's node-name;
  - else, if the constant's value is an absolute IRI: the set of all the nodes in the default available collection, such that that absolute IRI matches the node's **node-name** as an expanded QName. If the namespace in the IRI is the target namespace of an in-scope schema definition, the set of *element* nodes that it identifies contains also the nodes that represent the elements that belong to a substitution group where the head element's name matches the local part in the IRI, and that occur in a position where the schema requires the head element;
  - else, if the constant's value is a relative IRI: the set of all the nodes in the default available collection, such that
    - Either the node has a default namespace in its scope, and the relative IRI is the local name of the element. If the default namespace is the target namespace of an in-scope schema definition, the set of *element* nodes contains also the nodes that represent the elements that

belong to a substitution group where the head element's name matches the relative IRI, and that occur in a position where the schema requires the head element;
- or the node has a non-empty base URI, and the absolute URI built from the base URI and the relative URI matches the node's **node-name** as an expanded QName.

Editor's Note: ...or, simply: the relative URI matches the local name?

- A class identifiers that is not a constant of type `rif:iri`,
  - either is a constant of type `xs:string` whose value is an XML name, and it identifies the same set of *element* nodes in the default available collection as if the class identifier was the IRI obtained by casting the XML name into a QName and expanding the QName;
  - or can be cast into a valid `rif:iri` constant, and it identifies the same set of *element* nodes as if it were that `rif:iri` constant.

The interpretation, with respect to the available data source documents represented using the data model, of class identifiers that do not fall into one of the above categories is implementation dependent.

NB: the binding of prefixes to namespaces, when expanding QNames, comes from the **namespaces** property of the element nodes.

Editor's Note: One way to avoid the introduction of the `type(NAME)` form would be to interpret `rif:iri` constants as refering always to element names, and `xs:string` constant with XML name values as refering always to element types.

The node sets correspond to the sequences selected, respectively, by XPath 2.0 expressions of the form:

1. `/descendant::element(*,`*`QNAME`*`)` for identifiers where the local part has form `type(NAME)`;
2. or `/descendant::`*`QNAME`* otherwise (or `/descendant::schema-element(`*`QNAME`*`)` if the namespace is that of an in-scope schema definition).

Examples. TBC

Slot names

A frame, represented in RIF by the `Frame` construct, can be interpreted with respect to the available data source documents represented using the XQuery 1.0 and XPath 2.0 Data Model, as representing the relation between an *element* node and its *element* **children** or its **atributes**.

Specifically, in a frame where the object is an *element* node in the default available collection, the slot name can be interpreted with respect to subsets of that node's *element* **children** or **attributes**.

Given a frame `o [ slot -> v ]`, where `o` identifies an *element* node in the default available collection, let us call **`slot`-sequence**, the sequence of nodes specified as follows.:

- if `slot` is a constant of type `rif:iri`, or can be cast into a constant of type `rif:iri`, where the local name has the form `attribute(NAME)` where `NAME` is either an IRI or an XML name from which an IRI can be obtained by casting into a QName and expanding the QName, *the **`slot`-sequence** is the, possibly empty, subset of the **attributes** of the node identified by `o`, that contains only the node whose **node-name** matches the IRI (possibly completed with namespaces etc) **TBC***;
- else, if `slot` is a constant of type `rif:iri`, or can be cast into a constant of type `rif:iri`, the **`slot`-sequence** is the subset of the *element* nodes in the **children** property of the *element* node identified by `o`, whose **node-name** matches the IRI *(possibly completed if it is a relative IRI, see class identifiers, above; and including nodes representing elements in the substitution group, as appropriate) **TBC***;
- else, if `slot` is a constant of type `xs:string` whose value is an XML name, the **`slot`-sequence** is the same set of *element* nodes in the **children** property of the *element* node identified by `o`, as if the slot name was the IRI obtained by casting the XML name into a QName and expanding the QName.

Editor's Note: Multiple sub-elements with the same name are different values, since frames are multi-valued. The value of an object-oriented *object.attribute* term (or whatever the notation) would be the list containing the **`attribute`-sequence** as defined above, preserving the **document-order**.

The interpretation, with respect to the available data source documents represented using the data model, of frames whose slot names do not fall into one of the above categories is implementation dependent.

NB: the binding of prefixes to namespaces, when expanding QNames, comes from the **namespaces** property of the element nodes.

A frame `o [ slot -> v ]`, where `o` identifies an *element* node in the default available collection, is true if and only if

-
  - either `v` is the **typed-value** of one of the nodes with a defined **typed-value** in the **`slot`-sequence**;
  - or `v` is the identfier of one of the nodes in the **`slot`-sequence** whose **typed-value** is undefined.

Editor's Note: *...is true if and only if...*, or should it rather be *...is true if...*?

The `slot` sequences correspond to the sequences selected, respectively, by the XPath 2.0 expressions of the form:

1. `attribute::`*QNAME*, for slot names where the local part has the form `attribute(QNAME)`;
2. `child::`*QNAME*, resp. `child::schema-element(`*QNAME*`)` otherwise.

Examples. TBC

Conformance

TBC

The special case of RDF and OWL data sources

TBC

References

[DTD]
REF DTD tbd

[RIF-Core]
REF Core tbd

[RIF-FLD]
REF FLD tbd

[RIF-PRD]
REF PRD tbd

[RIF-RDF-OWL]
REF SWC tbd

[XF&O]
REF XF&O tbd

[XML-SCHEMA]
REF XML-S tbd

[XMLS-2]
REF XML-S part 2 tbd

[XPATH 2.0]
REF XPath 2.0 tbd

Appendix A: Glossay

TBC

Appendix B: Example

TBC