W3C Editor's Draft

RIF Combination with XML data

W3C Working Group Note 13 May 2011

This version:
http://www.w3.org/2005/rules/wg/draft/ED-rif-xml-data-20110513/
Latest editor's draft:
http://www.w3.org/2005/rules/wg/draft/rif-xml-data/
Previous version:
http://www.w3.org/2005/rules/wg/draft/ED-rif-xml-data-20100511/
Editors:
Christian de Sainte Marie, IBM

A color-coded version of this document showing changes made since the previous version is also available.

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May Be Superseded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/.

W3C Editor's Draft

Summary of Changes

@@@ tdb

Please Send Comments

Please send any comments to public-rif-comments@w3.org (public archive). Although work on this document by the Rule Interchange Format (RIF) Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion among developers is welcome at public-rif-dev@w3.org (public archive).

No Endorsement

Publication as a Working Group Note 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

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, with or without an associated XML schema. The specification relies on the abstract XQuery 1.0 and XPath 2.0 Data Model [XDM] to specify what information is accessible in the data, and on XPath 2.0 expressions [XPath 2.0] to select pieces of that information. It relies on XML Schema Component Designators [XSD-CD] expressions to provide unambiguous designators for XML schema types and schema elements in combinations where the XML data is associated with an XML schema.

XPath 2.0 is a non-XML expression language that allows the processing of values conforming to the data model defined in [XDM]. The result of an XPath expression may be a selection of nodes from the input data model, or an atomic value, or more generally, any sequence allowed by the data model. The name of the language derives from its most distinctive

feature, the path expression, which provides a means of hierarchic addressing of the nodes in an XML tree.

XML Schema Component Designators (XSD-CD) is a non-XML expression language that provides reliable and unambiguous designators for XML schema components. The specification divides the problem of constructing schema component designators into two parts: defining a designator for an assembled schema, and defining a designator for a particular schema component or schema components, understood relative to a designated schema. This specification uses only a limited subset of the schema component path expression language, from the second part.

According to this specification, XPath and component designator expressions are represented as string constants in RIF formulas. Although the semantics of RIF and XML data combinations is specified with respect to any valid XPath 2.0 expression, conforming implementations are required to support only a limited subset. Future versions of this specification may extend that subset.

The XQuery 1.0 and XPath 2.0 Data Model (XDM) is based on the *infoset* [Infoset], possibly augmented with information resulting from the validation of the XML data against an XML schema, or *post-schema validation infoset* (PSVI), according to the specification W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [XSD 1.1 Part 1]. An instance of the data model can also be constructed directly through application APIs, or from non-XML sources such as relational tables in a database. The semantics for the combination of RIF documents with XML data applies, therefore, to the combination of RIF documents with non-XML data as well, where the XML binding of the data is only used to refer to it in the interchanged rules. This is likely to be of particular interest when an XML schema is interchanged along with the RIF document.

Like the XQuery 1.0 and XPath 2.0 Data Model, the semantics of combinations of RIF documents and XML data that is specified in this document supports the following classes of XML data:

- 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 a normative standard semantics for RIF combinations with XML data: first, an XPath 2.0 and XSD-CD based syntax is specified, that is used to relate RIF formulas to XML data items in a combinations (section 2.1). Then, a model-theoretics semantics is given to RIF BLD combination with XML data, with and without associated XML schema (section 2.2); 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 2.3); 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 2.4).

Section 3 specifies how the `rif:Import` directive is extend to support the import of XML data and XML schemas in a RIF document.

Combination with XML data and/or schema constrains the interpretation of RIF formulas and give them special semantics. As a result, the semantics of RIF combined with XML data and schema is not the same as the semantics of RIF alone, even if the XML data and schema in the combination are empty.

Section 4 specifies the conformance criteria for this specification.

Implementation of this specification requires a good understanding of XPath 2.0 and the XQuery 1.0 and XPath 2.0 Data Model. However, this document can be read and understood without a deep knowledge of these two specifications. For the convenience of the reader, the definition of terms from the XPath 2.0, XSD-CD and the XDM specifications, that are of particular importance for this document, are repeated in the Appendix A: Glossary. However, the definition in the glossary are non-normative: the only normative definitions are as specified in the [XPath 2.0] and the [XDM] recommendations, and in the XSD-CD specification. Such terms are shown in square bracket, with a superscript indicating *XP* if the term is defined in XPath 2.0: [thus]^{XP}; or with a superscript indicating *XDM* if the term is defined in XDM: [thus]^{XDM}; or with a superscript indicating *CD* if the term is defined in XSD-CD: [thus]^{CD}.

This document follows the convention used in other RIF specifications, to start definition with: **Definition** and end them with the symbol □. In the same way, examples start with: **Example** and end with □. The end

symbol may be omitted if the definition or example ends with the section.

Several prefixes are used throughout this document for notational convenience. The following bindings are assumed.

1. `rif` bound to `http://www.w3.org/2007/rif`
2. `xs` bound to `http://www.w3.org/2001/XMLSchema`
3. `xml` bound to `http://www.w3.org/XML/1998/namespace`
4. `fn` bound to `http://www.w3.org/2005/xpath-functions`
5. `pred` bound to `http://www.w3.org/2007/rif-builtin-predicate`
6. `func` bound to `http://www.w3.org/2007/rif-builtin-function`
7. `xscd` bound to `http://www.w3.org/2009/xmlschema-ref`

2 RIF combination with XML data

This section specifies the normative semantics of the combination of RIF formulas 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 triple <*R*, *E*, *S*>, where *R* is a RIF formula (document or non-document), *E* is a, possibly empty, set of data model [nodes][XDM] that contains the information that is represented in the XML data, and *S* is a, possibly empty, set of XML schema definitions. □

For the purpose of evaluating the XPath 2.0 expressions that are used, in *R*, to access the XML data that is represented in *E*, the [in-scope schema definitions][XP] are the schema definitions in *S* and all their components.

This specification does not describe or prescribe how the set of data model [nodes][XDM], in a RIF+XML data combination, is obtained: it may result from parsing one or more XML documents and/or validating them against one or more XML schemas, or it may be created by methods other than parsing and/or schema-validating XML documents.

However, in a RIF+XML data combination <*R, E, S*>, any information item in *E* that represents an instance of a schema definition that is an element of *S* or a component in an element of *S*, must be valid with respect to that schema definition.

This specification does not prescribe the behaviour of a conformant implementation when the set of data model [nodes][XDM] in a RIF+XML data combination does not satisfy the above validity constraint, or any relevant constraint from the XPath 2.0, XDM or XSD-CD specifications.

2.1 Syntax

This section addresses how a RIF formula interacts with the XML data, in a RIF+XML data combination; that is, this section specifies how the access to XML data, and to XML schema information, if available, is represented in RIF formulas, for the purpose of combining them.

The basic idea is that the object-attribute-value semantics of RIF frame formulas is used, in the combination of RIF formulas and XML data, to exploit the intended semantics of the relation between a [context node][XP], an XPath 2.0 expression and the sequence that the expression matches in the context of the node: in the context of a RIF+XML data combination *<R, E, S>*, frame formulas of the form `object["`*`some XPath expression`*`"->val]`, where the `object` term is interpreted as an data model [node][XDM] in *E*, mean that `val` can be derived, in a way to be specified in this document, from the sequence of items that the XPath expression matches in the context of the node denoted by `object`.

This document specifies the semantics of RIF+XML data combinations, in general terms, for any valid XPath 2.0 expression. But some of the details are specified normatively for a limited subset of XPath 2.0 expressions only. Namely, conforming implementations must support the XPath 2.0 expressions specified by the following EBNF grammar:

```
RIFexpr               ::= StepExpr | ValueExpr | AbbreviatedExpr
AbbreviatedExpr       ::= ElementName | '@' AttributeName
ValueExpr             ::= 'fn:data(' ( StepExpr | '.' ) ')'
StepExpr              ::= StepAxis ( NameTest | KindTest ) ('[' Position ']
StepAxis              ::= 'self::' | 'child::' | 'attribute::'
NameTest              ::= ElementName | AttributeName
KindTest              ::= ElementTest
                        | AttributeTest
                        | SchemaElementTest
                        | SchemaAttributeTest
ElementTest           ::= 'element' '(' (ElementNameOrWildcard (',' TypeNam
SchemaElementTest     ::= 'schema-element' '(' ElementDeclaration ')'
ElementDeclaration    ::= ElementName
AttributeTest         ::= 'attribute' '(' (AttribNameOrWildcard (',' TypeNa
SchemaAttributeTest   ::= 'schema-attribute' '(' AttributeDeclaration ')'
AttributeDeclaration  ::= AttributeName
ElementNameOrWildcard ::= ElementName | '*'
ElementName           ::= QName
AttribNameOrWildcard  ::= AttributeName | '*'
AttributeName         ::= QName
TypeName              ::= QName
Position              ::= [1..9] [0..9]*
```

Future versions of this specification may extend that subset.

XPath expressions are represented in RIF as `xs:string` constants.

> **Editor's Note:** Alternatively, an additional symbol space, `rif:xpath`, could be added to RIF built-in data types: in that case, XPath expressions would be represented as `rif:xpath` constants.

XPath 2.0 unabbreviated syntax must be used in the normative RIF/XML syntax, except when the expression is a simple `StepExpr` and the test is a `NameTest`: in that case, the abbreviated syntax must be used in the normative RIF/XML syntax.

Both syntaxes, abbreviated and unabbreviated, are allowed in the non-normative RIF presentation syntax.

When a string constant is evaluated as an XPath expression, the [statically known namespaces]^{XP} are the in-scope namespaces for the containing element in the RIF formula.

Likewise, the object-class and subclass-superclass semantics of RIF member and subclass formulas is used to exploit the intended semantics of the relation between an XML element and its XML schema type, and of the relation between two XML schema types, when information on classes and types definitions is available from imported XML schemas. RIF member formulas of the form: `object # "`*`an XSD-CD component path expression`*`"`, where the `object` term is interpreted as a element [node]^{XDM} in $e \in E$, mean that the XML fragment represented by e satisfies the XML schema definition designated by the XSD-CD component path expression.

In the same way, subclass expressions of the form: "*sub*" ## "*SUP*" where, both, "*sub*" and "*SUP*" are XSD-CD schema component path expressions, mean that the schema types T_{sub} and T_{SUP}, whose definitions are designated by the XSD-CD expressions "*sub*" and "*SUP*", respectively, satisfy *[derived-from]*^{XP}*(*T_{sub}*,* T_{SUP}*)*, where the pseudo-function *derives-from* is defined as in XPath 2.0, section 2.5.4 - Sequence type matching.

This specification specifies the semantics of RIF+XML data combinations that contains such member or subclass formulas for a limited subset of XSD-CD expressions, only. Namely, conforming implementations must support absolute XSD-CD expressions that designate schema types, as specified by the following EBNF grammar:

```
RIFSchemaComponentPath       ::=  StepSepator RelativeSchemaComponentPath
RelativeSchemaComponentPath  ::=  Step (StepSeparator RelativeSchemaCompone
```

```
StepSeparator                ::=  '/'
Step                         ::=  AbbrevStep
AbbrevStep                   ::=  AbbrevElementStep | AbbrevTypeStep
AbbrevElementStep            ::=  NameTest
AbbrevTypeStep               ::=  '~' NameTest
NameTest                     ::=  QName | '0'
```

XSD-CD component path expressions are represented in RIF as `xs:string` constants.

> **Editor's Note:** Alternatively, an additional symbol space, `rif:xsd-cd`, could be added to RIF built-in data types: in that case, XSD-CD expressions would be represented as `rif:xsd-cd` constants.

XSD-CD abbreviated syntax must be used in the normative RIF/XML syntax. Both abbreviated and unabbreviated syntaxes are allowed in the non-normative RIF presentation syntax.

For the purpose of expanding QNames in NameTests, the in-scope namespace bindings are defined as all the namespace declaration in the scope of which the string is.

Example 2.1. Consider, for instance, the following rule, that says that an *EarlyCustomer* is a *Customer* whose *Account* number is lower than 1000 (using the RIF presentation syntax and XPath abbreviated syntax):

```
Forall ?x, ?y
   (_EarlyCustomer(?y) :-
   And( ?x["ex:Name" -> ?y]
        Exists ?z (And ?x["ex:Account" -> ?z]
                       External(pred:numeric-less-or-equal(External(xs:inte
```

Consider, further, the following XML fragment, 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>
```

Assuming, like in all further examples, that the pair *(ex, <http://example.org/customertable>)* belongs to the statically known namespaces when the XPath expressions are evaluated, the combination, under the semantics described below, of the above rule and XML data, with no associated XML schema, will entail two new facts, namely:

```
_EarlyCustomer("John")
_EarlyCustomer("Jane")
```

This rule shows how an XPath expression is used, in the rule, as an `xs:string` constant, e.g.: `"child::schema-element(ex:Name)"` or `"child::schema-element(ex:Account)"`, to access the children of an element [node][XDM] as frame slots.

Notice that, in the absence of classes and types definitions as would be provided by an associated XML schema, `NameTest` are enough to exploit all the information that is avalable in the XML data, in the combination. Notice further that, with no typing information available, atomic values retrieved from the XML data can only be interpreted as strings. A consequence is that such values have to be cast into the required types when used as arguments to RIF buit-in functions and predicates.

Notice, also, that, for the same reason, member (or subclass) formulas cannot be used in combination with the XML data: the only way (based only on the information that is available in the example) to ensure that only the `ex:Name` and `ex:Account` sub-elements of `ex:Customer` elements are taken into account would be to add a frame formula to the condition, to check that the variable `?x` is bound to an element that is, itself, named `ex:Customer"`:

```
Forall ?x, ?y
   (_EarlyCustomer(?y) :-
   And( ?x["ex:Name" -> ?y]
        ?x["self::ex:Customer"->?x]
        Exists ?z (And ?x["ex:Account" -> ?z]
                       External(pred:numeric-less-or-equal(External(xs:inte
```

Notice that, in the absence of an XML schema against which the data must be valid, the fact `?x["self::ex:Customer"->?x]` does not entail anything with respect to other properties of `?x`.

Now, assume that the following XML schema is associated with the above XML fragment:

```
<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>
```

When the XML schema is associated with the data, in the RIF+XML data combination, typing information can be taken into account, both, to focus on the data that represent customer information (that is, information represented in `ex:Customer` elements), and to ensure that arguments to built-in functions and predicates are properly typed. The example rule could, then, be written as follows (notice that this version of the rule uses unabbreviated XPath syntax):

```
Forall ?x, ?y
    (_EarlyCustomer(?y) :-
    And( ?x # "/ex:Customer"
```

```
            ?x["child::schema-element(ex:Name)" -> ?y]
            Exists ?z (And ?x["child::schema-element(ex:Account)" -> ?z]
                           External(pred:numeric-less-or-equal(?z 1000)))))
```

In that case, the assertion of the class membership: `?x # "/ex:Customer"`, entails that `?x` has all the properties that are required to validate against the schema definition of the element `ex:Customer`.

Another rule example, below, shows how another kind of XPath expression, used as an `xs:string` constant: `"@xml:lang"</nowiki>>`, is used, in a RIF+XML data combination, to access an attribute [node]^{XDM} as a frame slot:

```
Forall ?x, ?y
   (_SpeaksEnglish(?y) :-
   And( ?x["@xml:lang" -> "en"^^xs:language])
        ?x["ex:Name" -> ?y])
```

Assuming that the pair *(xml, <http://www.w3.org/XML/1998/ namespace>)* is also in the statically known namespaces, that rule, combined with the example XML data and XML schema, entails a single new fact:

```
_SpeaksEnglish("John")
```

Notice that all the rules, above, are well-formed RIF formulas, that can be validly consumed without being combined with any XML data and/or XML schemas. In that case, of course, this specification adds nothing to the specification of the RIF Core Dialect, as regards the semantics of the rules.

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

The specification of the model-theoretic semantics of a RIF BLD+XML data combination -- a RIF+XML data combination, *<R, E, S>* where *R* is a RIF-BLD (document or non-document) formula -- follows closely the specification of the semantics of a RIF BLD formula, 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, extended with an additional mapping, $\mathbf{I}_{DM}$, that interprets data model [nodes]^{XDM}, and constrained with additional conditions, derived from the information in *E* and *S*.

2.2.1 Semantic structures

Definition (Semantic structure). For the purpose of defining the model-theoretic semantics of a RIF BLD+XML data combination <*R, E, S*>, a semantic structure is as a tuple $I = <\boldsymbol{TV}, \boldsymbol{DTS}, \boldsymbol{D}, \boldsymbol{D}_{\text{ind}}, \boldsymbol{D}_{\text{func}}, \boldsymbol{I}_{\text{DM}}, \boldsymbol{I}_{\text{C}}, \boldsymbol{I}_{\text{V}}, \boldsymbol{I}_{\text{F}}, \boldsymbol{I}_{\text{NF}}, \boldsymbol{I}_{\text{list}}, \boldsymbol{I}_{\text{tail}}, \boldsymbol{I}_{\text{frame}}, \boldsymbol{I}_{\text{sub}}, \boldsymbol{I}_{\text{isa}}, \boldsymbol{I}_{=}, \boldsymbol{I}_{\text{external}}, \boldsymbol{I}_{\text{truth}}>$, where $\boldsymbol{TV}$, $\boldsymbol{DTS}$, $\boldsymbol{D}$, $\boldsymbol{D}_{\text{ind}}$, $\boldsymbol{D}_{\text{func}}$, $\boldsymbol{I}_{\text{C}}$, $\boldsymbol{I}_{\text{V}}$, $\boldsymbol{I}_{\text{F}}$, $\boldsymbol{I}_{\text{NF}}$, $\boldsymbol{I}_{\text{list}}$, $\boldsymbol{I}_{\text{tail}}$, $\boldsymbol{I}_{\text{frame}}$, $\boldsymbol{I}_{\text{sub}}$, $\boldsymbol{I}_{\text{isa}}$, $\boldsymbol{I}_{=}$, $\boldsymbol{I}_{\text{external}}$ and $\boldsymbol{I}_{\text{truth}}$ are defined as in RIF-BLD ([RIF-BLD], section 3.2, Semantic structures), and where

- $\boldsymbol{I}_{\text{DM}}$ is a total mapping from *E* to $\boldsymbol{D}_{\text{ind}}$, that interprets data model [nodes]$^{\text{XDM}}$.

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

Let *Classifiers(S)* denote the set of all the well-formed `RIFSchemaComponentPath` expressions, *expr*, such that one of the following holds, where *c* is the component selected by *expr* in the context of *S*:

- either *c* defines a named complex type, that is, the [component-kind()]$^{\text{CD}}$ of *c* is `xscd:complex-type-definition` and its [component-name()]$^{\text{CD}}$ is a QName;
- or *c* is the declaration of or a reference to an element with an anonymous complex type, that is: the [component-kind()]$^{\text{CD}}$ of *c* is `xscd:element-declaration`, its [component-name()]$^{\text{CD}}$ is a QName, the [component-kind()]$^{\text{CD}}$ of the child component of *c* along the [type axis]$^{\text{CD}}$ is `xscd:complex-type-declaration` and its [component-name()]$^{\text{CD}}$ is '0' (indicating an anonymous component).

Let, further, *Classes(S)* $\subseteq$ *Classifiers(S)* denote the subset of classifier XSD-CD expressions that select the declarations of named complex types.

Definition (RIF BLD+XML data combined interpretation). A ***RIF BLD+XML data combined interpretation*** is a triple (*E*, *I*, *S*), where *E* is a set of data model [nodes]$^{\text{XDM}}$, possibly empty; *S* is a set of XML schema definitions, possibly empty; and *I* is a semantic structure, such that all the following holds:

1. For all the [nodes]$^{\text{XDM}}$ $e \in E$, $\boldsymbol{I}_{\text{truth}}(I_{\text{frame}}(I_{\text{DM}}(e))(I_{\text{C}}(\text{"}expr\text{"}\text{^^}\texttt{xs:string}), RIFValue(e, expr))) =$

t (true) for any well-formed XPath expression, *expr*, for which *RIFValue(e, expr)* is defined; and

2. For all the individuals $o \in D_{Ind}$, there is an element [node]^{XDM} with a complex type, $e \in E$, such that $I_{DM}(e) = o$, if and only if there is an XSD-CD component path expression, *expr* $\in$ *Classifiers(S)*, such that $\mathbf{I}_{truth}(I_{isa}(o, I_C(\text{"}expr\text{"}\text{^^xs:string}))) = \mathbf{t}$ (true) and
   - either *expr* $\in$ *Classes(S)* and the expanded QName of the type in the selected definition is equal to the [declared type]^{XDM} of *e*;
   - or *e* represents an XML element that satisfies the element definition selected by *expr*;
3. ***DTS*** is extended to include all the named atomic types defined in the [in-scope schema definitions]^{XP};
4. $\mathbf{I}_{truth}(I_{sub}(I_C(\text{"}T_1\text{"}\text{^^xs:string}), I_C(\text{"}T_2\text{"}\text{^^xs:string})) = \mathbf{t}$ (true) for all the pairs of XSD-CD component expressions, $(T_1, T_2) \in$ *Classes(S)* $\times$ *Classes(S)*, that select two complex types, t_1 and t_2, such that $[derives\text{-}from]^{XP}(t_1, t_2)$ is true;

□

Notice that clause 2, in the above definition, implies that, if the class membership of an object, that interprets the [context node]^{XP} *e*, in a classifier amongst the schema definitions in the combined interpretation is asserted, then the object must have all the properties that are mandated by the classifier, and no property that the definition of the classifier does not allow. This is a consequence of the constraint that all the schema element [node]^{XDM}, in *E*, must be schema valid.

Editor's Note: Another possibility, to resolve the issue of incomplete instances, would be to introduce one or more null values in the RIF vocabulary, e.g. `rif:null`.

RIFValue is a total mapping from $E \times EXPR$ to D_{ind}, where *EXPR* is the set of all the well-formed XPath expressions. *RIFValue(e, expr)* associates a value in D_{ind} to the sequence of data model [nodes]^{XDM}, in *E*, that are matched by the XPath expression *expr* when the [context node]^{XP} is *e* and *S* contains the [in-scope schema definitions]^{XP}.

RIFValue is defined as follows.

Let $seq = (i_1,...,i_n)$ be the sequence of items matched by *expr* $\in$ *EXPR* in the context of $e \in E$, where $n \geq 0$ is the number of items in *seq*:

- if $n > 1$, $val = I_{list}(Val(i_1),...,Val(i_n))$
- if $n = 1$,

1. if *seq = (i)* is the [typed value]^{XDM} of an element or attribute [node]^{XDM} with a list type, then *val* = I_{list}(*Val(i)*);
2. else, if *expr* selects an schema attribute or child element that is optional (that is, zero or one occurrence) in the context of *e*, then *val = Val(i)*;
3. else, if *i* is a schema element [node]^{XDM}, and the schema definition of *e* allows for multiple occurrences of that schema element, then *val* = I_{list}(*Val(i)*);
4. otherwise, *val* = I_{list}(*Val(i)*) or *val = Val(i)*: according to the XPath 2.0 specification, an item is identical to a singleton sequence containing that item. Except for the multiple occurences and list type cases, that must be handled as specified above, the decision to handle a singleton as a single item or as a singleton sequence that contains that item is left to the implementation, but it must be consistent: that is, the sequence matched by the same XPath expression, *expr* ∈ *EXPR*, must be handled the same way, as a sequence or as an item, for all the [context nodes]^{XP}, *e* ∈ *E*;

- if *n = 0*, that is, if *expr* matches no item in the context of *e*,
  1. if *expr* selects a schema element for which the schema allows zero to more than one occurences in the context of *e*, then *val* = I_{list}(∅);
  2. otherwise, *RIFValue(e, expr)* is undefined.

Val interprets items as follows:

- if *i* is an atomic value with type `xs:untypedAtomic`, then *Val(i)* is the value of *i* as an `xs:string`;
- else, if *i* is an atomic value, *Val(i) = i*;
- else, if *i* is a [node]^{XDM} without a type name, or whose type name is `xs:untyped` or `xs:untypedAtomic`, with no attributes and with no or only text [node]^{XDM} children, then *Val(i)* is the [string value]^{XDM} of *i*: *Val(i)* = `fn:string(`*i*`)`;
- else, if *i* is a [node]^{XDM} with a simple type, then
  - if *i* is a [node]^{XDM} with a list type, then *val* is the [typed value]^{XDM} of *i* as a RIF list: *Val(i)* = I_{list}(`fn:data(`*i*`)`);
  - otherwise *Val(i)* is the [typed value]^{XDM} of *i*: *Val(i)* = `fn:data(`*i*`)`;
- otherwise, that is, if *i* is a [node]^{XDM} with a complex type, then *Val(i)* is the interpretation of *i* according to I_{DM}: *Val(i)* = I_{DM}(*i*).

In the above, `fn:string` and `fn:data` are the accessors defined in the Xpath 2.0 and XQuery 1.0 Functions and Operators recommendation

[XFO], sections 2.3 and 2.4 (for the reader's convenience, the definitions are copied, non-normatively, in the Glossary.)

A RIF BLD+XML data combined interpretation (*E, I, S*) determines the truth value, $TVal_{(E,I,S)}(\varphi)$, of a RIF-BLD non-document formula φ combined with the XML data in *E* and XML schema definitions in *S*, like a RIF BLD semantic structure, I_{BLD}, determines the truth value $TVal_{I_{BLD}}(\varphi)$, of a RIF-BLD non-document formula φ. The mapping $TVal_{(E,I,S)}$ is defined from the set of all non-document formulas to **TV**, the set of truth values in *I*, as follows.

Definition (Truth valuation of RIF BLD non-document formulas combined with XML data). ***Truth valuation*** of RIF BLD non-document formulas combined with XML data is determined using the function $TVal_{(E,I,S)}$, such that for all RIF BLD+XMl data combination, $<\varphi, E, S>$, where φ is a non-document RIF BLD formula, $TVal_{(E,I,S)}(\varphi) = TVal_I(\varphi)$, where $TVal_I$ is as defined in [RIF-BLD], section 3.4 - Interpretation of non-document formulas, with *I* being the semantic structure in the RIF BLD+XML data combined interpretation (*E, I, S*). □

Example 2.2. Assuming that *E* contains all the data model [nodes]^{XDM} in the data model instance that describes the XML sample fragment in example 2.1; that e_{table} denotes the element node, in *E*, that represents the information that is serialized in the `CustomerTable` element, in the example XML fragment; and that e_{John} denotes the element node, in *E*, that represents the information that is serialized in the first `Customer` sub-element (the one whose `Name` sub-element contains *John*), the following ground frames must be true in all the RIF BLD+XML data combined interpretations, (*E, I,* ∅), that is, where *E* is constructed from the infoset only, and where I_C(_John) = $I_{DM}(e_{John})$ and I_C(_Table) = $I_{DM}(e_{table})$:

```
1. _John ["@xml:lang" -> "en"]
2. _John ["ex:Name" -> "John"]
3. _John ["ex:Account" -> "111" ]
4. _Table ["ex:Customer[1]" -> _John ]
```

Assuming that *S* contains all the top-level schema definitions in the sample schema in example 2.1, ground frames #1 and #3 above must be false in all the RIF BLD+XML data combined interpretations, (*E, I, S*), everything else being equal; instead, ground frames #1 and #2, below, must be true:

```
1. _John ["@xml:lang" -> "en"^^xs:language]
2. _John ["ex:Account" -> 111 ]
```

The class membership formula #1, below, may be true in a RIF BLD+XML data combined interpretation, (*E, I,* ∅), but, in the absence of an

associated schema definition, the string `/ex:Customer` does not represent a component designator, and the class membership formula has no meaning specific to the RIF BLD+XML data combined interpretation. Howver, that same class membership formula must be true in every RIF BLD+XML data combined interpretations, (*E, I, S*):

1. `_John # "/ex:Customer"`

On the other hand, the class membership formula: `_PIN_Jane # "/ex:PIN"` must never be true as a consequence of being interpreted under a RIF BLD+XML data combined interpretations, (*E, I, S*). Indeed, `ex:PIN` is defined as a simple type: as such it is not in *Classe(S)* (nor in *Classifiers(S)*); but it is, instead, added to ***DTS***.

Example 2.3. Consider the following element with mixed-content, where the prefix ex is associated with the IRI *http://example.org/*, and assume that an element [node]^{XDM}, *e*, that represents it is included in *E*.

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

The following fact must be true in any RIF BLD+XML data combined interpretation, *(E, I, S)*, such that I_C(`_myLetter`) = I_{DM}(*e*):

- `_myLetter["fn:data(.)" -> "Thank you for ordering the Widget. It should arrive by 09-09-09"]`

Indeed, the element has a mixed content: per [XDM] (section 6.2), its [typed value]^{XDM} is, therefore, its string value as an `xs;untypedAtomic`, which is represented, per the definition of *RIFValue*, above, by that same [string value]^{XDM} as an `xs:string`.

2.2.3 Combined interpretation of RIF BLD documents and XML data

For the purpose of the interpretation of imported documents, RIF BLD defines the notion of semantic multi-structures, which are nonempty sets, {*J,I;* I^{i1}, I^{i2}, ...}, of semantic structures that differ only in interpretation of local constants.

In the same way, a RIF-BLD+XML data combination <*R, E, S*>, is interpreted using a RIF BLD+XML data combined multi-interpretation (*E*, $\hat{I}$, *S*), where $\hat{I}$ is a nonempty set, {*J,I;* I^{i1}, I^{i2}, ...}, of semantic structures that determine a non-empty set, {(*E, J, S*), (*E, I, S*); (*E,* I^{i1}, *S*), (*E,* I^{i2}, *S*),

...} of RIF BLD+XML data combined interpretations that differ only in interpretation of local constants.

To keep this document simple, and in the same way the definition of truth valuation of RIF BLD formulas combined with XML data, above, refers to the definition of truth valuation in the RIF-BLD recommendation, the definition of a RIF BLD+XML data combined multi-interpretation refers to the definition of a semantic multi-structure in [RIF-BLD].

Definition (RIF BLD+XML data combined multi-interpretation). A ***RIF BLD+XML data combined multi-interpretation*** is a triple, (E, $\hat{I}$, S), where $\hat{I}$ is a semantic multi-structures as defined in [RIF-BLD] (section 3.5 - Interpretation of documents), with the additional condition that for each semantic suctrure, $i \in \hat{I}$, (E, i, S) is a RIF BLD+XML data combined interpretation. □

Similarly, the truth valuation of a RIF+XML data combination <R, E, S>, is specified with reference to the truth valuation of a RIF BLD document formulas, using the RIF BLD+XML data combined multi-interpretation (E, $\hat{I}$, S) in place of a RIF-BLD semantic multi-structure.

Definition (Truth valuation of RIF BLD document formulas combined with XML data). ***Truth valuation*** of RIF BLD document formulas combined with XML data is determined using the function $TVal_{(E,\hat{I},S)}$, such that for all RIF BLD+XML data combination <P, E, S>, where P is a document RIF BLD formula, $TVal_{(E,\hat{I},S)}(P) = TVal_{\hat{I}}(P)$, where $TVal_{\hat{I}}$ is as defined in [RIF-BLD], section 3.5 - Interpretation of documents, with $\hat{I}$ being the semantic multi-structure in the RIF BLD+XML data combined multi-interpretation (E, $\hat{I}$, S). □

The notions of model and of logical entailment can be defined for a RIF BLD+XML data combination, based on the definitions of model and of logical entailment for RIF BLD (document and non-document) formulas: (E, $\hat{I}$, S) is a model of a RIF BLD+XML data combination <R, E, S>, if and only if $\hat{I}$ is a model of R, in the sense defined in [RIF-BLD] (section 3.6 - Logical entailment), and (E, $\hat{I}$, S) is a RIF BLD+Xml data combined multi-interpretation.

Definition (Model of a RIF BLD+XML data combination). A RIF BLD+XML data combined multi-interpretation (E, $\hat{I}$, S) is a ***model*** of a RIF BLD+XML data combination <R, E, S>, where R is a RIF BLD document or non-document formula, if and only if $TVal_{(E,\hat{I},S)}(R)$ = **t** (true). □

Definition (Logical entailment of a RIF BLD+XML data combination). Let φ and ψ be (document or non-document) formulas. We say that a RIF BLD+XML data combination <φ, E, S> ***entails*** the RIF BLD+XML data combination <ψ, E, S>, denoted <φ, E, S> |= <ψ, E, S>, if and only if every model of the RIF BLD+XML data combination <φ, E, S> is also a model of the RIF BLD+XML data combination <ψ, E, S>. □

Notice that, in the case where *E* and *S* are empty, that is, if a RIF-BLD formula is combined with an empty XML data set and an empty set of XML schema definitions, the interpretation of that RIF BLD formula under a RIF BLD+XML data combined multi-interpretation (∅, *Î*, ∅) is still different from its interpretation under the standard RIF BLD semantics as defined in [RIF BLD].

Example 2.4. Let *expr* be a well-formed XPath 2.0 expression, and let

- φ = `Forall ?x ?y ?z (?x["fn:data(expr)"->?z] :- ?x["expr"->?y])`
- ψ = `Forall ?x ?y ?z (?y["fn:data(.)"->?z] :- ?x["expr"->?y])`

Notice that <*φ*, ∅, ∅> |= <*ψ*, ∅, ∅>, provided that the pair *(fn, http://www.w3.org/2005/xpath-functions)* belongs to the statically known namespaces when the XPath expressions are evaluated; but *φ* |≠ *ψ*. This is because the `xs:string` constants `"expr"`, `"fn:data(expr)"` and `"fn:data(.)"` are interpreted as simple string constants under the standard RIF BLD semantics, whereas they are interpreted according to the XPath 2.0 semantics under the semantics of RIF+XML data combination.

Editor's Note: An alternative, less conservative definition for logical entailment in a RIF BLD+XML data combination would be: *Let φ and ψ be (document or non-document) formulas. We say that φ entails ψ in a RIF BLD+XML data combination if and only if every model (*E, Î, S*) of any RIF BLD+XML data combination* <φ, E, S> *is also a model of the RIF BLD+XML data combination* <ψ, E, S>.□ The use cases for the broader definition remain to be examined.

2.3 Operational semantics of RIF PRD+XML data combinations

The effect on the operational semantics of a RIF PRD formula of the combination with XML data, with or without an associated XML schema, is that all the ground facts that must be true under a RIF BLD+XML data combined interpretation, must be added to the set of ground facts that represents the state of the fact base.

Notice that, whereas the other clauses affect only the initial state of the fact base, clause 2 in the definition of a RIF BLD+XML data combined interpretation modifies the semantics of the `Assert` action, when the action asserts the membership of a new object in a class that is

interpreted as a schema classifier. Indeed, clause 2, in the definition, implies that the object in a membership relationship with an XML schema classifier (element or type) must interpret an instance of the class: that is, the new member object must validate against the schema definition associated to the class. As a consequence, in the context of a RIF PRD+XML data combination, <*R, E, S*>,

```
Assert( object # ClassExpr )
```

where *ClassExpr* ∈ *Classifiers(S)*, requires, as a prerequisite, that, for all the XPath 2.0 expressions, *expr*, that select a mandatory sub-element or attribute of the class, the following fact be true:

```
object [ "expr" -> value ]
```

where `value` has a value that is valid with respect to the schema definition of the selected sub-element or attribute.

Formally, that means that the definition of the RIF-PRD transition relation, defined in [RIF-PRD] (section 3.2, Operational semantics of atomic actions), is modified as follows.

Let *W* denote the set of all the states of the fact base, and *L* denote the set of all the ground atomic actions in the RIF-PRD action language, as defined in [RIF-PRD].

Let, further, *expr*-valid$_{<R,E,S>}$, denote a property of a constant, *o* ∈ *Const*, with respect to a state of the fact base, *w* ∈ *W*, in the context of a RIF PRD+XML data combination <*R, E, S*>. We say that o is *expr*-valid$_{<R,E,S>}$ in *w* if and only if *expr* ∈ *Classifiers(S)* and all the following are true, where *Φ* is a set of ground atomic formulas that represents *w*:

- for each child element, *c*, with QName: *name*, that is required in the content model defined by the schema definition designated by *expr* in *S*, there is a frame: *o["child::*name"*->value]*, in *Φ*, such that *value* has a value that is valid for *c* according to the schema definition;
- for each attribute, *a*, with QName: *name*, that is required in the content model defined by the schema definition designated by *expr* in *S*, there is a frame: *o["attribute::*name"*->value]*, in *Φ*, such that *value* has a value that is valid for *a* according to the schema definition;
- there is no subset, $F = \{ o["e"->v_1], ..., o["e"->v_n]\} \subseteq \Phi$, $n \geq 0$, of frames with object: *o*, and slot name: "*e*", where *e* is an XPath 2.0 child or attribute step expression, such that
  - either the child element or attribute designated by *e* is not allowed by the schema definition;
  - or the set of values, $\{v_i\}$, contains a value that is not valid, according to the schema definition, for the child element or attribute designated by *e*;

- or the number of facts in *F*, *n*, is not within the bounds set in the schema definition for the multiplicity of the child element or attribute designated by *e*.

Definition (RIF-PRD transition relation in a RIF-PRD+XML data combination). In a RIF-PRD+XML data combination <*R, E, S*>, the semantics of RIF-PRD atomic actions is specified by the transition relation $\rightarrow_{<R,E,S>} \subseteq W \times L \times W$. A triple $(w, \alpha, w') \in \rightarrow_{<R,E,S>}$ if and only if $w \in W$, $w' \in W$, α is a ground atomic action, and

- if α is `Assert(`φ`)`, where φ = o `#` "*expr*" and *expr* ∈ *Classifiers(S)*, then o is *expr*-valid$_{<R,E,S>}$ in *w* and *w'* = *w* ∪ *{φ};*
- Otherwise, that is, if α is an `Assert` whose target has a different form, or a `Retract`, `Modify` or `Execute` ground atomic action, then $(w, \alpha, w') \in \rightarrow_{<R,E,S>}$ if and only if $(w, \alpha, w') \in \rightarrow_{\text{RIF-PRD}}$, where $\rightarrow_{\text{RIF-PRD}}$ is the standard RIF-PRD transition relation as defined in [RIF-PRD] (section 3.2, Operational semantics of atomic actions).

□

The operational semantics of a RIF PRD+XML data combination is, then, exactly as specified in [RIF-PRD], except that $\rightarrow_{<R,E,S>}$ replaces $\rightarrow_{\text{RIF-PRD}}$ in the definition of the transition relation, $\rightarrow_{\text{PRS}}$, that is one of the components of a RIF-PRD production rule system, *PRS* (cf. [RIF-PRD], Section 4.2.4, Operation semantics of a production rule system).

Notice, further, that, since RIF-PRD, unlike RIF-BLD, allows class membership to be asserted for new objects only, the object in such an assertion cannot have some or all the required properties already. As a consequence, all the mandatory properties must be asserted in the same action block where the class membership is asserted.

Formally, the consequence is that the combination with schema-valid XML data puts an additional well-formedness constraints on RIF-PRD action blocks.

Definition (Well-formed action block in a RIF-PRD+XML data combination). In a RIF-PRD+XML data combination <*R, E, S*>, an action block is ***well-formed*** if an only if it is a well-formed RIF-PRD action block, according to the definition in [RIF-PRD] (section 3.1.3 - Well-formed action blocks) and it satisfies the following additional constraint:

- if an action in the action block asserts a membership atomic formula, `Assert(o # "expr")`, where *expr* ∈ *Classifiers(S)*, then the other actions in the action block are sufficient to make o *expr*-valid$_{<R,E,S>}$ in any state of facts resulting from the execution of the action block, that is: all the ground frame formulas that are required to make o *expr*-valid$_{<R,E,S>}$ in a state of facts, are asserted in the action block.

□

Editor's Note: Strictly speaking, since actions are ordered in RIF-PRD action blocks, and since the operational semantics of atomic actions is defined with respect to a production rule system state, independently from the system state being a system cycle state or a system transitional state, the ground frame formulas that are required to make o *expr*-valid$_{<R,E,S>}$ in a state of facts should be asserted before the membership formula is. However, it is conjectured that the atomic actions in an action block that satisfies the weaker constraint can, always, be re-orderd to satisfy the stronger constraint, without otherwise affecting the semantics of the action block as a whole; whence the absence of the ordering constraint in the above definition.

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.

Example 2.5. In a RIF-PRD+XML data combination *<R, E, S>*, where *S* contains all the top-level definitions in the XML schema in example 2.1 (all the examples assume that the pair *(ex, <http://example.org/customertable>)* belongs to the [statically known namespaces]$^{small>XP</small>}$ when the XPath and XSD-CD expressions are evaluated):

- the following action blocks are well-formed as the action part of rules in *R*

```
Do ((?newPin New())
    Assert(?newPin["self::ex:PIN"->999]) )

Do ((?newPin New())
    Assert(?newPin["self::ex:PIN"->"999"^^ex:PIN]) )
```

(ex:PIN is an atomic type defined in S*. As a consequence, it is included, per clause 3 in the definition of a RIF BLD+XML data combined interpretation, in the set of data types that are taken into account to interpret RIF formulas.)*

```
Do ((?newPin New())
    Assert(?newPIN # "/ex:PIN") )
```

(Since ex:PIN is a simple type, the XSD-CD expression /ex:PIN *is not in* Classifiers(S)*. As a consequence, the class membership assertion, in the example above, adds no well-formedness constraint that is specific to the RIF PRD+XML data combination: its semantics are standard.)*

```
Do ((?newCust New())
    Assert(?newCust # "/ex:Customer")
    Assert(?newCust["ex:Name"->"Jojo"])
    Assert(?newCust["ex:Account"->1111])
    Assert(?newPin["ex:PIN"->999])
    Assert(?newCust["@xml/lang"->"fr"^^xs:language]) )

Do ((?newCust New())
    Assert(?newCust # "/ex:Customer")
    Assert(?newCust["ex:Name"->"Jojo"])
    Assert(?newCust["ex:Account"->1111])
    Assert(?newCust["@xml/lang"->"fr"^^xs:language]) )
```

- the following action blosks are not well-formed as the action part of rules in *R*

```
Do ((?newCust New())
    Assert(?newCust # "/ex:Customer")
    Assert(?newCust["ex:Name"->"Jojo"])
    Assert(?newCust["ex:Account"->1111])
    Assert(?newPin["ex:PIN"->-5])
    Assert(?newCust["@xml/lang"->"fr"^^xs:language]) )
```

(-5 is out of range for an `ex:PIN` value)

```
Do ((?newCust New())
    Assert(?newCust # "/ex:Customer")
    Assert(?newCust["ex:Name"->"Jojo"])
    Assert(?newCust["ex:Name"->"Le balafré"])
    Assert(?newCust["ex:Account"->1111])
    Assert(?newCust["@xml/lang"->"fr"^^xs:language]) )
```

(The schema definition requires a single `ex:Name` value for each `ex:Customer` information item).

2.4 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.

3 Importing XML data and XML 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. [RIF-RDF-OWL] extends it to permit the identification of 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 further to permit the identification of XML data and/or XML schemas to be combined with a RIF document.

One use case for the combination of RIF and XML data is when a RIF document imports explicitly the data with which the rules are to be combined: 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 combine them with 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*.

3.1 The extended `Import` directive

Specifically, the `Import` directive is extended as follows:

- The import of any XML document is allowed; that is, the IRI of any XML document is allowed in the `location` sub-element of the `Import` directive. The IRI identifies the imported data document);
- Two new profiles are defined:
    1. the value `http://www.w3.org/2007/rif-import-profile#xml-data` is allowed in the `profile` sub-element of the `import` directive, to indicate that no XML schema is associated with the imported XML data;
    2. any IRI that identifies an XML schema is allowed in the `profile` sub-element of the `import` directive, to indicate

that the schema definitions in the XML schema are part of the combination;

- Finally, a new value: `http://www.w3.org/2007/rif-import-location#consumer-side-data`, is allowed for the location of an import. That value indicates that the `profile` in the `Import` directive applies to consumer-side data only.

The following constraints must be satisfied:

- If the content of the `location` in an `Import` directive is `http://www.w3.org/2007/rif-import-location#consumer-side-data`, the `profile` must contain the IRI of an XML schema;
- Conversely, if the profile is `rif:xml-data`, the `location` must identify an XML document;
- If the `profile` of an `Import` directive contains the IRI of an XML Schema and if the XML document identified in the `location` sub-element of that `Import` directive provides a `schema-location`, the `profile` must identify the same XML schema as the `schema-location` IRI;

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 `location` that is neither `http://www.w3.org/2007/rif-import-location#consumer-side-data` nor an IRI that identifies an XML document. And this specification does not prescribe the behaviour of a conformant implementation when an `Import` directive contains a `profile` that is neither `http://www.w3.org/2007/rif-import-profile#xml-data` nor an IRI that identifies an XML schema.

Example 3.1. 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#consumer-side-data http://example.org/customertable.xsd)`
4. `Import(http://www.w3.org/2007/rif-import-location#consumer-side-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 the first and second constraints: the dummy location `consumer-side-data`, that indicates that the profile is to be applied to consumer-side data, is incompatible with the profile `xml-data`. Therefore, the directive is out of the scope of this specification.

3.2 Interpretation of Imports

For the purpose of interpreting the combination of a RIF document, *R*, and XML data, let *dataLocation(R)* denote the set of the URIs in the `location` sub-elements of the `import` directives, in *R* and in the RIF documents that are imported, directly or indirectly, in *R*, whose `profile` is either `rif:xml-data` or identifies an XML schema; and let *dataDocuments(R)* denote the set of the document [nodes][XDM] in the instances of the data model that encapsulate the information in all the XML documents that are identified by an URI in *dataLocation(R)*. If *dataLocation(R)* contains the dummy URI `http://www.w3.org/2007/rif-import-location#consumer-side-data`, *dataDocuments(R)* includes, also, the document [node][XDM] in an instance of the data model that encapsulates the consumer-side data. In the absence of consumer-side data, the document node bears no information.

If an `Import` directive identifies an XML schema as its `profile`, the corresponding instance of the data model includes the information from the PSVI that results from validating the XML data referenced in the `location` against that schema. Otherwise, the instance of the data model is built from the infoset.

If an `Import` directive has the value: `http://www.w3.org/2007/rif-import-location#consumer-side-data` as a locator, the instance of the data model that encapsulates the consumer-side data, if any, includes the information from the PSVI that results from validating the data against the XML schema identified in the profile of that directive.

As a consequence, if different `Import` directives associate different XML schemas to the consumer-side data, the instance of the data model that encapsulates the consumer-side data, if any, includes the information

from the PSVI that results from validating the data against the XML schema that results from importing or including, into an otherwise empty schema, all the XML schemas identified in the `profile` sub-elements of `Import` directives with the value: `http://www.w3.org/2007/rif-import-location#consumer-side-data` as a locator.

Given a RIF document, *R*, the set of data model nodes, *E*, in the RIF+XML data combination *<R, E, S>* is the set of all the document nodes in *dataDocuments(R)*, and of all the data model nodes that can be reached from those document nodes.

In the same way, *S*, the schema definitions component in the RIF+XML data combination *<R, E, S>*, includes all the top level schema definitions found in the XML schemas that are imported as the `profile` of an `import` directive in *R* or in a RIF document that is imported, directly or indirectly, in *R*.

For the purpose of evaluating XPath 2.0 expressions, when interpreting a RIF+XML data combination *<R, E, S>*, the [in-scope schema definitions]^{XP} include all the schema definitions in *S*, all the schema type, schema element and schema attribute definitions that are components of the top level definitions in *S*, and all the schema types defined in [XSD 1.1 Part2].

Example 3.2. Continuing example 3.1, above, 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, since directive #2 supersedes directive #1. Including effectively useless directives should be avoided.

Combining directive #2 and #3 says that the rules in the importing document are meant to be applied only to data that validates against the XML schema in example 2.1, where the data is imported or from the consumer-side.

4 Conformance

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

5 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 Editor's Draft, 22 June 2010, http://www.w3.org/2005/rules/wg/draft/ED-rif-core-20100622/. Latest version available at http://www.w3.org/2005/rules/wg/draft/rif-core/.

[RIF-BLD]
RIF Basic Logic Dialect Harold Boley, Michael Kifer, eds. W3C Editor's Draft, 22 June 2010, http://www.w3.org/2005/rules/wg/draft/ED-rif-bld-20100622/. Latest version available at http://www.w3.org/2005/rules/wg/draft/rif-bld/.

[RIF-DTB]
RIF Datatypes and Built-Ins 1.0 Axel Polleres, Harold Boley, Michael Kifer, eds. W3C Editor's Draft, 22 June 2010, http://www.w3.org/2005/rules/wg/draft/ED-rif-dtb-20100622/. Latest version available at http://www.w3.org/2005/rules/wg/draft/rif-dtb/.

[RIF-PRD]
RIF Production Rule Dialect Christian de Sainte Marie, Gary Hallmark, Adrian Paschke, eds. W3C Editor's Draft, 22 June 2010, http://www.w3.org/2005/rules/wg/draft/ED-rif-prd-20100622/. Latest version available at http://www.w3.org/2005/rules/wg/draft/rif-prd/.

[RIF-RDF-OWL]
RIF RDF and OWL Compatibility Jos de Bruijn, editor. W3C Editor's Draft, 22 June 2010, http://www.w3.org/2005/rules/wg/draft/ED-rif-rdf-owl-20100622/. Latest version available at http://www.w3.org/2005/rules/wg/draft/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/.

[XFO]
XQuery 1.0 and XPath 2.0 Functions and Operators, Ashok Malhotra, Jim Melton, and Norman Walsh, Editors. World Wide Web Consortium, 23 Jan 2007. This version is http://www.w3.org/TR/2007/REC-xpath-functions-20070123/. The latest version is available at http://www.w3.org/TR/xpath-functions/.

[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 1]
W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, Editors. W3C Working Draft, 3 December 2009. This version is http://www.w3.org/TR/2009/WD-xmlschema11-1-20091203/. The latest version is available at http://www.w3.org/TR/xmlschema11-1/.

[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/.

[XSD-CD]
W3C XML Schema Definition Language (XSD): Component Designators, , Editors. W3C Candidate Recommandation 19 January 2010. W3C Recommendation 23 Jan 2007. This version is

http://www.w3.org/TR/2010/CR-xmlschema-ref-20100119/. The latest version is available at http://www.w3.org/TR/xmlschema-ref/.

6 Appendix A: Glossary (non-normative)

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

component-kind() ***(from [XSD-CD], section 4.5.1)***
The component-kind accessor returns an expanded name identifying the kind of the schema component.

component-name() ***(from [XSD-CD], section 4.5.2)***
The component-name accessor returns zero or one xs:QName values giving the name of the component.

Context node ***(from [XPath 2.0], section 2.1.1)***
The context item is the item currently being processed. An item is either an atomic value or a node. When the context item is a node, it can also be referred to as the context node.

Declared type ***(from [XDM], section 3.3.1.1)***
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.

derived-from ***(from [XPath 2.0], section 2.5.4)***
[The] pseudo-function named *derives-from(AT, ET)* [...] takes an actual simple or complex schema type *AT* and an expected simple or complex schema type *ET*, and either returns a boolean value or raises [an error].

- *derives-from(AT, ET)* returns *true* if *ET* is a known type and any of the following three conditions is true:
    1. *AT* is a schema type found in the [in-scope schema definitions][XP], and is the same as *ET* or is derived by restriction or extension from *ET*

2. *AT* is a schema type not found in the [in-scope schema definitions][XP], and an implementation-dependent mechanism is able to determine that *AT* is derived by restriction from *ET*
3. There exists some schema type *IT* such that *derives-from(IT, ET)* and *derives-from(AT, IT)* are *true*.

- *derives-from(AT, ET)* returns *false* if *ET* is a known type and either the first and third or the second and third of the following conditions are true:
  1. *AT* is a schema type found in the [in-scope schema definitions][XP], and is not the same as *ET*, and is not derived by restriction or extension from *ET*
  2. *AT* is a schema type not found in the [in-scope schema definitions][XP], and an implementation-dependent mechanism is able to determine that *AT* is not derived by restriction from *ET*
  3. No schema type *IT* exists such that *derives-from(IT, ET)* and *derives-from(AT, IT)* are *true*.
- *derives-from(AT, ET)* raises [an error] if:
  1. *ET* is an unknown type, or
  2. *AT* is an unknown type, and the implementation is not able to determine whether *AT* is derived by restriction from *ET*.

Document order *(derived from [XDM], section 2.4)*

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.

`fn:data` *(from [XFO], section 2.4)*

`fn:data` takes a sequence of items and returns a sequence of atomic values.

The result of fn:data is the sequence of atomic values produced by applying the following rules to each argument:

- If the item is an atomic value, it is returned.
- If the item is a [node][XDM]:
  - If the [node][XDM] does not have a [typed value][XDM] an error is raised.
  - Otherwise, `fn:data()` returns the [typed value][XDM] of the [node][XDM].

`fn:string` *(from [XFO], section 2.3)*

`fn:string` returns the value of its arguments represented as a `xs:string`. If no argument is supplied, the [context item][XP] is used as the default argument. The behavior of the function if the argument is omitted is exactly the same as if the [context item][XP] had been passed as the argument.

If the [context item][XP] is undefined, an error is raised.
If the list of arguments is the empty sequence, the zero-length string is returned.

If the argument is a node, the function returns the [string-value][XDM] of the node.
If the argument is an atomic value, then the function returns the value of the argument cast as a `xs:string`.

Governing type definition *(from [XSD 1.1 Part 1], section 3.2.4.2 for an attribute, section 3.3.4.6 for an element)*

The governing type definition of an attribute, in a given schema-validity ·assessment· episode, is the {type definition} of the ·governing attribute declaration·, unless the processor has stipulated another type definition at the start of ·assessment· (see Assessing Schema-Validity (§5.2)), in which case it is the stipulated type definition.
The governing type definition of an element information item E, in a given schema-validity ·assessment· episode, is the first of the following which applies:

1. An ·instance-specified type definition· which ·overrides· a type definition stipulated by the processor (see Assessing Schema-Validity (§5.2)).
2. A type definition stipulated by the processor (see Assessing Schema-Validity (§5.2)).
3. An ·instance-specified type definition· which ·overrides· the ·selected type definition· of E.
4. The ·selected type definition· of E.
5. The value ·absent· if E is ·skipped·.
6. An ·instance-specified type definition· which ·overrides· the ·locally declared type·.
7. The ·locally declared type·.
8. An ·instance-specified type definition·.

If none of these apply, there is no ·governing type definition· (or, in equivalent words, it is ·absent·).

In-scope schema definitions *(from [XPath 2.0], section 2.1.1)*

This is a generic term for all the element declarations, attribute declarations, and schema type definitions that are in scope during processing of an expression.] It includes *[the In-scope schema types, the In-scope element declarations, the substitution groups and the In-scope attribute declarations]*.

Node *(from [XDM], section 2.1)*

(from [XDM], section 2.1) There are seven kinds of Nodes in the data model: document, element, attribute, text, namespace, processing instruction, and comment.
(from [XPath 2.0], section 2) Each node has a unique node identity, a typed value, and a string value. In addition, some nodes have a name. The typed value of a node is a sequence of zero or more atomic values. The string value of a node is a value of type `xs:string`. The name of a node is a value of type `xs:QName`.

Normalized value *(from [XSD 1.1 Part 1], section 3.1.4)*

The normalized value of an element or attribute information item is an initial value which has been normalized according to the value of the whiteSpace facet, and the values of any other pre-lexical facets, associated with the simple type definition used in its validation.
The initial value of an attribute information item is the normalized attribute value, as specified in [XML], section 3.3.3 (Attribute-value normalization). Similarly, the initial value of an element information item is the string composed of, in document order, the character code of each character information item in the $[children]_{info}$ of that element information item.

Schema normalized value *(from [XSD 1.1 Part 1], section 3.2.5.4 for attributes, section 3.3.5.4 for elements)*

If an attribute's normalized value is valid with respect to the governing type definition, then the schema normalized value of the attribute is the normalized value as validated, otherwise it is absent.
If an element information item is not nilled and either the governing type definition is a simple type definition or its {content type} has {variety} *simple*, then the schema normalized value of the element is the lexical form of its value constraint if applicable; else, if the element's initial value is valid with respect to the simple type definition as defined by String Valid ([XSD 1.1 Part 1], section 3.16.4), then the schema normalized value is the normalized value of the item as validated; otherwise it is absent.

Statically-known namespaces *(from [XPath 2.0], section 2.1.1)*

This is a set of (prefix, URI) pairs that define all the namespaces that are known during static processing of a given expression.

String value

The string value of a node is a string and can be extracted by applying the `fn:string` function to the node.
(from [XDM], section 3.3.1.3) Element and attribute nodes have both typed-value and string-value properties. However, implementations are allowed some flexibility in how these properties are stored. An implementation may choose to store the string-value only and derive the typed-value from it, or to store the typed-value only and derive the string-value from it, or to store both the string-value and the typed-value.

In order to permit these various implementation strategies, some variations in the string value of a node are defined as insignificant. Implementations that store only the typed value of a node are permitted to return a string value that is different from the original lexical form of the node content.

type-axis() ***(from [XSD-CD], section 4.4.5)***

The type axis contains simple type definition and complex type definition components linked to the current component through {type definition}, {type definitions}, {content type}, or {simple type definition} arcs

Typed value

The typed value of a node is a sequence of atomic values and can be extracted by applying the `fn:data` function to the node. *(from [XDM], section 3.3.1.2)* 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. [...] 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. 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.

7 Appendix B: Embedding imported data sources as RIF facts (non-normative)

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

8 Appendix C: Examples using the normative RIF/ XML syntax

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

9 Appendix D: Change Log (non-normative)

- Complete rework of the semantics

Since the WD published 22 June 2010:

- Completed the semantics
- Replaced the ad hoc syntax with XPath 2.0 and XML schema component designator syntaxes