Resource Description Framework (RDF) Model and Syntax

W3C Working Draft 16 Feb 1998

This Version:
http://www.w3.org/TR/1998/WD-rdf-syntax-19980216
Latest Version:
http://www.w3.org/TR/WD-rdf-syntax
Previous Version:
http://www.w3.org/TR/WD-rdf-syntax-971002
Editors:
Ora Lassila <ora.lassila@research.nokia.com>, Nokia Research Center
Ralph R. Swick <swick@w3.org>, World Wide Web Consortium
Acknowledgements

Status of This Document

This draft specification is a work in progress representing the current consensus of the W3C RDF Model and Syntax Working Group. This is a W3C Working Draft for review by W3C members and other interested parties. Publication as a working draft does not imply endorsement by the W3C membership. While the design has stabilized quite a bit since the previous version and we do not anticipate substantial changes, we still caution that further changes are possible and therefore we recommend that only experimental software or software that can be easily field-upgraded be implemented to this specification at this time. The RDF Model and Syntax Working Group will not allow early implementation to constrain their ability to make changes to this specification prior to final release. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite W3C Working Drafts as other than "work in progress".

Note: As working drafts are subject to frequent change, you are advised to reference the above URL for "Latest version" rather than the URLs for specific working draft versions themselves. The latest version URL will always point to the most current version of this draft.

Comments on this specification may be sent to www-rdf-comments@w3.org. The archive of public comments is available at http://www.w3.org/Archives/Public/www-rdf-comments.

Table of Contents

---

1. Introduction

RDF - the Resource Description Framework - is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. RDF metadata can be used in a variety of application areas; for example: in *resource discovery* to provide better search engine capabilities; in *cataloging* for describing the content and content relationships available at a particular Web site, page, or digital library; by *intelligent software agents* to facilitate knowledge sharing and exchange; in *content rating*; in describing *collections* of pages that represent a single logical "document"; for describing *intellectual property rights* of Web pages, and in many others. RDF with *digital signatures* will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications.

Metadata is "data about data" or specifically in the context of RDF "data describing web resources". The distinction between "data" and "metadata" is not an absolute one; it is a distinction created primarily by a particular application. Many times the same resource will be interpreted in both ways simultaneously. RDF encourages this view by using XML as the encoding syntax for the metadata. The resources being described by RDF are, in general, anything that can be named via a URI. The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines the semantics of any application domain. The definition of the mechanism should be domain neutral, yet the mechanism should be suitable for describing information about any domain.

This document introduces a model for representing RDF metadata and one syntax for expressing and transporting this metadata in a manner that maximizes the interoperability of independently developed web servers and clients. The syntax described in this document is best considered as a "serialization syntax" for the underlying RDF representation model. The serialization syntax uses the eXtensible Markup Language [XML]. RDF and XML are complementary; there will be alternate ways to represent the same RDF data model, some more suitable for direct human authoring. Future work may lead to including such alternatives in this document.

2. RDF Data Model

At the core of RDF is a model for representing named properties and their values. These properties serve both to represent attributes of resources (and in this sense correspond to usual attribute-value pairs) and to represent relationships between resources. The RDF data model is a syntax-independent way of representing RDF expressions. This specification shows three representations of the data model; as 3-tuples, as a graph, and in XML. These representations have equivalent meaning. The mapping between the representations used in this specification is not intended to constrain in any way the internal representation used by implementations.

RDF expressions that are syntactically very different could mean the same thing. This concept of equivalence in meaning is very important when performing queries, aggregation and a number of other tasks at which RDF is aimed. The equivalence is defined in a clean machine understandable way. Two RDF expressions are equivalent if and only if their corresponding data model representations are the same. Two RDF string values (defined below) are deemed to be the same if their ISO/IEC 10646 representations match.

2.1. RDF Core; "Layer 0"

The core data model is precisely defined as:

1. There is a set called *Nodes*.
2. There is a subset of *Nodes* called *PropertyTypes*.
3. There is a set of 3-tuples called *Triples*, whose elements are called *properties*. The first item of each 3-tuple (triple) is an element of *PropertyTypes*, the second item is an element of *Nodes* and the third item is either an element of *Nodes* or an atomic value (an RDF string).

In this data model both the resources being described and the values describing them are represented as nodes in a directed labeled graph. The arcs connecting pairs of nodes correspond to the names of the property types. Each arc is said to be *labeled* by the corresponding property type.

As RDF serialization is written in XML, the character repertoire of an RDF string is ISO/IEC 10646 [ISO10646]. An actual RDF string, whether in an XML document or in some other representation of the RDF data model, may be stored using a direct encoding of ISO/IEC 10646 or an encoding which can be mapped to ISO/IEC 10646. Language tagging is part of the string value; it is applied to sequences of characters within an RDF string and does not have an explicit manifestation in the data model.

Two RDF strings are deemed to be the same if their ISO/IEC 10646 representations match. Each RDF application must specify which one of the following definitions of 'match' it uses:

a. the two representations are identical, or
b. the two representations are canonically equivalent as defined by The Unicode Standard [Unicode].

The application may ignore language tagging within a string; all RDF applications must specify whether or not language tagging in string values is significant; that is, whether or not language is considered when performing string matching or other processing.

An RDF expression is represented pictorially in text with nodes in '[...]' and arcs in '--...-->' as follows:

```
[resource R] ---PropertyType P--> [value V]
```

This is read "V is the value of the property type P for resource R", or left-to-right; "R has property type P with value V".

Consider as a simple example the statement:

> *Ora Lassila is the author of the resource http://www.w3.org/People/Lassila*

This statement can be represented as follows:

```
[http://www.w3.org/People/Lassila] ---Author--> "Ora Lassila"
```

where the notation '[URI]' denotes the node representing the resource identified by URI and quotation marks (") denote an atomic value.

According to the formal definition, this property, i.e. the arc labeled "Author" plus its source and target nodes, is the triple:

```
{Author, [http://www.w3.org/People/Lassila], "Ora Lassila"}
```

where "Author" denotes a node in *PropertyTypes* that corresponds to this arc label.

A set of triples that all refer to the same node (the second item in the triple) can be grouped as a unit called a *Description*. Descriptions are particularly useful when specifying a number of properties of the same resource. A Description (here containing two properties) is diagrammed as follows:

```
[resource R] ---PropertyType P1--> [value Vp1]
            |
             ---PropertyType P2--> [value Vp2]

[Bag B] --------RDF:1------------> [Property P1Vp1]
       |
        --------RDF:2------------> [Property P2Vp2]
```

Each RDF Description has its own node (the node identified as "Bag B" in the diagram) identifying the collection of triples, as described below. The Description can therefore be described by properties; that is, a Description can itself be used as the source node of other arcs describing properties of the Description.

The property types in a given Description, as well as any characteristics or restrictions of the property values themselves, are defined by one or more *schemas.* Each property type used in a Description is declared to be from exactly one schema. A schema is identified by a URI. The schema URI may be treated merely as an identifier or it may refer to a machine-understandable description of the schema. By definition, to *understand* a particular schema used by a Description means to understand the semantics of each of the properties in that Description that are declared to be associated with the schema. An application that has no knowledge of the particular schema will still be able to parse the Description into the property type and property value components and will be able to transport the Description intact (e.g. to a cache or to another application).

2.2. Utility Relations; "Layer 1"

The utility relations exist to permit Descriptions about Descriptions and to provide definitions for collections of nodes.

The triple composed of a property type, a resource, and a value is an RDF *property*. Such a property can itself be the target node of an arc (i.e. the value of some other property) or the source node of an arc (i.e. it can have properties). In these cases, the original property must be *reified*; that is, converted into additional nodes and arcs. The term *reification* means to convert the relation expressed by the arc into a concrete node to which we can refer. Reification allows us to express modalities (e.g. beliefs about properties) or simply attach any properties to other properties.

For example, the reification of the expression

```
[http://www.w3.org/People/Lassila] ---Author--> "Ora Lassila"
```

is the four properties

```
[X] ---InstanceOf--> [RDF:Property]
[X] ---PropName----> [Author]
[X] ---PropObj-----> [http://www.w3.org/People/Lassila]
[X] ---Value-------> "Ora Lassila"
```

That is, a node X and four new triples are added:

```
{InstanceOf, [X], [RDF:Property]}
{PropName,   [X], [Author]}
{PropObj,    [X], [http://www.w3.org/People/Lassila]}
{Value,      [X], "Ora Lassila"}
```

Reified properties are drawn as a single node with several arcs emanating from it representing the resource, property name, and value:

```
[property P1] ----InstanceOf-> [RDF:Property]
             |
              ----PropName---> [PropertyType P]
             |
              ----PropObj----> [resource R]
             |
              ----Value------> [value Vp1]
```

This allows RDF to be used to make statements about other statements; for example, the statement "Ralph believes that the document 'The Origin of Species' was authored by Charles Darwin" is represented as:

```
[Ralph] --Believes--> [statement1] --InstanceOf-> [RDF:Property]
                                  |
                                   --PropName---> [Author]
                                  |
                                   --PropObj----> [http://loc.gov/Books/Species]
                                  |
                                   --Value------> "Charles Darwin"
```

To help in reifying properties, RDF defines the *InstanceOf* relation (property type) to provide primitive typing, as shown in the previous example.

The formal definition of *InstanceOf* is:

4. There is an element of *PropertyTypes* known as RDF:InstanceOf.
5. Elements of *Triples* that have RDF:InstanceOf as the first item in the triple must have elements of *Nodes* as their second item and third item.

The formal definition of reification is:

6. There is an element of *Nodes*, not contained in *PropertyTypes*, known as RDF:Property.
7. There are three elements in *PropertyTypes* known as RDF:PropName, RDF:PropObj, and RDF:Value.
8. The reification of an element *t* of *Triples* is an element *n* of *Nodes* representing the reified *t* and the elements t_1, t_2, t_3, and t_4 of *Triples* where:
   a. the first item of t_1 is RDF:PropName, the second item is *n*, and the third item is the first item of *t*,
   b. the first item of t_2 is RDF:PropObj, the second item is *n*, and the third item is the second item of *t*,
   c. the first item of t_3 is RDF:Value, the second item is *n*, and the third item is the third item of *t*,
   d. the first item of t_4 is RDF:InstanceOf, the second item is *n*, and the third item is RDF:Property.

To reify a property, all that is done is to add to the data model an additional node and the four triples with first items (labeled arcs) of RDF:PropName, RDF:PropObj, RDF:Value, and RDF:InstanceOf respectively, second item the additional node, and third item the corresponding property type, resource node, value node, and the RDF:Property node respectively. Note the difference between the node RDF:Property and *properties*. RDF:Property is the node defining a particular class of objects. The instances of that class are the properties; i.e. triples.

Frequently it is necessary to create a collection of nodes; for example to state that a property has an ordered sequence of values. RDF defines three kinds of collections: ordered lists of nodes, called *Sequences*, unordered lists of nodes, called *Bags*, and lists that represent alternatives for the (single) value of a property, called *Alternatives*.

Formally, these three collection types are defined by:

9. There are three elements of *Nodes*, not contained in *PropertyTypes*, known as RDF:Seq, RDF:Bag, and RDF:Alt.
10. There is a subset of *PropertyTypes* corresponding to the ordinals (1, 2, 3, ...) called Ord. We refer to elements of Ord as RDF:1, RDF:2, RDF:3, ...

To create a collection of nodes, create a new node with an RDF:InstanceOf arc pointing to one of the three node types RDF:Seq, RDF:Bag, or RDF:Alt. The remaining arcs from that new node point to each of the members of the collection and are labeled using the elements from Ord. There may be at most one arc from a single collection node labeled with each element of Ord and the elements of Ord must be used in sequence starting with RDF:1. For nodes that are instances of the RDF:Alt collection type, there must be exactly one member whose arc label is RDF:1 and that is the default value for the Alternatives node.

3. Transporting RDF

Descriptions may be associated with the resource they describe in one of four ways:

1. The Description may be contained within the resource ("embedded"; e.g. in HTML).
2. The Description may be external to the resource but supplied by the transfer mechanism in the same retrieval transaction as that which returns the resource ("along-with"; e.g. via HTTP).
3. The Description may be retrieved independently from the resource, including from a different source ("service bureau"; e.g. using HTTP).
4. The Description may contain the resource ("wrapped"; e.g. RDF itself).

All resources will not support all association methods; in particular, many kinds of resources will not support embedding and only certain kinds of resources may be wrapped.

A human- or machine-understandable description of an RDF schema may be accessed through content negotiation by dereferencing the schema URI. If the schema is machine-understandable it may be possible for an application to learn some of the semantics of the property types named in the schema on demand. The logic and syntax of RDF schemas are described in a separate document; "RDF Machine Understandable Schemas" (in progress).

This specification does not define any schemas; all references in this document to specific schema URIs and associated property types are intended as illustration only.

4. RDF Grammar

The RDF data model provides an abstract, conceptual framework for defining and using metadata. A concrete syntax is also needed for the purposes of authoring and exchanging this metadata. The syntax does not add to the model; APIs may be provided to manipulate RDF without reference to a concrete syntax. RDF uses the Extensible Markup Language [XML] encoding as its syntax. However, RDF will not require (and conforming implementations must not require) an XML Document Type Declaration for the contents of Descriptions. In this respect RDF requires at most the XML well-formedness constraints. Schemas used by RDF Descriptions can be XML DTDs however other schema languages are expected to be more common for use with RDF.

The syntax descriptions below use the Extended Backus-Naur Form notation as defined in section 6, Notation, of [XML] to describe the essential RDF syntax elements. The EBNF here is condensed for human readability; in particular, the italicized "*`rdf`*" is used to represent a variable namespace name rather than the more precise BNF notation "`'<' NSname ':...'`" and the requirement that the property and type names in end-tags exactly match the names in the corresponding start-tags is implied by the XML rules. As RDF is written using XML, all syntactic flexibilities of XML are also implicitly included; e.g. whitespace rules, quoting using either single quote (') or double quote ("), character escaping, case sensitivity, language tagging, etc. RDF requires the XML namespace facility, currently under review by the XML Working Group, to associate the schema, by way of its URI, with the properties in the Description.

This specification defines two syntaxes for encoding an RDF data model instance. The *serialization syntax* expresses the full capabilities of the data model in a very regular fashion. The *abbreviated syntax* includes additional constructs that provide a more compact form to represent a subset of the data model. Conformant RDF interpreters are expected to implement both the full serialization syntax and the abbreviated syntax.

4.1 Serialization Syntax

RDF serialization syntax takes the form:

```
  [1] RDF            ::= '<rdf:RDF>' expression* '</rdf:RDF>'
  [2] expression     ::= description | collection
  [3] description    ::= '<rdf:Description' idRefAttr? bagIdAttr? '>'
                              property* '</rdf:Description>'
  [4] collection     ::= sequence | bag | alternatives
  [5] sequence       ::= '<rdf:Seq' idAttr? '>' collMember* '</rdf:Seq>'
  [6] bag            ::= '<rdf:Bag' idAttr? '>' collMember* '</rdf:Bag>
  [7] alternatives   ::= '<rdf:Alt' idAttr? '>' collMember+ '</rdf:Alt>'
  [8] collMember     ::= hrefItem | valueItem
  [9] hrefItem       ::= '<rdf:LI' hrefAttr '/>'
 [10] valueItem      ::= '<rdf:LI>' value '</rdf:LI>'
 [11] idRefAttr      ::= idAttr | hrefAttr
 [12] hrefAttr       ::= 'rdf:HREF="' resourceURI '"'
 [13] idAttr         ::= 'ID="' IDsymbol '"'
 [14] bagIdAttr      ::= 'rdf:BAGID="' IDsymbol '"'
 [15] resourceURI    ::= (see RFC1738)
 [16] IDsymbol       ::= (any legal XML name symbol)
```

The RDF element is a simple wrapper that marks the boundaries in an XML document where the content is explicitly intended to be mappable into an RDF data model instance.

Description contains the remaining elements that cause the creation of properties in the model instance. Each XML element *E* contained by a Description results in the creation of a property (a triple that is an element of the formal set *Triples*), where:

1. The first item (PropName) is the node that represents the resource whose URI is the expansion of the namespace-qualified Generic Identifier (tag name) of *E*.
2. The second item (PropObj) is
   - the node that represents the resource whose URI is given by the value of the HREF attribute of the Description or
   - a new node representing a new resource whose identification is the value of the ID attribute of the Description, if present; else the new resource has no identification.
3. If *E* is an empty element (no content), the third item (Value) is the node that represents the resource whose URI is given by the HREF attribute of *E*. If the content of *E* contains no XML markup, the Value is the content of *E* (a string). Otherwise, the content of *E* must be another Description or collection and the Value is the node named by the (possibly implicit) ID or HREF of that Description or collection.

The Description element itself creates an instance of a Bag node. The members of this Bag are the nodes corresponding to the reification of each of the properties in the Description. If the BAGID attribute is specified its value is the identifier of this Bag, else the Bag is anonymous.

When HREF is specified with Description, the properties in the Description refer to the resource named in the HREF. A Description element without an HREF attribute creates an in-line resource. Typically such a resource will be a surrogate, or proxy, for some other real resource that does not have a recognizable URI. This in-line resource has a URI formed using the value of the ID attribute of the Description element, if present. When another Description or property value refers to the in-line resource it will use the value of the ID in an HREF attribute. When the other Description refers to the bag of nodes corresponding to the reified properties it will use the value of BAGID in an HREF attribute. Either ID or HREF may be specified on Description but not both together in the same element. The values for each ID and BAGID attribute must not appear more than once within a document nor may the same value be used in an ID and a BAGID; that is, these two attributes have XML AttType ID.

Seq, Bag, and Alt create an instance of a sequence, bag, or alternatives collection node type respectively. This node is the PropObj of an InstanceOf property pointing to the corresponding collection node type. The members of the collection are denoted by LI. Each XML element *E* contained by an LI corresponds to one member of the collection and results in the creation of a tuple where:

1. The first item (PropName) is assigned consecutively according to the (XML) order of lexical appearance of each member starting with "RDF:1" for each collection.
2. The second item (PropObj) is a node *C* that represents the collection created by the Seq, Bag, or Alt. The ID attribute, if specified, provides the URI fragment identifier for *C*.
3. (same as rule 3 above) If *E* is an empty element (no content), the third item (Value) is the node that represents the resource whose URI is given by the HREF attribute of *E*. If the content of *E* contains no XML markup, the Value is content of *E* (a string). Otherwise, the content of *E* must be another Description or collection and the Value is the node named by the(possibly implicit) ID or HREF of that Description or collection.

The resourceURI identifies the target resource; i.e. the resource to which the Description applies or the resource that is included in the collection. The BAGID attribute on a Description element and the ID attribute on a collection element permit that Description or collection to be referred to by other Descriptions. The ID on a collection element also provides the name that is used in an HREF attribute on a property element to make the collection the value of that property.

An RDF property takes the form:

```
[17] property       ::= '<' propName idAttr? '>' value '</' propName '>'
                      | '<' propName idRefAttr '/>'
[18] propName       ::= 'rdf:Value' | 'rdf:InstanceOf' | name
                      | NSname ':' name
[19] value          ::= expression | string
[20] name           ::= (any legal XML name symbol)
[21] NSname         ::= (any legal XML namespace prefix)
[22] string         ::= (any XML text, with "<", ">", and "&" escaped)
```

Within *property*, the resourceURI used in an HREF attribute identifies the resource that is the value of this property. The value of the ID attribute, if specified, is the identifier for the node that represents the reification of the property. If an expression is specified as a property value the value is the node given by the HREF of the Description or the (possibly implied) ID of the Description or collection. The propName Value may be used for expressing non-binary relations; see example 5.3. `string`s must be well-formed XML; the usual XML content quoting and escaping mechanisms may be used if the string contains character sequences (e.g. "<" and "&") that violate the well-formedness rules or that otherwise might look like markup. The xml:lang attribute may be used as defined by [XML] to associate a language with the property value. When used, xml:lang adds no triples to the data model.

It is recommended that property names always be qualified with a namespace prefix to unambiguously connect the property definition with the corresponding RDF schema.

4.2 Abbreviated Syntax

The RDF abbreviated syntax can be employed for each property in a Description where:

1. The given property name appears at most once in the Description and
2. The property value is a string.

The abbreviated syntax can also be employed for each property where:

1. The property value is a node and
2. The values in any properties specified for that value node are strings.

In addition, the abbreviated syntax provides a shorthand for specifying a node with an InstanceOf property.

The abbreviated syntax replaces productions [3] and [17] of the grammar for the serialization syntax in the following manner:

```
  [3a] description    ::= '<rdf:Description' idRefAttr? bagIdAttr? propAttr* '/>'
                        | '<rdf:Description' idRefAttr? bagIdAttr? propAttr* '>'
                              property* '</rdf:Description>'
                        | typedNode
 [17a] property       ::= '<' propName idAttr? '>' value '</' propName '>'
                        | '<' propName idRefAttr? bagIDAttr? propAttr* '/>'
  [23] typedNode      ::= '<' typeName idRefAttr? bagIDAttr? propAttr* '/>'
                        | '<' typeName idRefAttr? bagIDAttr? propAttr* '>'
                              property* '</' typeName '>'
  [24] typeName       ::= NSname ':' name
  [25] propAttr       ::= propName '="' string '"'
                          (with embedded quotes escaped)
```

Each property type and value expressed in XML attribute form by productions [3a] and [25] is equivalent to the same property type and value expressed as XML content of the corresponding Description according to production [17]. Specifically; each XML attribute *A* specified with a Description start tag other than the attributes ID, HREF, or BAGID results in the creation of a property (a triple), where:

1. The PropName (first item) is the node that represents the resource whose URI is the expansion of the namespace-qualified attribute name of *A*.
2. The PropObj (second item) is the node that represents the resource whose URI is given by the value of the HREF attribute or whose identifier is given by the value of the ID attribute of the Description.
3. The Value (third item) is the attribute value of *A* (a string).

The typedNode form may be used to create or declare instances of nodes of specific types and to further describe those nodes. A Description expressed in typedNode form by production [23] is equivalent to the same Description expressed by production [3a] with the same ID, BAGID, and HREF attributes plus an additional InstanceOf property in the Description where the value of the InstanceOf property is the fully expanded URI corresponding to the typeName of the typedNode. Specifically, when a typedNode is given, each XML attribute *A* other than ID, HREF, or BAGID and each property *P* result in the creation of a triple where:

1. The first item (PropName) is the node that represents the resource whose URI is the expansion of the namespace-qualified attribute name of *A* or the property name of *P*.
2. The second item (PropObj) is the node *n* that represents the resource whose URI is given by the value of the HREF attribute or whose identifier is given by the value of the ID attribute of the typedNode element.
3. The third item (Value) is the attribute value of *A* (a string) or the value of *P*.

plus an additional triple where the PropObj is the same node *n*, the PropName is RDF:InstanceOf, and the Value is the node represented by the typeName in the typedNode. If the typedNode is given as the value of a property, the node *n* is the Value of that property in which the typedNode was specified.

Property types and values expressed in XML attribute form by productions [17a] and [25] are equivalent to the same property types and values expressed as XML content of a single Description element *D* and the resource refered to by *D* is the value of the property named by the XML element according to productions [19], [2], and [3]. Specifically; each property start tag containing attribute specifications other than ID, HREF, or BAGID results in the creation of a node *R* where *R* is the Value of the property. If HREF is specified, *R* is the node corresponding to the resource whose URI is the value of the attribute, else if ID is specified *R* is a new node whose identifier is the value of the attribute, otherwise *R* is anonymous. Each XML attribute *A* except ID, HREF, and BAGID specified with that same property start tag results in the creation of a property (a triple), where:

1. The PropObj is the node *R*.
2. The PropName is the node that represents the resource whose URI is the expansion of the namespace-qualified attribute name of *A*.
3. The Value is the attribute value of *A* (a string).

The value of the BAGID attribute, if specified, is the identifier for the Bag corresponding to the Description *D*; else the Bag is anonymous.

5. Signed RDF

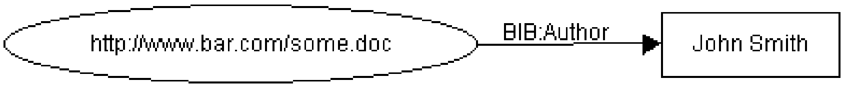
[[*This section to be completed*]]

Descriptions may be *signed* to facilitate decisions that require trust. Simple signatures include checksums or other Descriptions about independently verifiable characteristics of a resource. The simplest example of a signature is a statement that the associated Description applies only to the version of the resource labeled with a given creation date. Stronger signatures will include cryptographic measures to increase the likelihood of detection of falsification of or inadvertent changes to the signed Description or the resource(s) to which it applies.

6. Examples

6.1. Examples Using the RDF Core

With the core defined, we can construct and exchange directed graph models of arbitrary complexity. We could begin by saying very simple things, such as "John Smith is the Author of the document whose URL is http://www.bar.com/some.doc". This description can be modeled with the directed graph:



D

Figure 1: directed graph diagram

(We use a notation where nodes are represented as ellipses, arcs as arrows, and atomic values - i.e. strings - are given in rectangles.)

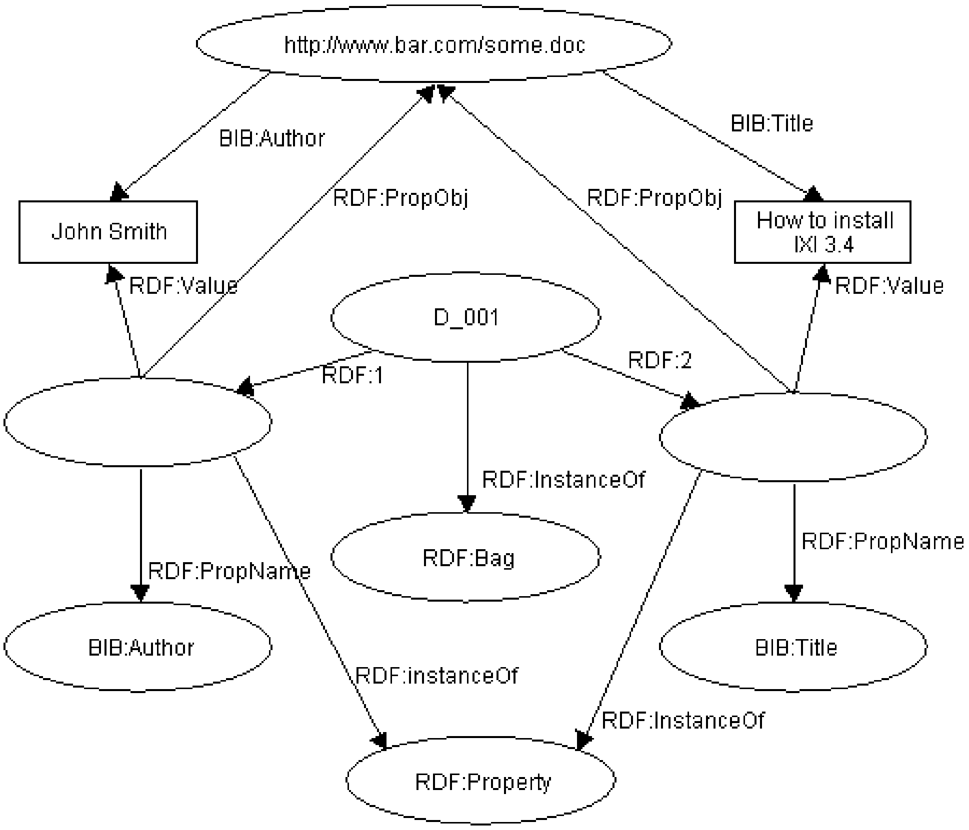
This small graph can be exchanged in the serialization syntax as:

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author>John Smith</BIB:Author>
  </RDF:Description>
</RDF:RDF>
```

In most of the example figures the Bag corresponding to the reified properties in each Description will be omitted from the graph to improve readability. To illustrate the complete graph, consider the following example:

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc" RDF:BAGID="D_001">
    <BIB:Author>John Smith</BIB:Author>
    <BIB:Title>How to install IXI 3.4</BIB:Title>
  </RDF:Description>
</RDF:RDF>
```

As shown in Figure 2, the Description itself becomes a Bag node where the contents of the collection list the individual properties that were part of that particular Description.

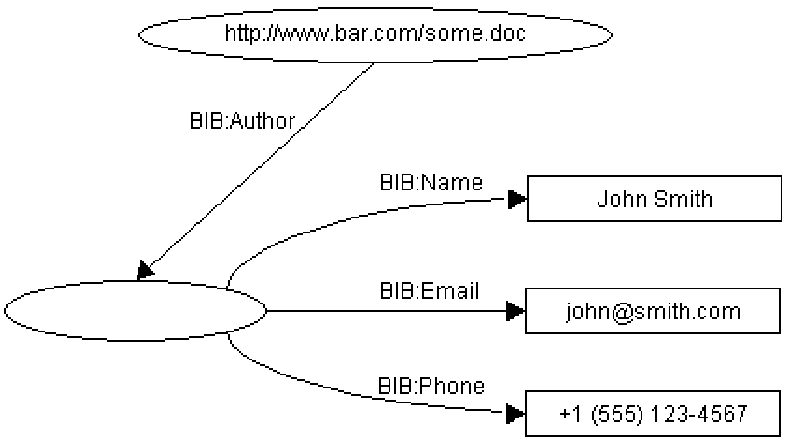


D

Figure 2: Descriptions create Bag nodes

Note that the Description uses the HREF attribute to say that the properties refer to the existing node whose URI is "http://www.bar.com/some.doc".

We could create a more elaborate model in order to say additional things about John Smith, such as his contact information. We might construct the model:



D

Figure 3: Structured property value (Bag omitted)

which could be exchanged using the RDF serialization representation:

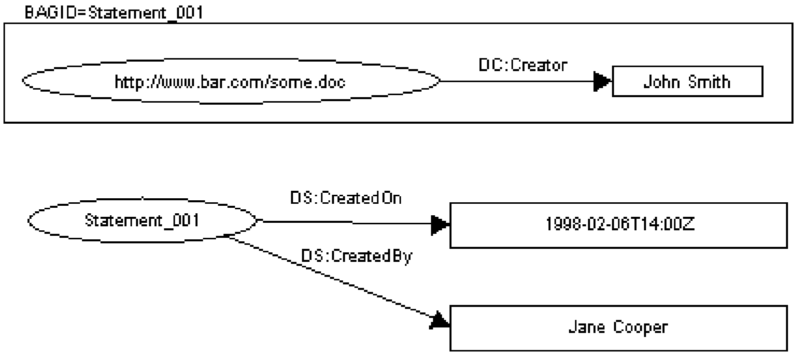
```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author>
      <RDF:Description>
        <BIB:Name>John Smith</BIB:Name>
        <BIB:Email>john@smith.com</BIB:Email>
        <BIB:Phone>+1 (555) 123-4567</BIB:Phone>
      </RDF:Description>
    </BIB:Author>
  </RDF:Description>
</RDF:RDF>
```

Note here that the Description that is the value of the Author property does not have an HREF attribute. Therefore, this Description creates a new (unnamed) node and this new node is the property value of the Author property.

The serialization above provides the same properties as this second serialization, with an ID specified for the anonymous resource:

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author RDF:HREF="#John_Smith"/>
  </RDF:Description>

  <RDF:Description ID="John_Smith">
    <BIB:Name>John Smith</BIB:Name>
    <BIB:Email>john@smith.com</BIB:Email>
    <BIB:Phone>+1 (555) 123-4567</BIB:Phone>
  </RDF:Description>
</RDF:RDF>
```

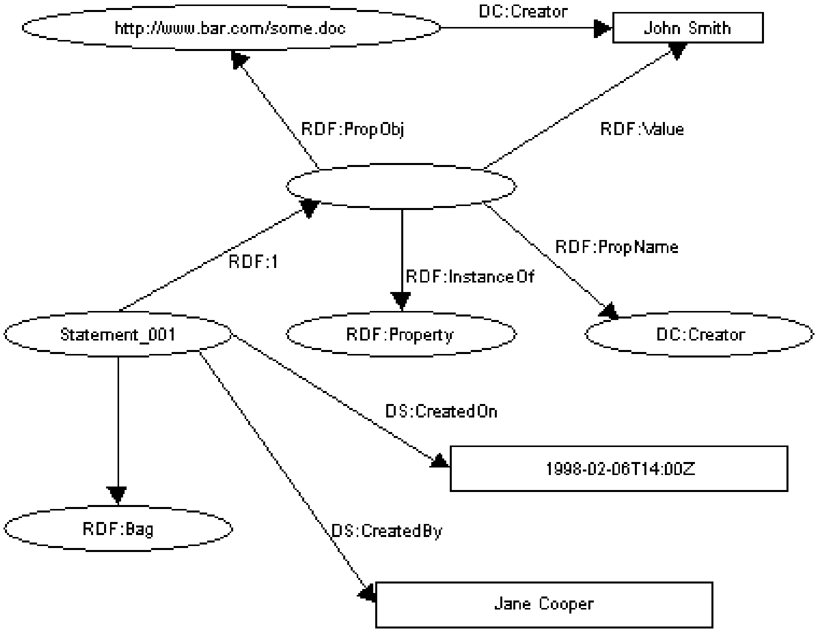
As an example of making a statement about a statement, consider the case of wanting to indicate when a particular description was written and who wrote the description.



D

Figure 4: Descriptions of Descriptions (Bag omitted)

What we want to say in the model is expressed by this pair of graphs; that we have an XML encoding of some description, and that there is some other XML content that makes further statements about that encoding. If we look into the details, the graph is really as shown in Figure 5.



D

Figure 5a: Details of reification for making Descriptions of Descriptions, part 1

This model would be expressed as:

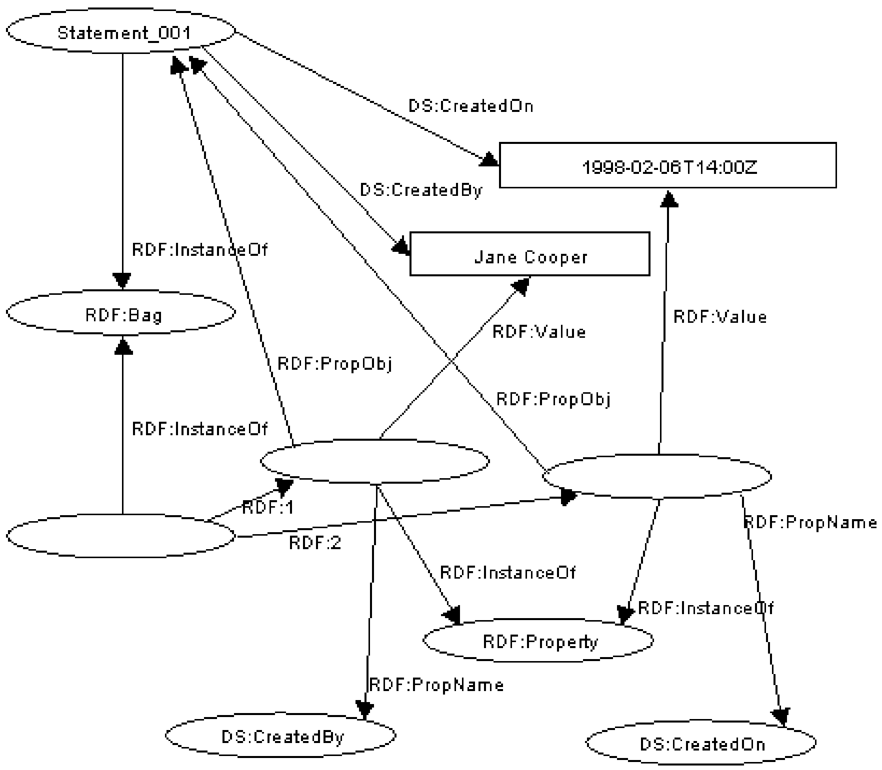
```
<?xml:namespace name="http://purl.org/metadata/dublin_core#" as="DC"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://www.w3.org/Schemas/DS-Schema#" as="DS"?>
```

```
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc"
                   RDF:BAGID="Statement_001">
    <DC:Creator>John Smith</DC:Creator>
  </RDF:Description>

  <RDF:Description RDF:HREF="#Statement_001">
    <DS:CreatedOn>1998-02-06T14:00Z</DS:CreatedOn>
    <DS:CreatedBy>Jane Cooper</DS:CreatedBy>
  </RDF:Description>
</RDF:RDF>
```

Observe that the CreatedOn and CreatedBy properties refer not to "http://www.bar.com/some.doc" but rather to the statement "John Smith is the creator of http://www.bar.com/some.doc". That statement was, according to this graph, made by Jane Cooper.

Also note that node labels such as "RDF:Property" are shorthand for a full URI such as "http://www.w3.org/TR/WD-rdf-syntax#Property".

Figure 5a omits the Bag corresponding to the second Description element; that Bag is shown in Figure 5b

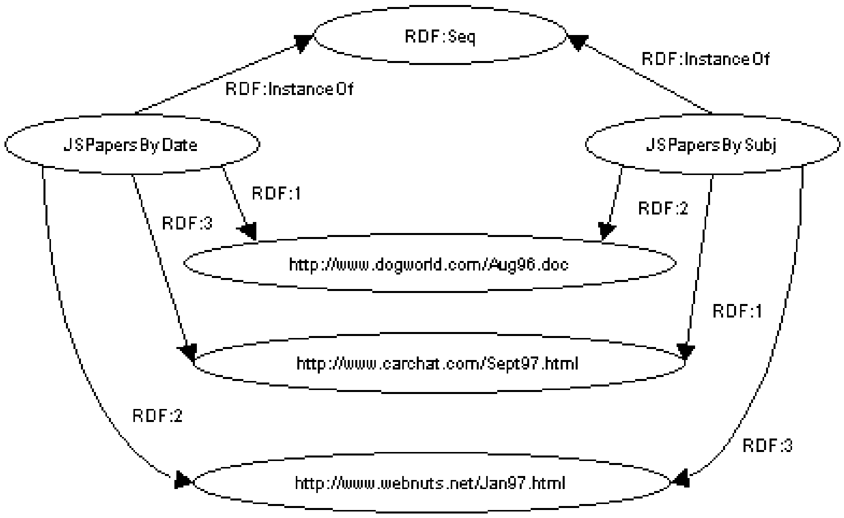


D

Figure 5b: Details of reification for making Descriptions of Descriptions, part 2

6.2. Sequence Examples

As an example of Sequences, we might look at some of the works written by John Smith. Since John is rather prolific, we could use Sequences to keep lists of his works sorted by publication date, or according to the alphabetical order of the subject of the article:



D

Figure 6: Sequences

This model could be exchanged as:

```
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Seq ID="JSPapersByDate">
    <RDF:LI RDF:HREF="http://www.dogworld.com/Aug96.doc"/>
    <RDF:LI RDF:HREF="http://www.webnuts.net/Jan97.html"/>
    <RDF:LI RDF:HREF="http://www.carchat.com/Sept97.html"/>
  </RDF:Seq>

  <RDF:Seq ID="JSPapersBySubj">
    <RDF:LI RDF:HREF="http://www.carchat.com/Sept97.html"/>
    <RDF:LI RDF:HREF="http://www.dogworld.com/Aug96./doc"/>
    <RDF:LI RDF:HREF="http://www.webnuts.net/Jan97.html"/>
  </RDF:Seq>
</RDF:RDF>
```

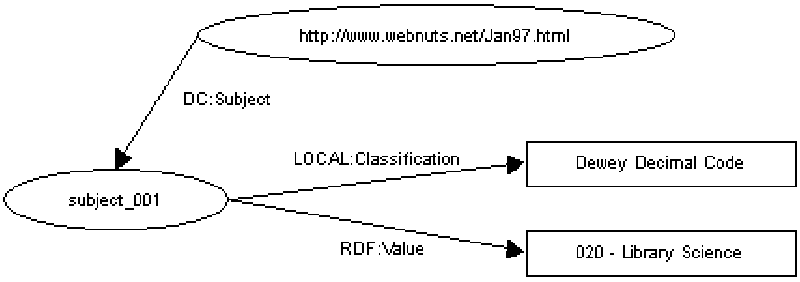
To further illustrate aggregates, consider an example of a document with two authors specified alphabetically, a title specified in two different languages, and having two equivalent locations on the Web:

```
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://purl.org/metadata/dublin_core#" as="DC"?>

<RDF:RDF>
   <RDF:Description RDF:HREF="http://www.foo.com/cool.html">
```

```
      <DC:Creator>
         <RDF:Seq ID="CreatorsAlphabeticalBySurname">
           <RDF:LI>Mary Andrew</RDF:LI>
           <RDF:LI>Jacky Crystal</RDF:LI>
         </RDF:Seq>
      </DC:Creator>

      <DC:Identifier>
         <RDF:Bag ID="MirroredSites">
           <RDF:LI RDF:HREF="http://www.foo.com.au/cool.html"/>
           <RDF:LI RDF:HREF="http://www.foo.com.it/cool.html"/>
         </RDF:bag>
      </DC:Identifier>

      <DC:Title>
         <RDF:Alt>
            <RDF:LI xml:lang="en">The Coolest Web Page</RDF:LI>
            <RDF:LI xml:lang="it">Il Pagio di Web Fuba</RDF:LI>
         </RDF:Alt>
      </DC:Title>

    </RDF:Description>
 </RDF:RDF>
```

6.3. Using Binary Relations to Represent Relations of Higher Arity

The RDF data model intrinsically only supports binary relations. However, in this section we show how we can represent higher arity relations using just binary relations. As an example, consider the subject of one of John Smith's recent articles - library science. We could use the Dewey Decimal Code for library science to categorize that article. Dewey Decimal codes are far from the only subject categorization scheme. So we might want to define a "Subject" node that not only told us the subject of a paper but also indicated the categorization scheme it came from. That might look like:



D

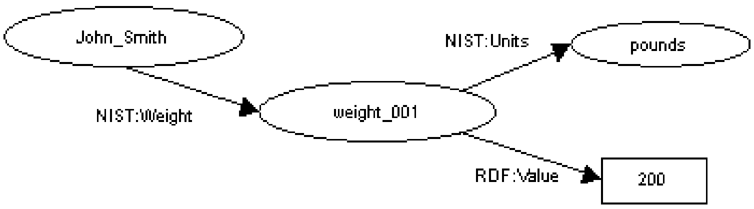
Figure 7: A ternary relation (Bag omitted)

which could be exchanged as:

```
 <?xml:namespace name="http://purl.org/metadata/dublin_core#" as="DC"?>
 <?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
 <?xml:namespace name="http://mycorp.com/schemas/my-schema#"  as="LOCAL"?>
 <RDF:RDF>
   <RDF:Description RDF:HREF="http://www.webnuts.net/Jan97.html">
     <DC:Subject>
       <RDF:Description ID="subject_001">
```

```
        <LOCAL:Classification>Dewey Decimal Code</LOCAL:Classification>
        <RDF:Value>020 - Library Science</RDF:Value>
      </RDF:Description>
    </DC:Subject>
  </RDF:Description>
</RDF:RDF>
```

A common use of this higher-arity capability is when dealing with units of measure. A person's weight is not just a number like 94, it also requires us to specify the unit of measure used. In this case we might be using either pounds or kilograms. We could use a relationship with an additional arc to record the fact that John Smith is a rather strapping gentleman:



D

Figure 8: Unit of measure as a ternary relation (Bag omitted)

which can be exchanged as:

```
<?xml:namespace name="http://www.nist.gov/RDFschema/" as="NIST"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="John_Smith">
    <NIST:Weight>
      <RDF:Description ID="weight_001">
        <NIST:Units RDF:HREF="http://www.nist.gov/units/pounds"/>
        <RDF:Value>200</RDF:Value>
      </RDF:Description>
    </NIST:Weight>
  </RDF:Description>
</RDF:RDF>
```

assuming the node "pounds" was defined in a NIST schema with the URI http://www.nist.gov/units/pounds.

6.4 Abbreviation Examples

The expression

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author>John Smith</BIB:Author>
  </RDF:Description>
</RDF:RDF>
```

may be abbreviated as

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc"
    BIB:Author="John Smith"
   />
</RDF:RDF>
```

This abbreviation is especially useful when embedding RDF in HTML documents; the text "John Smith" in the unabbreviated form will be rendered by conforming HTML 4.0 and earlier formatters while nothing would be rendered in the abbreviated form.

The expression

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author>
      <RDF:Description>
        <BIB:Name>John Smith</BIB:Name>
        <BIB:Email>john@smith.com</BIB:Email>
        <BIB:Phone>+1 (555) 123-4567</BIB:Phone>
      </RDF:Description>
    </BIB:Author>
  </RDF:Description>
</RDF:RDF>
```

may be abbreviated as

```
<?xml:namespace name="http://docs.r.us.com/bibliography-info/" as="BIB"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.bar.com/some.doc">
    <BIB:Author
      BIB:Name="John Smith"
      BIB:Email="john@smith.com"
      BIB:Phone="+1 (555) 123-4567"
     />
  </RDF:Description>
</RDF:RDF>
```

The expression

```
<?xml:namespace name="http://purl.org/metadata/dublin_core#" as="DC"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://mycorp.com/schemas/my-schema#"  as="LOCAL"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.webnuts.net/Jan97.html">
    <DC:Subject>
      <RDF:Description ID="subject_001">
        <LOCAL:Classification>Dewey Decimal Code</LOCAL:Classification>
        <RDF:Value>020 - Library Science</RDF:Value>
      </RDF:Description>
    </DC:Subject>
  </RDF:Description>
</RDF:RDF>
```

may be abbreviated as

```
<?xml:namespace name="http://purl.org/metadata/dublin_core#" as="DC"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://mycorp.com/schemas/my-schema#"  as="LOCAL"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.webnuts.net/Jan97.html">
    <DC:Subject ID="subject_001"
      LOCAL:Classification="Dewey Decimal Code"
      RDF:Value="020 - Library Science"
     />
  </RDF:Description>
</RDF:RDF>
```

The expression

```
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://mycorp.com/my-schema#" as="LOCAL"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.webnuts.net/">
    <LOCAL:Abstract>
      <RDF:Description>
        <RDF:InstanceOf href="http://mycorp.com/my-schema#Sitemap"/>
        <LOCAL:Title>Map of our Site</LOCAL:Title>
        <LOCAL:ValidUntil>1999-02-01T00:00Z</LOCAL:ValidUntil>
      </RDF:Description>
    </LOCAL:Abstract>
  </RDF:Description>
</RDF:RDF>
```

may be abbreviated as

```
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://mycorp.com/my-schema#" as="LOCAL"?>
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.webnuts.net/">
    <LOCAL:Abstract>
      <LOCAL:Sitemap
        LOCAL:Title="Map of our Site"
        LOCAL:ValidUntil="1999-02-01T00:00Z"
       />
    </LOCAL:Abstract>
  </RDF:Description>
</RDF:RDF>
```

7. Acknowledgements

This specification is the work of the W3C RDF Model and Syntax Working Group. This Working Group has been most ably chaired by Eric Miller of the Online Computer Library Center and Bob Schloss of IBM. We thank Eric and Bob for their tireless efforts in keeping the group on track and we especially thank OCLC and IBM for supporting them and us in this endeavor.

The members of the Working Group who helped design this specfication, debate proposals, provide words, proofread numerous drafts and ultimately reach consensus are: Renato Iannella (DSTC), Tsuyoshi SAKATA (DVL), Murray Maloney (Grif), Bob Schloss (IBM), Naohiko URAMOTO (IBM), Bill Roberts (KnowledgeCite), Ron Daniel (LANL), Arthur van Hoff (Marimba), Andrew Layman

(Microsoft), Chris McConnell (Microsoft), Jean Paoli (Microsoft), R.V. Guha (Netscape), Ora Lassila (Nokia), Ralph LeVan (OCLC), Eric Miller (OCLC), Charles Wicksteed (Reuters), Misha Wolf (Reuters), Wei Song (SISU), Lauren Wood (SoftQuad), Tim Bray (Textuality), Paul Resnick (University of Michigan), Tim Berners-Lee (W3C), Dan Connolly (W3C), Jim Miller (W3C), Ralph Swick (W3C).

This document is the collective work of all of the above. The editors are indebted to each of them for helping to create this document.

Appendix A: Brief Explanation of XML Namespaces

This material is non-normative, but believed necessary for an understanding of the RDF serialization syntax. It will be replaced by references to the appropriate sections of [XML] once they have been published. The material here is our best understanding of the namespace proposal, but may differ from what the final outcome will be.

The XML Working Group is currently developing a facility that will allow (at least) Generic Identifiers (tag names) to have a prefix which will make them unique and will prevent name clashes when developing documents that mix elements from different schemas. This facility allows a document's prolog to contain a set of Processing Instructions (PIs) of the form:

```
<?xml:namespace name="some-uri" as="some-abbreviation"?>
```

for example

```
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>
<?xml:namespace name="http://www.purl.org/metadata/dublin_core#" as="DC"?>
```

Elements in the document may then use generic identifiers of the form <RDF:Description> or <DC:Title>. Those element names would expand to URIs such as http://www.w3.org/TR/WD-rdf-syntax#Description.

Appendix B: Notes about Usage

B.1. Property Names

The RDF serialization and abbreviated syntaxes use XML as their encoding. XML elements and attributes are case sensitive, so RDF property names are therefore also case sensitive. This specification does not require any specific format for property names other than that they be legal XML *names*. For its own identifiers, RDF has adopted the convention that all property names use "InterCap style"; that is, the first letter of the property name is capitalized and the remainder is lowercase; e.g. *Value*. When the property name is a composition of words or fragments of words, the words are concatenated with the first letter of each word capitalized and no additional punctutation; e.g. *PropName*.

B.2. Namespace URIs

RDF uses the proposed XML namespace mechanism to implement globally unique identifiers for all properties. In addition, the namespace URI serves as the URI of the corresponding RDF schema. An RDF processor can expect to use the schema URI to access the schema content. This specification places no further requirements on the content that might be supplied at that URI, nor how (if at all) the URI might be modified to obtain alternate forms or a fragment of the schema.

B.3. Embedding in HTML

The recommended technique for embedding RDF expressions in an HTML document is simply to insert the RDF in-line. This will make the resulting document non-conformant to HTML specifications up to and including HTML 4.0 but the RDF Working Group hopes that the HTML specification will evolve to support this. Two practical issues will arise when this technique is employed with respect to browsers conforming to specifications of HTML up to and including HTML 4.0. Alternatives are available to authors in these cases; it is up to the author to choose the appropriate alternative in each circumstance.

1. Some HTML 2.0 browsers will assume a </HEAD> tag immediately before the first RDF element that appears within <HEAD>.

   Authors concerned about very old browsers may place all RDF expressions at the end of the document head.
2. All HTML browsers conforming to specifications up to and including HTML 4.0 will render any content appearing in RDF property values expressed as XML element types (i.e. production [17]).

   Authors concerned about preventing their RDF content from rendering in old browsers may use the abbreviated syntax to move the property value into an attribute. Not all properties can be expressed this way.

In the event that none of the alternatives above provides the capabilities desired by the author, the RDF expressions may be left external to the HTML document and linked with an HTML <LINK> element. The recommended relation type for this purpose is REL="meta"; e.g.

```
<link rel="meta" href="mydocMetadata">
```

B.4 Transporting in HTTP Headers

We specify a simple extension to HTTP that allows a client to request that RDF metadata be included in a header along with the document. We deal here only with the HTTP protocol; we hope that other protocols will be similarly extended.

[[*This section to be completed.*]]

B.5 Requesting Metadata via HTTP

RDF metadata can also be retrieved separately from the resource(s) to which it refers. To request metadata in this way, a client contacts a metadata service bureau. A service bureau is an HTTP server that understands a particular query syntax. It can provide metadata for documents that reside on other servers as well as for documents available through protocols other than HTTP.

[[*This section to be completed.*]]

Appendix C: Open Issues

C.1. Resource Scope

Should RDF introduce a way for an HREF to refer to a collection of resources which cannot be enumerated in any simple way (for example, by providing a URI "prefix" and indicating that the target is the collection of URIs with that prefix). The mechanism could also provide ways of restricting the set of URIs selected (e.g., only certain media types would qualify).

C.2. Mandatory Extensions

In PICS, there is a need for specifying that if a processing agent does not understand the semantics of a certain property the entire "label" should be discarded. The question for the RDF specification is whether it is sufficient to say that this is a problem which needs to be addressed by those designing a PICS schema for RDF, or whether a mechanism like this should be part of the core RDF.

C.6. Canonicalization of Models

Section 2 makes reference to testing the equivalence of RDF expressions. We need a definition of a canonical form for an instance of an RDF data model to permit comparison of two such instances. This canonical form will likely be produced by eliminating triples that represent reified properties but that are not used elsewhere in the model.

C.7. Support for schemes (ala Dublin Core)

This issue is deferred pending input from Dublin Core community and further experience with the use of namespaces

Scheme is a term used in the Dublin Core community to denote the specific controlled vocabulary from which a Value has been chosen. Examples of vocabularies covered by schemes include units of measure, library subject heading codes, and Dewey Decimal numbers. This same facility might also be used to denote a particular encoding format; e.g. date in ISO 8601 form, or name in "family, given" form. We recognize that this mechanism will be common to many RDF applications and it seems appropriate to indicate a recommended practice in the core specification.

C.10. Interactions between XML Markup Declarations and RDF

We have noticed in the description discussion that there are features of XML -- such as XLL (XML-link) -- that may have unanticipated side- effects on the intended RDF semantics. In particular, markup declarations in the document prolog may cause the XML parser to alter the interpretation of an RDF element. We need to evaluate the impact of such processing, especially in the context of digitally signed RDF descriptions.

C.20. Namespace defaulting

This issue is deferred pending a decision on XML namespaces by the XML Working Group

Should RDF specify the method for determining a namespace when a namespace qualifier is omitted?

C.21. Distributive referents

A concise mechanism is desired to state properties that apply to each element of a collection rather than to the collection as a whole.

C.22. How to compute the property type URI

This issue is deferred pending a decision on XML namespaces by the XML Working Group

The XML namespace proposal does not specify how the namespace URI is to be combined with a qualified element name to produce a URI for the element description. RDF would like to be able to compute a URI to access the RDF Schema definition of a particular property type.

C.23. Language tagging of substrings

This issue will be deferred until after the 1.0 specification.

A property value may need to be a string tagged in multiple languages; how do we accomplish this?

C.24. Requesting Metadata via HTTP

We need to specify how a client can request RDF metadata separately from the resource to which the metadata refers. This protocol is necessary to enable metadata service bureaus.

C.25. Transporting in HTTP Headers

We need to specify how RDF metadata will be transported in an HTTP header so that a client can request metadata along with the resource when it is infeasible or inappropriate to embed the metadata inside the resource.

Appendix D: References

[ISO10646]
: ISO/IEC 10646. The applicable version of this standard is defined in the XML specification [XML].

[NAMESPACES]
: Proposal for Namespaces in XML; http://www.w3.org/TR/1998/NOTE-xml-names-0119.

[Unicode]
: The Unicode Standard. The applicable version of this standard is defined in the XML specification [XML].

[XML]
: Extensible Markup Language (XML); defined in http://www.w3.org/TR/REC-xml.

Ora Lassila <ora.lassila@research.nokia.com>
Ralph R. Swick <swick@w3.org>

Revision History:
16-February-1998: Editorial cleanup, prep for second public distribution
6-February-1998: Editorial cleanup, add and revise some examples
11-January-1998: Renaming and collapsing of several elements
14-November-1997: Further refinement, especially regarding assertions
3-November-1997: Edits in preparation for second public distribution
2-October-1997: First public draft
1-October-1997: Edits in preparation for first public distribution
1-August-1997: First draft to Working Group

Last updated: 1998-02-19T02:49:22Z
Last saved: $Date: 1998/02/19 02:50:18 $