Simplified RDF Syntax for Power System Model Exchange

Arnold deVos, Langdale Consultants
adv@langdale.com.au

Revision 2	2000-10-04	Initial Release
Revision 4	2000-10-05	Corrections. Example added.
Revision 5	2000-10-10	Expanded references.
Revision 6	2000-11-16	Corrected rdf:ID attribute. Added units handling.

Introduction

This note defines a subset of the RDF Syntax [1]. This simplified syntax is proposed for exchanging power system models between utilities.

The aim of the proposal is to make it easier for implementors to construct deserializers for RDF data, to simplify their choices when serializing RDF data, and to improve the effectiveness of general XML tools such as XSLT processors when used with the serialized RDF data.

The proposed syntax is a proper subset of the standard RDF syntax. Thus, it can be read by existing RDF deserializers such as SirPAC [2]. In this, it differs from other proposals for a simplified syntax, such as [3], [4].

The proposed syntax does not sacrifice any of the power of the RDF data model. That is, any RDF data can be exchanged using this syntax. Moreover, features of RDF such as the ability to extend a model defined in one document with statements in second document are preserved.

A potential disadvantage of the proposed syntax is that is may be verbose in some cases.

Notation

The simplified syntax is defined in the following section. Each kind of element is defined in a subsection beginning with a model of the element, followed by some defining text, and a reference to the RDF grammar. The semantics of the element are not detailed, refer to the RDF recommendation [1] for that information.

Copyright Langdale Consultants 16 November, 2000

The notation for the element model is as follows:

1. A symbol in italics in the position of an element type, attribute name or attribute value indicates the type of name or value required. The symbol will be defined in the text.
2. The symbol *rdf* stands for whatever namespace prefix is chosen by the implementation for the RDF namespace. Similarly the symbol *cim* stands for the chosen EPRI CIM namespace prefix.
3. A comment within the element model indicates the allowed content. A symbol in italics stands for a kind of element or other content defined in the text. A construction (*a*|*b*) indicates that *a* and *b* are alternatives. A construction *a** indicates zero or more repetitions of *a*.
4. All other text in the model is literal.

Syntax Definition

Document Element

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:cim="cim-namespace-uri">
      <!-- Content: (definition|description)* -->
</rdf:RDF>
```

1. The element type is rdf:RDF.
2. The RDF namespace must be declared as http://www.w3.org/1999/02/22-rdf-syntax-ns#
3. The EPRI CIM namespace must be declared, including version information, in the form: http://iec.ch/TC57/2000/CIM-schema-<version># . The current <version> is cimu09a.
4. Other namespaces may be declared.

The RDF grammar clause: 6.1

Definition Element

```
<classname rdf:ID=identity>
      <!-- Content: (literal-property|units-property|resource-property)* -->
</classname>
```

1. The definition element introduces a new resource and gives its type.
2. The element type, *classname*, is the qname of a class from the EPRI CIM schema or other schema declared declared as a namespace in the document element.
3. The value of the id attribute, *identity*, is a XML name symbol chosen by the implementation. It must be unique in the document. (It is not necessarily related to the power system resource name.)

The RDF grammar clause: 6.13

Description Element

```
<rdf:Description rdf:about=resource-uri>
      <!-- Content: (literal-property|units-property|resource-property)* -->
</rdf:Description>
```

1. The description element adds information about a resource introduced elsewhere in this or another document.
2. The *resource-uri* is a URI-reference that identifies the subject resource. (This can be reduced to the fragment part for identifying resources defined in the present document.)

The RDF grammar clause: 6.13

Copyright Langdale Consultants 16 November, 2000

Literal-Property Element

```
<propname>
      <!-- Content: text -->
</propname>
```

1. The literal-property element introduces a property and a literal value applying to the enclosing resource.
2. The element type, *propname*, is the qname of a property from the EPRI CIM schema or other schema declared as a namespace in the document element.
3. The content *text* is any XML text with <, >, and & escaped representing the value of the property.

The RDF grammar clause: 6.12

Units-Property Element

```
<propname rdf:parseType="Resource">
      <rdf:value>
            <!-- Content: text -->
      </rdf:value>
      <cim:units rdf:resource="units-resource-uri"/>
</propname>
```

1. The units-property element introduces a property and a literal value annotated with units applying to the enclosing resource. It may be used instead of a literal-property element anywhere the latter is valid.
2. The element type, *propname*, is the qname of a property from the EPRI CIM schema or other schema declared as a namespace in the document element.
3. The content *text* is any XML text with <, >, and & escaped representing the value of the property..
4. The *units-resource-uri* is URI that identifies the units used for the value. The CIM schema will contain a number of standard units resources.

The RDF grammar clause: 6.12

Resource-Property Element

```
<propname rdf:resource=resource-uri/>
```

1. The resource-property element introduces a property and a resource as its value applying to the enclosing resource.
2. The element type, *propname*, is the qname of a property from the EPRI CIM schema or other schema declared as a namespace in the document element.
3. The *resource-uri* is a URI-reference that identifies a resource.

The RDF grammar clause: 6.12

Design Rationale and Alternatives

The following points explain some of the choices made in the simplified syntax.

1. The literal properties could be represented by property attributes (RDF grammar clause 6.10). This would be more compact. However, property elements were chosen because they are easier to deal with in XSLT and Xpath expressions. (For example, they can be sorted.) They also make it easier to represent multi-line text.
2. The syntax is flat, with a two-level resource/property structure. More deeply nested structures might be more compact. Morever, a well chosen nested structure might permit common queries to be more easily encoded in XSLT and Xpath expressions. On the other hand, the flat structure was chosen because it is the simplest structure possible and is easy to produce and interpret. By avoiding any application dependency on the details of a nesting structure it should be a more portable syntax.
3. All resources are given a type at the time they are introduced (by the definition element). However, the RDF model allows a resource to be untyped. In the present application untyped resources are not required.

Interoperability Suggestions

A useful feature of RDF syntax is that it allows an arbitrary subset of a power system model to be serialized in a document. This is a two edged sword, however. A document produced by one party may not be usable by a second party if it does not contain all the properties expected. Moreover, a document containing a partial model may not be usable if the resource URI's do not agree with other documents.

The following suggestions apply to the content of a document and may help maximize the range of applications that can use it.

1. Include the likely primary key properties of each resource at the point it is introduced. For example, the `cim:PowerSytemResource.powerSystemResourceName` and `cim:PowerSystemResource.ChildOf` properties are likely to be required properties. Reason: a large class of applications will want to load a database with the model data. Many database schema will require primary key values on insertion.
2. Include single-valued properties rather than their many-valued inverse. For example, use `cim:PowerSystemResource.ChildOf` and not `cim:PowerSystemResource.ParentOf`. Reason: Because these properties are inverses, a statement predicated on one implies the converse statement predicated on the other. It is less error prone to include only one side and makes editing or transforming the document easier. On the other hand, it could be argued that including both is the most interoperable choice - your mileage may vary.
3. Use stable identifiers for `rdf:id` when a model is spread among multiple documents, or when multiple versions of the model are in use. A stable identifier is one that is never reused for another purpose and never changes. An identifier that is initially derived from a random number is one way to minimize accidental reuse. Identifiers based on property values (such as a name) and sequentially allocated identifiers are subject to change. Reason: this avoids referential integrity errors between documents.
4. Use sequentially generated identifiers where referential integrity is not an issue and the document is likely to be edited or displayed in XML form. Reason: it is difficult to manually follow references within a large document if they are not organized sequentially and even more difficult if they are unpronounceable hex strings.

Example

The following example shows two resources in the simplified syntax:

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://w3.org/1999/02/22-rdf-syntax-ns#" xmlns:cim="http://epri.com/CCAPI/CIM-schema-08b.xml#">

    <cim:HostControlArea rdf:ID="ID1">
        <cim:HostControlArea.controlAreaName>WEST</cim:HostControlArea.controlAreaName>
        <cim:HostControlArea.OperatesAs rdf:resource="#ID2"/>
    </cim:HostControlArea>
    <cim:SubControlArea rdf:ID="ID2">
        <cim:PowerSystemResource.powerSystemResourceName>DEFT</cim:PowerSystemResource.powerSystemResourceName>
    </cim:SubControlArea>
</rdf:RDF>
```

References

[1] "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation, February 1999, http://www.w3.org/TR/REC-rdf-syntax

[2] "SiRPAC - Simple RDF Parser & Compiler", http://www.w3.org/RDF/Implementations/SiRPAC

[3] "A Strawman Unstriped Syntax for RDF in XML", Tim Berners-Lee, November 1999, http://www.w3.org/DesignIssues/Syntax

[4] "Simplified Syntax for RDF", Sergey Melnik, November 1999, http://www-db.stanford.edu/~melnik/rdf/syntax.html