RosettaNet
Lingua franca for eBusiness

Comments on Universal Business Language (UBL) Naming and Design Rules

Issue 1.0
06 September 2003

Table of Contents

1 Document Management

1.1 Legal Disclaimer

RosettaNet®, its members, officers, directors, employees, or agents shall not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document. The use of said document shall constitute your express consent to the foregoing exculpation.

1.2 Copyright

1.3 Trademarks

1.4 Document Version History

Version	Who	Date	Description
1.0	Nikola Stojanovic, Suresh Damodaran	Sep 6, 2003	Initial version

1.5 Audience

This document's primary audience is the UBL Naming and Design Rules Subcommittee.

1.6 Acknowledgements

We are thankful to Uniform Code Council team, especially, Rajesh Manglani and Ted O'sinski for their detailed analysis of the rules, and comments on the rules.

2 Comments

2.1 Purpose of the review

The RosettaNet team has produced its own XML Design Guidelines [RNXD] with the purpose of using them in developing its own XML Schema documents. As the basic purpose of [RNXD] document is analogous to the purpose of the UBL Naming and Design Rules [UBLN] and as the two groups do have a formal liaisoning channel, this is RosettaNet’s feedback to UBL Naming and Design Rules Subcommittee on rules expressed in UBL Schema Naming and Design Rules Checklist [UBLN].

2.2 Generic

2.2.1 Creation Procedure

Approaches of creation of Schemas differ between RosettaNet and UBL. RosettaNet creates Schemas from UML models, which is not the case with UBL. Also, many of the rules in [UBLN] are specific to CCTS. RosettaNet does not use CCTS. CCTS related rules are not evaluated in this paper.

2.2.2 Versioning

Here are some suggestion related to versioning approach:

1. targetNamespace MUST NOT include minor version number.

As [UBLN] major and minor version numbers are embedded in a particular namespace that implies that changing minor version number doesn’t allow replacement of old schema with the new schema respective to keeping the same instance documents.

Suggestions:

Enforce embedding only major version number in the “targetNamespace”. This enables RosettaNet to not enforce a schema version bumping rule: A major version change happens only if an XML instance that was validated with a previous version of the schema cannot be validated with the new version of the schema. By using only the major version in namespace, needless changes to multiple files can be avoided.

2. Usage of xsd:schema built-in “version” attribute

While [UBLN] forces inclusion of version information in the namespace, it doesn’t convey version of the Schema Module itself. In case when lifecycle of the Schema Module is independent of the lifecycle of the relevant namespace, it is useful to have an independent Schema Module version.

Suggestions:

“version” attribute of xsd:schema MUST be present and its value MUST reflect the version of the Schema. This will allow the major.minor version available with the schema definition for any processor that still wants to make changes based on the minor version.

3. Versioning of types in Schemas

In case when lifecycle of a “type” inside a Schema Module is independent of the lifecycle of the Schema Module, it is useful to embed the “type version” inside the Schema.

Suggestions:

Require a new annotation for “TypeVersion” to every type definition. Example below.

```
<xs:annotation>
      <xs:appinfo xml:lang="US_EN">
         <Constraint> Schematron constraint if any</Constraint>
         <Context> Reusable type here </Context>
         <CreationDate> 20/06/2003 </CreationDate>
         <Keyword> Invoicing </Keyword>
         <LastUpdateDate> 20/06/2003 </ LastUpdateDate >
          <Definition> State the definition here </DEfinition>
          <TypeVersion> 0.14 </TypeVersion>
      </xs:appinfo>
</xs:annotation>
```

4. Exposing “Schema Version” via instance documents

Sometimes it is beneficial to be able to correlate a given instance document fragment to the type definition in a particular namespace so that processing application at the destination can take appropriate action(s).

Suggestions:

A common global attribute “schemaVersion” of the “xsd:token” type MUST be declared as an optional attribute for all Root Schema type definitions.

Instance documents SHOULD set the value of the “schemaVersion” attribute. The “schemaVersion” attribute MAY contain more then one value of the Schema versions that the instance document is compatible with.

2.3 Specific Rules

2.3.1 Rule 38

Rule: Schema location must include the complete URI which is used to identify schema modules.

Comment: Schemas might be placed in different root directories. Also, for security reasons it is advisable not to reveal the location of the Schemas.

Suggestions:

“schemaLocation” attribute MUST point to the imported Schema via relative path with respect to the location where the current Schema is stored.

2.3.2 Rule 47

Rule: Each minor version must be given a separate namespace.

Comment: See discussion about versioning in section 2.2.2.

Suggestions:

Remove the rule.

2.3.3 Rule 83

Rule: Trading partners may agree on other character encodings to use among themselves. It is recommended in all case that encoding declarations be provided in the XML declarations of documents.

Comment: For interoperability reasons it would be beneficial to restrict the character encoding.

Suggestions:

Trading partners MUST use either UTF-8 or UTF-16 character encodings among themselves. It is recommended in all case that encoding declarations be provided in the XML declarations of documents. 'UTF-8' or 'UTF-16' MUST be used as the value for character set and encoding type.

2.3.4 Rule 84

Rule: messages must express semantics fully in schemas and not rely merely on well-formedness

Comment: It is not clear what the rule enforces. Is the intention to say that all instance documents need to be valid?

Suggestions:

Change the wording to make the rule clear.

2.3.5 Rule 89

Rule: Acronyms and abbreviations must only be taken from the latest version of the Pocket Oxford English Dictionary. The first occurrence listed for a word will be the preferred item to be used.

Comment: This rule seems to contradict with Rule 87. Also, there exist industry specific acronyms that are in wide use.

Suggestions:

Remove the rule.

2.3.6 Rule 103

Rule: Substitution groups MUST NOT be used.

Comment: Sometimes it is useful to use substitution groups.

Suggestions:

Substitution groups MAY be used with caution.

2.3.7 Rule 107

Rule: The XSD prefix MUST be used. (xmlns:xsd=http://www.w3.org/2001/xmlSchema)

Comment: It is also common to use “xs” prefix.

Suggestions:

The “xsd” or “xs” namespace prefix MUST be used. (e.g. xmlns:xsd=http://www.w3.org/2001/xmlSchema)

2.3.8 Rule 115

Rule: All documents shall have a container for metadata and which proceeds the body of the document and is named "Head" _____________. (anything but header).

Comment: This rule is not clear.

Suggestions:

Change the wording to make the rule clear.

2.3.9 Rule 116

Rule: All elements with a cardinality of 1..n, (and lack a qualifying structure) must be contained by a list container named "(name of repeating element)List", which has a cardinality of 1..1.

Comment: Lists could be empty.

Suggestions:

All elements with a cardinality of 0..n, (and lack a qualifying structure) must be contained by a list container named "(name of repeating element)List", which has a cardinality of 1..1.

2.4 New rules

Here are some rules that don’t exist in [UBLN] and that are suggested for inclusion in the future versions of the document.

Note: If accepted change rule number accordingly.

2.4.1 Rule RN 1

Rule: "tns” namespace prefix SHOULD be used to indicate xsd:targetNamespace when xsd:targetNamespace is not the same as the default namespace of the Schema.

Comment: When looking into Schema it is useful to be able to distinguish between components that belong to “this namespace” from components that are imported from other namespaces. Using “tns” namespace prefix makes this distinction consistent across all Schemas.

2.4.2 Rule RN 2

Rule: Order of xsd:schema attributes MUST be as follows: targetNamespace declaration, declaration binding “xsd” namespace prefix, default namespace declaration, declaration binding “tns” prefix, any other declarations binding prefixes to other namespaces, elementFormDefault declaration, attributeFormDefault declaration and version declaration"/>

Comment: Consistent placement / ordering of components helps with human readability and debuggability of Schemas.

2.4.3 Rule RN 3

Rule: XML Schema built-in default values MUST be specified consistently.

Comment: Having mixed approach when indicating XML Schema built-in default values, like sometimes indicating minOccurs=”1” and sometimes not, is often confusing for the human audience.

2.4.4 Rule RN 4

Rule: Schemas MUST follow consistent structuring rules.

Comment: Consistent placement / ordering of components helps with human readability and debuggability of Schemas. RosettaNet uses the following structuring rules.

Rule

1. Logically related constructs SHOULD be placed together in the same file in order to support better abstraction, reusability and clarity.
2. Logically related constructs within the same file SHOULD be placed in close proximity to promote understanding.
3. The documentation for a Schema SHOULD be placed just after the top-level xs:schema element. The documentation for individual components as listed above SHOULD be placed immediately after the component name declaration / definition.
4. When not in violation of the previous rules, the following SHOULD be the desired order of global Schema components.

   Reusable global element(s),
   Global element named groups,
   Global reusable attributes,
   Global attribute named groups,
   Global simple types,
   Global complex types with sequence content model,
   Global complex types with choice content model,

All of these components are internally sorted alphabetically by names.

Ordering of components within Type definition

Rule

Within the type definition, the sequences, choice, groups and sub-content models SHOULD be ordered in alphabetical order. Also within each content model (like sequence, choice, groups etc) elements SHOULD be sorted in alphabetical order.
The only exception is in the order of attributes and attribute groups. In element and type definitions, the attributes and attribute groups SHOULD be listed alphabetically at the end, after the content model and elements.

Rationale

This ordering scheme permits easy reading of Schemas for debugging purposes.

2.4.5 Rule RN 5

Rule: "xsd:redefine MUST NOT be used

Comment: Besides of the possibly of changing the semantics of redefined definitions, xsd:redefine might also cause conflicts when further modifications to redefined definitions are needed. Possibility of redefining already redefined definitions makes the usage of xsd:redefine even more problematic.

2.4.6 Rule RN 6

Rule: While creating names for inner elements, concatenating the name of the inner element to the name of the outer element SHOULD be avoided. The exception to this rule is the following:
if the outer element name cannot be prefixed with *all* inner element names sensibly, then each inner element name SHOULD be created by concatenating the outer element name to it.

In the example below, both the elements Address and Phone would be sensible as “ContactAddress” and “ContactPhone” – because of this, concatenating Contact with the Address and Phone is avoided.

```
<complexType name="ContactType">
   <complexContent>
      <extension base="us:SomeBaseType">
         <sequence>
            <element name="Address" type="xyz:AddressType"/>
            <element name="Phone" type="xyz:PhoneType"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<element name=”Contact” type=”ContactType”>
```

Comment: The XML Schema naming rules for elements and types follow the names used in previous stages (RosettaNet uses UML), and therefore, are not enforceable. RosettaNet, however, has some rules on naming, and the above is one of the few. It is unclear this rule is contradicting any existing UBL rules or not.

3 References

Source	**Description**
[CCTS]	*UN/CEFACT Draft Core Components Specification* 30 September, 2002, Version 1.85
[RNXD]	*RosettaNet Design Guidelines*, version 0.97
[UBLN]	*UBL Schema Naming and Design Rules Checklist* http://lists.oasis-open.org/archives/ubl-ndrsc/200306/doc00003.doc