Contents

Document Schema Definition Languages (DSDL) — Part 2: Grammar-based validation — RELAX NG Compact Syntax

Amendment 1

Compact Syntax

Page 34

Add the following text before the Bibliography:

Annex C
(normative)

RELAX NG Compact Syntax

C.1 Introduction

This Annex describes a compact, non-XML syntax for RELAX NG. This syntax is an alternative to the XML syntax described in Clause 5 of ISO 19757-2:2003.

The goals of the compact syntax are to:

— maximize readability

— support all features of RELAX NG; it must be possible to translate a schema from the XML syntax to the compact syntax and back without losing significant information

— support separate translation; a RELAX NG schema may be spread amongst multiple files; it must be possible to represent each of the files separately in the compact syntax; the representation of each file must not depend on the other files

C.2 Syntax

The following is a summary of the syntax in EBNF. Square brackets are used to indicate optionality. The start symbol is topLevel.

```
topLevel ::=
decl* (pattern | grammarContent*)

decl ::=
(namespace identifierOrKeyword = namespaceURILiteral)
| (default namespace [identifierOrKeyword] = namespaceURILiteral)
| (datatypes identifierOrKeyword = literal)

pattern ::=
(element nameClass { pattern })
| (attribute nameClass { pattern })
| (pattern (, pattern)+)
| (pattern (& pattern)+)
| (pattern (| pattern)+)
| (pattern ?)
| (pattern *)
| (pattern +)
| (list { pattern })
| (mixed { pattern })
| identifier
| (parent identifier)
| empty
| text
| ([datatypeName] datatypeValue)
| (datatypeName [{ param* }] [exceptPattern])
| notAllowed
| (external anyURILiteral [inherit])
| (grammar { grammarContent* })
| (( pattern ))
```

```
param ::=
identifierOrKeyword = literal

exceptPattern ::=
- pattern

grammarContent ::=
start
| define
| (div { grammarContent* })
| (include anyURILiteral [inherit] [{ includeContent* }])

includeContent ::=
define
| start
| (div { includeContent* })

start ::=
start assignMethod pattern

define ::=
identifier assignMethod pattern

assignMethod ::=
=
||=
|&=

nameClass ::=
name
| (nsName [exceptNameClass])
| (anyName [exceptNameClass])
| (nameClass | nameClass)
| (( nameClass ))

name ::=
identifierOrKeyword
| CName

exceptNameClass ::=
- nameClass

datatypeName ::=
CName
| string
| token

datatypeValue ::=
literal

anyURILiteral ::=
literal

namespaceURILiteral ::=
literal
| inherit

inherit ::=
inherit = identifierOrKeyword
```

```
identifierOrKeyword ::=
identifier
| keyword

identifier ::=
NCName
| quotedIdentifier

quotedIdentifier ::=
\ NCName

CName ::=
NCName : NCName

nsName ::=
NCName :*

anyName ::=
*

literal ::=
literalSegment (~ literalSegment)+

literalSegment ::=
(" Char* ")
| (' Char* ')
| (""" (["] ["] Char)* """)
| (''' ([']  ['] Char)* ''')

keyword ::=
attribute
| default
| datatypes
| div
| element
| empty
| external
| grammar
| include
| inherit
| list
| mixed
| namespace
| notAllowed
| parent
| start
| string
| text
| token
```

NCName is defined in W3C XML-Names[2]. Char is defined in W3C XML[1]. A formal definition for the compact syntax is defined in an Oasis Committee Specification[3].

In order to use a keyword as an identifier, it must be quoted with \. It is not necessary to quote a keyword that is used as the name of an element or attribute or as datatype parameter.

The value of a literal is the concatenation of the values of its constituent literalSegments. A literalSegment is always terminated only by an occurrence of the same delimiter that began it. The delimiter used to begin a literalSegment may be either one or three occurrences of a single or double quote character. Newlines are allowed only in literalSegments delimited by three quote characters. The value of a literal segment consists of the characters between its delimiters. One way to get a literal whose value contains both a single and a double quote is to divide the literal into multiple literalSegments so that the single and double quote are in separate literalSegments. Another way is to use a literalSegment delimited by three single or double quotes.

Annotations can be specified as described in C.5.

There is no notion of operator precedence. It is an error for patterns to combine the **|**, &, , and - operators without using parentheses to make the grouping explicit. For example, foo **|** bar, baz is not allowed; instead, either (foo **|** bar), baz or foo **|** (bar, baz) must be used. A similar restriction applies to name classes and the use of the **|** and - operators. These restrictions are not expressed in the above EBNF.

The value of an anyURILiteral specified with include or external is a URI reference to a grammar in the compact syntax.

C.3 Lexical structure

Whitespace is allowed between tokens. Tokens are the strings occurring in double quotes in the EBNF in C.2, except that literalSegment, nsName, CName, identifier and quotedIdentifer are single tokens.

Comments are also allowed between tokens. Comments start with a # and continue to the end of the line. Comments starting with ## are treated specially; see C.5.

A Unicode character with hex code N can be represented by the escape sequence \x{N}. Using such an escape sequence is completely equivalent to the entering the corresponding character directly. For example,

```
element \x{66}\x{6f}\x{6f} { empty }
```

is equivalent to

```
element foo { empty }
```

C.4 Declarations

A datatypes declaration declares a prefix used in a QName identifying a datatype. For example,

```
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"
element height { xsd:double }
```

In fact, in the above example, the datatypes declaration is not required: the xsd prefix is predeclared to the above URI.

A namespace declaration declares a prefix used in a QName specifying the name of an element or attribute. For example,

```
namespace rng = "http://relaxng.org/ns/structure/1.0"
element rng:text { empty }
```

As in XML, the xml prefix is predeclared.

A default namespace declaration declares the namespace used for unprefixed names specifying the name of an element (but not of an attribute). For example,

```
default namespace = "http://example.com"
element foo { attribute bar { string } }
```

is equivalent to

```
namespace ex = "http://example.com"
element ex:foo { attribute bar { string } }
```

A default namespace declaration may have a prefix as well. For example,

```
default namespace ex = "http://example.com"
```

is equivalent to

```
default namespace = "http://example.com"
namespace ex = "http://example.com"
```

The URI may be empty. This makes the prefix stand for the absent namespace URI. This is necessary for specifying a name class that matches any name with an absent namespace URI. For example,

```
namespace local = ""
element foo { attribute * - local:* { string }* }
```

is equivalent to

```
<element xmlns="http://relaxng.org/ns/structure/1.0" name="foo" ns="http://example.com">
  <zeroOrMore>
    <attribute>
      <anyName>
        <except>
          <nsName ns=""/>
        </except>
      </anyName>
      <data type="string"/>
    </attribute>
  <zeroOrMore>
</element>
```

RELAX NG has the feature that if a file does not specify an ns attribute then the ns attribute can be inherited from the including file. To support this feature, the keyword inherit can be specified in place of the namespace URI in a namespace declaration. For example,

```
default namespace this = inherit
element foo { element * - this:* { string }* }
```

is equivalent to

```
<element xmlns="http://relaxng.org/ns/structure/1.0" name="foo">
  <zeroOrMore>
    <element>
      <anyName>
        <except>
          <nsName/>
        </except>
      </anyName>
      <data type="string"/>
    </element>
  <zeroOrMore>
</element>
```

In addition, the include and external patterns can specify inherit = prefix to specify the namespace to be inherited by the referenced file. For example,

```
namespace x = "http://www.example.com"
external "foo.rng" inherit = x
```

is equivalent to

```
<externalRef href="foo.rng"
  ns="http://www.example.com"
  xmlns="http://relaxng.org/ns/structure/1.0"/>
```

In the absence of an inherit parameter on include or external, the default namespace will be inherited by the referenced file.

In the absence of a default namespace declaration, a declaration of

```
default namespace = inherit
```

is assumed.

C.5 Annotations

C.5.1 Support for annotations

The RELAX NG XML syntax allows foreign elements and attributes to be used to annotate a RELAX NG schema.

A schema in the compact syntax can also have annotations, which will turn into foreign elements and attributes when the schema is translated into XML syntax. The way these annotations are specified depends on where the foreign elements and attributes are to appear in the translated schema. There is also a special shorthand syntax when the foreign element is a documentation element.

C.5.2 Initial annotations

An annotation in square brackets can be inserted immediately before a pattern, param, nameClass, grammarContent or includeContent. It has the following syntax:

```
annotation ::=
[ annotationAttribute* annotationElement* ]

annotationAttribute ::=
name = literal

annotationElement ::=
name [ annotationAttribute* (annotationElement | literal)* ]
```

Each of the annotationAttributes will turn into attributes on the corresponding RELAX NG element. Each of the annotationElements will turn into initial children of the corresponding RELAX NG element, except in the case where the RELAX NG element cannot have children, in which case they will turn into following elements.

C.5.3 Documentation shorthand

Comments starting with ## are used to specify documentation elements from the http://relaxng.org/ns/compatibility/annotations/1.0 namespace. For example,

```
## Represents a language
element lang {
## English
"en" |
## Japanese
"jp"
}
```

turns into

```
<element name="lang"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <a:documentation>Represents a language</a:documentation>
  <choice>
    <value>en</value>
    <a:documentation>English</a:documentation>
    <value>jp</value>
    <a:documentation>Japanese</a:documentation>
  </choice>
</element>
```

## comments can only be used immediately before a pattern, nameClass, grammarContent or includeContent. Multiple ## comments are allowed. Multiple adjacent ## comments without any intervening blank lines are merged into a single documentation element. Any ## comments must precede any annotation in square brackets.

C.5.4 Following annotations

A pattern or nameClass may be followed by any number of followAnnotations with the following syntax:

```
followAnnotation ::=
>> annotationElement
```

Each such annotationElement turns into a following sibling of the RELAX NG element representing the pattern or nameClass.

C.5.5 Grammar annotations

An annotationElement may be used in any place where grammarContent or includeContent is allowed. For example,

```
namespace x = "http://www.example.com"
start = foo
x:entity [ name="picture" systemId="picture.jpeg" notation="jpeg" ]
foo = element foo { empty }
```

turns into

```
<grammar xmlns:x="http://www.example.com"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <ref name="foo"/>
  </start>
  <x:entity name="picture" systemId="picture.jpeg" notation="jpeg"/>
  <define name="foo">
    <element name="foo">
      <empty/>
    </element>
  </define>
</grammar>
```

If the name of such an element is a keyword, then it must be quoted with \.

C.6 Conformance

C.6.1 Types of conformance

There are three types of conformant implementation of the compact syntax.

C.6.2 Validator

A validator conforming to this specification must be able to determine whether a textual object is a correct RELAX NG Compact Syntax schema as specified in this Annex. It must also be able to determine for any XML document and for any correct RELAX NG Compact Syntax schema whether the document is valid (as defined in Clause 9 of ISO 19757-2:2003) with respect to the translation of the schema into XML syntax. It need not be able to output a representation of the translation of the schema into XML syntax.

The requirements in the preceding paragraph are subject to the provisions of the second paragraph of Clause 11 of ISO 19757-2:2003.

C.6.3 Structure preserving translator

A structure preserving translator must be able to translate any correct RELAX NG Compact Syntax schema into an XML document whose data model is strictly equivalent to the translation specified in this Annex. For this purpose, two instances of the data model (as specified in Clause 5 of ISO 19757-2:2003) are considered strictly equivalent if they are identical after applying the simplifications specified in Sections 7.2, 7.3, 7.4, 7.8, 7.9 and 7.10 of ISO 19757-2:2003, with the exception that the base URI in the context of elements may differ.

NOTE 1: The RELAX NG compact syntax is not a representation of the XML syntax of a RELAX NG schema; rather it is a representation of the semantics of a RELAX NG schema. Details of the XML syntax that were judged to be insignificant are not captured in the compact syntax. For example, in the XML syntax if the name class for an element or attribute pattern consists of just a single name, it can be expressed either as a name attribute or as a name element; however, in the compact syntax, there is only one way to express such a name class. The simplifications listed in the previous paragraph correspond to those syntactic details that are not captured in the compact syntax.

When comparing two include or externalRef patterns in the XML source for strict equivalence, the value of the href attributes are not compared; instead the referenced XML documents are compared for strict equivalence.

C.6.4 Non-structure preserving translator

A non-structure preserving translator must be able to translate any correct RELAX NG Compact Syntax schema into an XML document whose data model is loosely equivalent to the translation specified in this Annex. For this purpose, two instances of the data model (as specified in Clause 5 of ISO 19757-2:2003) are considered loosely equivalent if they are such that, after applying all the simplifications specified in Section 7 of this part of ISO/IEC 19757, one can be transformed into the other merely by reordering and renaming definitions. After the simplifications have been applied, the context of elements is ignored when comparing the two instances.

NOTE 2: A validator for the compact syntax can be implemented as a combination of a non-structure preserving translator for the compact syntax and a validator for the XML syntax.

Bibliography

[1] *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[2] *Namespaces in XML*, W3C Recommendation, 14 January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[3] *RELAX NG Compact Syntax*, Committee Specification, 21 November 2002, http://www.oasis-open.org/committees/relax-ng/compact-20021121.html