XMLmind XML Editor - Support of RELAX NG Schemas

Hussein Shafie, Pixware `<xmleditor-support@xmlmind.com>`

December 2, 2005

Abstract

This document describes how RELAX NG schemas are supported by XMLmind XML Editor.

Table of Contents

1. Implementation of RELAX NG in XMLmind XML Editor ... 1
2. Specifying which RELAX NG schema to use for validating a document ... 2
    2.1. The `relaxng` configuration element ... 2
    2.2. The `<?xxe-relaxng-schema>` processing instruction ... 2
    2.3. Sample XXE configurations using RELAX NG schemas ... 3
3. XMLmind XML Editor-friendly content models ... 3
    3.1. The non-validating, lenient, editing mode ... 6
    3.2. Problems with attributes ... 8
    3.3. Help provided by the "Show Content Model" window ... 10
    3.4. Other content models which are not XXE-friendly ... 12
4. Command line tools ... 13
    4.1. rngvalid ... 13
    4.2. rngdoc ... 14
5. Missing features ... 14

1. Implementation of RELAX NG in XMLmind XML Editor

The implementation of RELAX NG in XMLmind XML Editor (XXE for short) is based on Jing, an *OpenSource*, *industrial strength*, *streaming* validator written by James Clark.

The trimmed version of Jing included in XXE (`relaxng.jar`) is used

- to load and validate RELAX NG schemas associated to XML documents (XML and compact syntaxes are both supported);
- to fully validate XML documents conforming to RELAX NG schemas, each time these documents are opened and saved, and each time a full validation is explicitly requested by the user (command Tools|Check Validity).

Jing is not used to implement guided editing. That is, Jing is not used to determine the content model of the element being edited. A quick, incremental, version of the algorithm that computes the derivative of a pattern is used for that.

The implementation of W3C XML Schema Datatypes used in RELAX NG schemas (e.g. `xsd:int`) is the work of XMLmind. This implementation is very different from the implementation of W3C XML Schema Datatypes included in the original Jing. This implementation is shared by the version of Jing included in XXE and by our own W3C XML Schema validator.

XXE supports attribute default values as specified in RELAX NG DTD Compatibility. The compatibility of the schema with this feature is (*very strictly*) checked by XXE and not by Jing. This means that a schema found valid by Jing but improperly using this feature will be rejected by XXE.

2. Specifying which RELAX NG schema to use for validating a document

This section is just a primer. The reference documentation about this topic is really XMLmind XML Editor - Configuration and Deployment.

2.1. The `relaxng` configuration element

A document type declaration (`<!DOCTYPE>`) can be used to associate a DTD to a document. Attributes `xsi:schemaLocation`/`xsi:noNamespaceSchemaLocation` can be used to associate W3C XML Schemas to a document. But there is no standard way to associate a RELAX NG schema to a document. Therefore this association must be made using an external specification such as the Namespace Routing Language (NRL).

In the case of XMLmind XML Editor, this external specification is simply a configuration element called `relaxng`.

XHTML example:

```
<configuration name="XHTML Strict [RELAX NG]"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns="http://www.xmlmind.com/xmleditor/schema/configuration">
  <include location="xxe-config:schema/ns_xhtml.incl" />

  <detect>
    <rootElementNamespace>http://www.w3.org/1999/xhtml</rootElementNamespace>
  </detect>

  <relaxng name="http://www.w3.org/1999/xhtml"❶
           location="xxe-config:common/rng/xhtml1/xhtml-strict.rng" />

  <preserveSpace elements="html:pre html:style html:script" />❷

  <css name="XHTML" location="xhtml_rng.css" />
  <template name="Page" location="page.html" />
</configuration>
```

❶ The `name` attribute, which is similar to the public ID of a DTD, is absolutely not required. However, if it is absent, the corresponding RELAX NG schema cannot be cached by XXE.

❷ Unlike the DTD, `xhtml-strict.rng` does not specify a `preserve` default value for attribute `xml:space` of elements such as `pre`. Therefore, the `preserveSpace` configuration element must be used to specify whitespace-preserving elements.

2.2. The `<?xxe-relaxng-schema>` processing instruction

This processing instruction is a non standard, proprietary, way to associate a document to a RELAX NG schema. *Its use should be restricted to testing and other quick and dirty experiments.*

DocBook example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xxe-relaxng-schema name="-//OASIS//RELAX NG DocBook V4.3//EN"❶
  location="http://www.docbook.org/rng/4.3/docbook.rnc"
  compactSyntax="true" encoding="US-ASCII" ?>

<!DOCTYPE article [❷

<!ENTITY % sgml.features "IGNORE">
<!ENTITY % xml.features "INCLUDE">

<!ENTITY euro "&#x20AC;">

<!ENTITY % dbcent PUBLIC
"-//OASIS//ENTITIES DocBook Character Entities V4.3//EN"
"http://www.oasis-open.org/docbook/xml/4.3/dbcentx.mod">
%dbcent;

]>

<article>
  <title></title>
  <section>
    <title></title>
    <para></para>
  </section>
</article>
```

1. The `<?xxe-relaxng-schema>` processing instruction has pseudo-attributes identical to the attributes of the `relaxng` configuration element.
2. Using a RELAX NG schema should not prevent you from specifying a document type declaration for character entities.

2.3. Sample XXE configurations using RELAX NG schemas

The examples used in this section come from two ready-to-use XXE configurations:

`docbook5/`

Allows to create and edit DocBook V5.0b1 documents conforming to the RELAX NG schema coming from DocBook.org.

Caution

Do not use this schema and its associated configuration for serious work. This configuration has been created mainly to test the support of RELAX NG in XXE.

`xhtml_rng/`

Allows to create and edit XHTML 1.0 documents conforming to the modular RELAX NG schema written by James Clark.

Note that these two configurations do not conflict with the DTD-based XHTML and DocBook configurations which are bundled with XXE.

These configurations are found in `XXE_install_dir/doc/rngsupport/config/`. To use any of them, simply copy the corresponding directory to

- `XXE_install_dir/addon/`
- OR to `XXE_user_preferences_dir/addon/` (recommended). XXE user preferences directory is:
  - `$HOME/.xxe/` on Unix,
  - `%SystemDrive%\Documents and Settings\%USERNAME%\Application Data\XMLmind\XMLeditor\` on Windows 2000/XP,
  - `%SystemDrive%\winnt\Profiles\%USERNAME%\Application Data\XMLmind\XMLeditor\` on Windows NT.

For example, to create DocBook documents conforming to a RELAX NG schema, on Windows, copy `XXE_install_dir/doc/rngsupport/config/docbook5/` to `%SystemDrive%\Documents and Settings\%USERNAME%\Application Data\XMLmind\XMLEditor\addon\`.

3. XMLmind XML Editor-friendly content models

Validating a document against a RELAX NG schema is similar to matching some text against a regular expression. If the document ``matches" the schema, the document is valid, and this, no matter which sub-expressions were used during the match.

Example: string "`b`" matches regular expression "`(a?,b)|(b,c?)`" and we don't care if it matches sub-expression "`(a?,b)`" or sub-expression "`(b,c?)`". The situation is exactly the same with RELAX NG schemas, simply replace the characters and the character classes used in a regular expression by RELAX NG patterns.

The job of a RELAX NG schema is a validate a document as a whole, and that's it. For XXE, the problem to solve is different. One of the main jobs of XXE is to guide the user when he/she edits an XML document. That is, one of the main jobs of XXE is to identify the content model of the element which is being edited, in order to suggest the right attributes and the right child elements for it.

To do that, XXE needs to know precisely which ``sub-expressions were used during the match''. Unfortunately, sometimes, this is impossible to do.

All examples used in this section are found in *`XXE_install_dir`*`/doc/rngsupport/samples/`. Note that they are all valid schemas and valid documents.

Example 1. Ambiguous elements

RELAX NG schema, target.rnc:

```
start = build-element

build-element = element build {
    target-element*
}
target-element = element target {
    attribute name { xsd:ID },
    element list { ref-element* }?,
    element list { action-element* }?
}
ref-element = element ref {
    attribute name { xsd:IDREF }
}
action-element = element action { text }
```

Document conforming to the above schema, target_bad.xml:

```
<build>
  <target name="all">
    <list>
    </list>
  </target>

  <target name="compile"/>
  <target name="link"/>
</build>
```

If you open `target_bad.xml` in XXE and select the `list` element, *XXE is lost*: is it the `list` element which contains `refs` or is the `list` element which contains `actions`? Both `list` content models are fine in the case of an empty `list` element!

Now, if you open target_good.xml in XXE, there is no problem at all:

```
<build>
  <target name="all">
    <list>
      <ref name="compile"/>
      <ref name="link"/>
    </list>
    <list>
      <action>cc -c *.c</action>
      <action>cc *.o</action>
    </list>
  </target>

  <target name="compile"/>
  <target name="link"/>
</build>
```

The previous examples show that:

> ## Important
>
> XXE cannot make a difference between two child elements having the same name and having different content models, *unless these two child elements have themselves distinct attributes and/or distinct child elements*.

RELAX NG schema, sect.rnc:

```
start = doc-element

doc-element = element doc {
    (simple-sect|
     recursive-sect)+
}
simple-sect = element sect {
    attribute class {"simple"}, paragraph-element*
}
recursive-sect = element sect {
    attribute class {"recursive"}, (recursive-sect|simple-sect)*
}
paragraph-element = element paragraph { text }
```

Document conforming to the above schema, sect.xml:

```
<doc>
  <sect class="recursive">
    <sect class="recursive"></sect>

    <sect class="simple">
      <paragraph>Paragraph 2.</paragraph>
    </sect>
  </sect>

  <sect class="simple"></sect>
</doc>
```

XXE has no problem at all with empty `<sect class="recursive">` and empty `<sect class="simple">` because these elements have the *same required attribute* `class` *but with different fixed values*. However, it is easy to defeat XXE by slightly modifying the schema.

RELAX NG schema, sect2.rnc:

```
start = doc-element

doc-element = element doc {
    (simple-sect|
     recursive-sect)+
}

simple-sect = element sect {
    attribute class {"simple"}, paragraph-element*
}

recursive-sect = element sect {
    attribute class {"recursive"}?, (recursive-sect|simple-sect)*
}

paragraph-element = element paragraph { text }
```

Document conforming to the above schema, sect2.xml:

```
<doc>
  <sect>
    <sect></sect>

    <sect class="simple">
      <paragraph>Paragraph 2.</paragraph>
    </sect>
  </sect>

  <sect class="simple"></sect>
</doc>
```

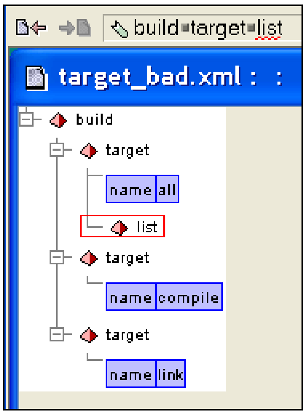
3.1. The non-validating, lenient, editing mode

When XXE is ``lost", it automatically enters a *lenient editing mode*. In this mode, XXE can no longer guide the user when he/she edits the element which poses problems.

The node path bar is used to signal elements which are in this non-validating, lenient editing mode:

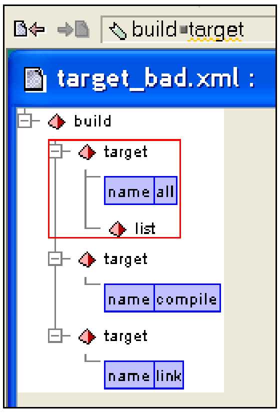
- An element underlined in *red* means that this element is in *non-validating mode 2*. In this mode, XMLmind XML Editor is not able to suggest the right attributes and the right child elements to the user. The user may add and remove any attributes and child elements he/she wants, at any place and in any number.

Figure 1. XXE is completely lost when empty `list` is selected



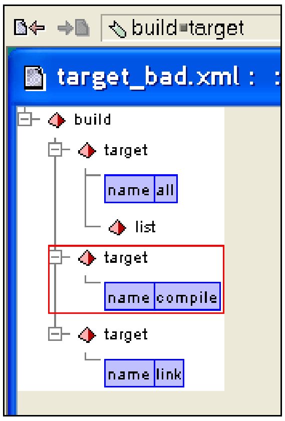
- An element underlined in *orange* means that this element is in *non-validating mode 1*. In this mode, XMLmind XML Editor still suggests the right attributes and child elements to the user. But these are only suggestions: the user may add and remove any attributes and child elements he/she wants, and this, at any place and in any number.

Figure 2. XXE has problems when `target` containing empty `list` is selected



Note that the lenient editing mode is local to an element and its descendants. It is not used for the whole document, but just for the element for which XXE has troubles.

Figure 3. XXE has no problem at all when `target` named `compile` is selected



Also note that, after modifying an element which poses problems to XXE, if these problems are solved, XXE will automatically switch to its normal, strict, validating mode.

3.2. Problems with attributes

Important

XXE cannot make a difference between two attributes (within the same element) having the same name and having different content models, *unless these two attributes have both fixed values*.

Example 2. Same attribute name, different content models

RELAX NG schema, person.rnc:

```
start = persons-element

persons-element = element persons {
    person-element+
}
person-element = element person {
    (attribute age { xsd:int } |
     (attribute age { "seeBirthDate" },
      attribute birthDate { xsd:date })),
    element firstName { text },
    element lastName { text }
}
```

Document conforming to the above schema, person.xml:

```
<persons>
  <person age="33">
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </person>

  <person age="seeBirthDate" birthDate="1980-04-15">
    <firstName>Erica</firstName>
    <lastName>Kyle</lastName>
  </person>
</persons>
```

XXE has problems with attribute `age`. It cannot make a difference between attribute `age` which contains an integer and attribute `age` which contains fixed value `seeBirthDate`, even if this seems very easy to do. For performance reasons, XXE does not attempt to be very smart for what it considers to be rare cases.

Note that if you replace attributes `age` and `birthDate` by similar child elements `age` and `birthDate`, XXE will behave exactly the same. See person2.rnc and person2.xml.

Important

XXE cannot make a difference between two child elements having the same name and having different *data-only* content models, *unless these two child elements have both fixed values*.

Example 3. Same attribute name, different fixed values

RELAX NG schema, div.rnc:

```
start = doc-element

doc-element = element doc {
    div-element+
}
div-element = element div {
    (attribute class {"section"}, div-element+) |
    (attribute class {"paragraphs"}, paragraph-element+)
}
paragraph-element = element paragraph { text }
```

Document conforming to the above schema, div.xml:

```
<doc>
  <div class="section">
    <div class="section">
      <div class="paragraphs">
        <paragraph>Paragraph 1.</paragraph>
      </div>
    </div>

    <div class="paragraphs">
      <paragraph>Paragraph 2.</paragraph>
    </div>
  </div>

  <div class="paragraphs">
    <paragraph>Paragraph 3.</paragraph>
  </div>
</doc>
```

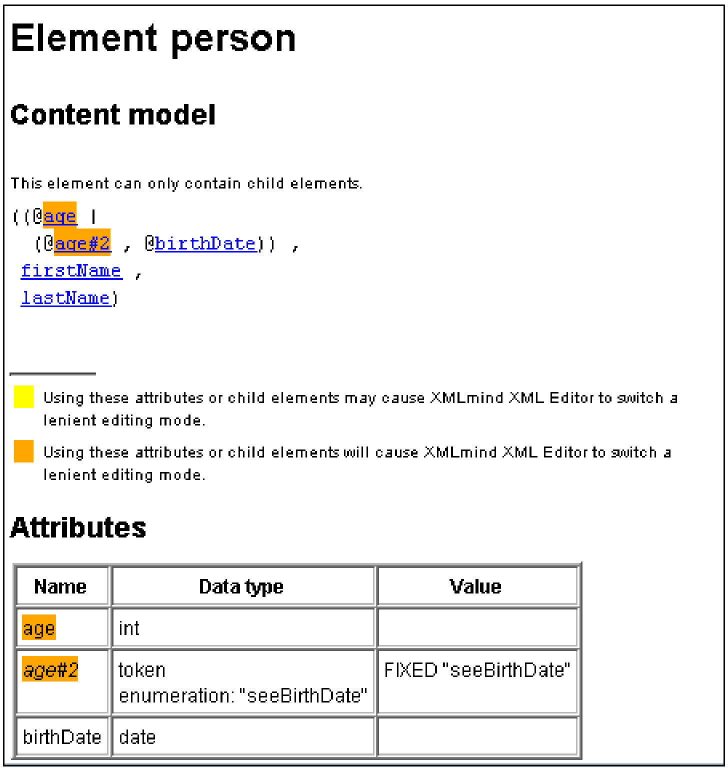
XXE has no problem at all with attribute `class`, because even if there are two attributes named `class` within element `div`, they have different fixed values.

Note that if you replace attribute `class` by similar child element `class`, XXE will behave exactly the same. See div2.rnc and div2.xml.

3.3. Help provided by the "Show Content Model" window

The node path bar is not the only tool in XXE which can help the user recognize attributes and elements which pose problems to the XML editor. The window opened by command Help|Show Content model also displays very useful information.

Figure 4. Person.xml example when element `person` is selected



Element person

Content model

This element can only contain child elements.

```
((@age |
  (@age#2 , @birthDate)) ,
 firstName ,
 lastName)
```

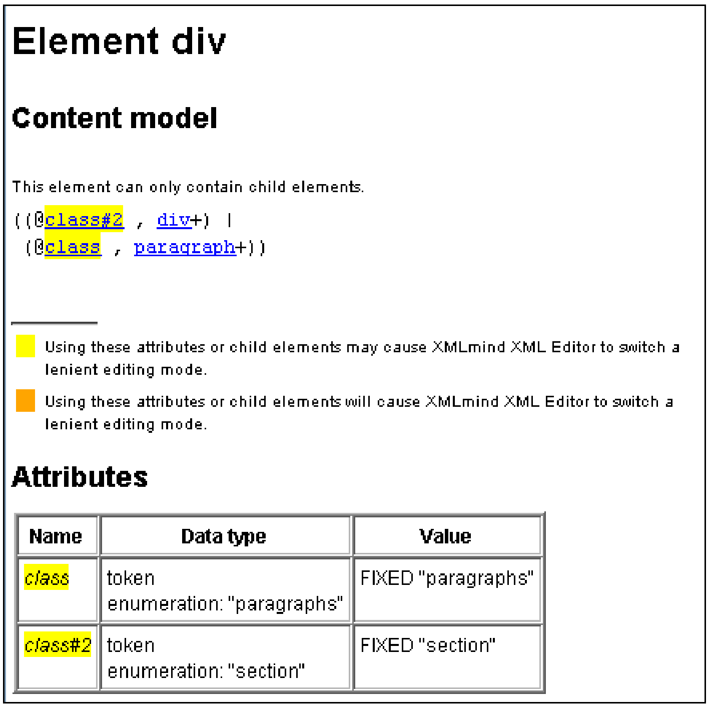
🟨 Using these attributes or child elements may cause XMLmind XML Editor to switch a lenient editing mode.

🟧 Using these attributes or child elements will cause XMLmind XML Editor to switch a lenient editing mode.

Attributes

Name	Data type	Value
age	int	
age#2	token enumeration: "seeBirthDate"	FIXED "seeBirthDate"
birthDate	date	

Figure 5. Div.xml example when element `div` is selected



Element div

Content model

This element can only contain child elements.

```
((@class#2 , div+) |
 (@class , paragraph+))
```

🟨 Using these attributes or child elements may cause XMLmind XML Editor to switch a lenient editing mode.

🟧 Using these attributes or child elements will cause XMLmind XML Editor to switch a lenient editing mode.

Attributes

Name	Data type	Value
class	token enumeration: "paragraphs"	FIXED "paragraphs"
class#2	token enumeration: "section"	FIXED "section"

3.4. Other content models which are not XXE-friendly

Example 4. Not specific to RELAX NG

RELAX NG schema, name.rnc:

```
start = names-element

names-element = element names {
    name-element+
}
name-element = element name {
    element fullName { text } |
    (element firstName { text } & element lastName { text })
}
```

Document conforming to the above schema, name.xml:

```
<names>
  <name><fullName>John Smith</fullName></name>

  <name><firstName>John</firstName><lastName>Smith</lastName></name>

  <name><lastName>Smith</lastName><firstName>John</firstName></name>
</names>
```

XXE allows to replace the `firstName`, `lastName` pair by a `fullName`. Simply select both child elements and use command Edit|Replace. But it is impossible to replace a `fullName` by a `firstName`, `lastName` pair.

The only way to do this is to select the `fullName` to be replaced and then, to use command Edit|Force Deletion. This will force XXE to enter the lenient editing mode. Remember that in this mode, the user is allowed to add any child elements he/she wants, including a `firstName`, `lastName` pair[1].

Note that the above example is not specific to RELAX NG. It is possible to model this kind of content with a DTD or a W3C XML Schema.

The example below is very similar but can only be expressed using a RELAX NG schema. This is the case, because, unlike a DTD and a W3C XML Schema, a RELAX NG schema can be used to specify the places within an element where text nodes may occur.

---

[1] The right approach here is to define two *named element templates* for element `name`, one containing a `fullName` child element and the other containing a `firstName`, `lastName` pair.

Example 5. Specific to RELAX NG

RELAX NG schema, name2.rnc:

```
start = names-element

names-element = element names {
    name-element+
}
name-element = element name {
    text |
    (element firstName { text } & element lastName { text })
}
```

Document conforming to the above schema, name2.xml:

```
<names>
  <name>John Smith</name>

  <name><firstName>John</firstName><lastName>Smith</lastName></name>

  <name><lastName>Smith</lastName><firstName>John</firstName></name>
</names>
```

The situation is worse with the name2.rnc example than with the name.rnc example. It is always allowed to delete a text node and this includes the text node containing "`John Smith`". That is, there is no way to force XXE to enter its lenient mode in order to be able to replace text node "`John Smith`" by a `firstName`, `lastName` pair.

In such case, using named element templates is the only way to cope with such content models. Simply specify two named element templates for element `name`, one containing a text node with a placeholder string and the other containing a `firstName`, `lastName` pair.

4. Command line tools

4.1. rngvalid

Rngvalid is a script which is used to invoke the version of Jing included in XXE (`relaxng.jar`).

Usage: **rngvalid** ?options? *`relax_ng_schema`* ?*`xml_document`* ... *`xml_document`*?

Validate documents *`xml_document`* ... *`xml_document`* against schema *`relax_ng_schema`*. If documents are not specified, just validate the schema.

Options are (what follows is copied from the documentation of Jing):

-c
: The schema uses RELAX NG Compact Syntax.

-e *`enc`*
: Uses the encoding *`enc`* to read the schema.

-f
: Checks that the document is *feasibly valid*. A document is *feasibly valid* if it could be transformed into a valid document by inserting any number of attributes and child elements anywhere in the tree. This is equivalent to transforming the schema by wrapping every `data`, `list`, `element` and `attribute` element in an optional element and then validating against the transformed schema. This option may be useful while a document is still under construction. This option also disables checking that for every IDREF there is a corresponding ID.

-i
: Disables checking of ID/IDREF/IDREFS. By default, Jing enforces the constraints imposed by RELAX NG DTD Compatibility with respect to ID/IDREF/IDREFS.

-t

Prints the time used by Jing for loading the schema and for validation.

Examples:

```
C:\Program Files\XMLmind_XML_Editor> rngvalid demo\bugreport\bugreport.rng

C:\Program Files\XMLmind_XML_Editor> rngvalid demo\bugreport\bugreport.rng \
    demo\bugreport_rng.xml

C:\Program Files\XMLmind_XML_Editor> rngvalid -c doc\rngsupport\samples\target.rnc

C:\Program Files\XMLmind_XML_Editor> rngvalid -c doc\rngsupport\samples\target.rnc \
    doc\rngsupport\samples\target_good.xml doc\rngsupport\samples\target_bad.xml
```

4.2. rngdoc

Rngdoc can be used to generate an HTML reference manual for a RELAX NG schema.

The generated HTML reference manual, organized like "*DocBook: The Definitive Guide*" by Norman Walsh and al., lists all elements and attributes specified in the schema.

This manual is intended to help content authors create instances conforming to a given RELAX NG schema. This manual is not intended to help schema authors document their design.

Usage: **rngdoc** ?options? *`relax_ng_schema`* *`out_dir`*

Generate an HTML reference manual for schema *`relax_ng_schema`* in directory *`out_dir`*.

Options are:

-rnc

Specifies that *`relax_ng_schema`* uses the compact syntax. Default: XML syntax.

-rncencoding *`rnc_charset`*

Specifies the character encoding of the schema using the compact syntax. Default: platform default encoding.

-css *`CSS_URL`*

Specifies which CSS style sheet to use in the generated HTML. Default: no CSS.

-charset *`html_charset`*

Specifies the character encoding of the generated HTML. Default: platform default encoding.

-xxe

Add annotations which are useful when the RELAX NG schema is used by XMLmind XML Editor. Default: don't annotate.

Example:

```
C:\Program Files\XMLmind_XML_Editor> rngdoc -xxe demo\bugreport\bugreport.rng C:\temp
```

5. Missing features

These features will almost certainly be implemented in future versions of XXE:

- An XXE configuration for editing RELAX NG schemas more comfortably[2].
- A format plug-in for the compact syntax (probably based on Trang, an excellent multi-format schema converter written by James Clark).

---

[2] It is already possible to open RELAX NG schemas (using the XML syntax) in XXE and automatically benefit from a full *semantic* validation.

- `include` and `externalRef` elements in RELAX NG schemas are XML catalog aware, but the equivalent notations in the compact syntax are not yet XML catalog aware.