Stage 3 proposal: 13112, RELAX NG For Vocabulary

Contents

Stage 3 proposal: 13112, RELAX NG For Vocabulary

Defines the coding requirements for RELAX NG DITA vocabulary and constraint modules and document type shells.

Champion

Champions: Eliot Kimber, Robert Anderson, Michael Priestly

Tracking information

Event	Date	Links
Stage 1 proposal accepted	5 June 2012	*https://www.oasis-open.org/apps/org/workgroup/dita/download.php?document_id=46183*
Stage 2 proposal submitted	19 June 2013	HTML: *https://www.oasis-open.org/apps/org/workgroup/dita/download.php?document_id=49607*, DITA source: *https://www.oasis-open.org/apps/org/workgroup/dita/download.php?document_id=49606*
Stage 2 proposal discussed	20 August 2013	*http://markmail.org/message/twdyijhlafuwpdwd*
Stage 2 proposal approved	26 August 2013	*http://markmail.org/message/dvxgc6ug7o6h6s2u*
Stage 3 proposal submitted to reviewers	22 Nov 2013	Scott Hudson, Robert Anderson
Stage 3 proposal (this document) submitted		

Approved technical requirements

The requirement is to use RELAX NG as a recognized and codified schema language for DITA vocabulary and constraint definitions. As a schema syntax, RELAX NG offers a number of significant advantages over both DTD and XSD, advantages that make RELAX NG ideally suited to DITA's modular vocabulary architecture. The syntax features of RELAX NG make defining vocabulary modules and the document shells that use them about as easy as it can be, avoiding both the syntactic complexity and unfamiliarity of DTDs and the verbosity of XSD along with XSD's challenges with the redefine feature.

An additional requirement is to use RELAX NG as the master form for all vocabulary definitions from which all other schema formats can be generated, reducing the effort required to maintain multiple forms of DITA modules and document type shells. In short, the requirement is "let the creators and maintainers of DITA modules and shells use the best available tools for the task."

Any schema language used for DITA vocabulary must support defaulted attributes and must support the integration and configuration of vocabulary and constraint modules into working document type shells.

Dependencies or interrelated proposals

Is not dependent on any other proposals.

Modified DTDs

New topic type, vocabularyModuleDesc, which is used within DITA RNG modules. This topic type is a specialization of <topic>. It is required specifically to provide RNG module metadata required to enable generation of other syntax forms of the module. It is not intended for use outside of this context. For that reason, it should not be considered part of the main TC-defined DITA vocabulary, but part of the RNG facility. The markup is documented as part of the RNG facility (see *Module Description Markup for use in DITA-specific RELAX NG grammars* on page 20).

Vocabulary module description topic type.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="urn:oasis:names:tc:dita:rng:vocabularyModuleDesc.rng"
  schematypens="http://relaxng.org/ns/structure/1.0"?>
<grammar xmlns:dita="http://dita.oasis-open.org/architecture/2005/"
  xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  >
 <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
   <moduleTitle>DITA Module Description Module</moduleTitle>
   <headerComment><![CDATA[
=============================================================
                    HEADER
 =============================================================
  MODULE:    DITA Module Description
  VERSION:   1.3
  DATE:      September 2013

 =============================================================

 =============================================================
 SYSTEM:     Darwin Information Typing Architecture (DITA)

 PURPOSE:    Defines markup that must occur at the start of a
             DITA RelaxNG XML-syntax vocabulary module,
             constraint module, or document type shell. Provides
             both documentation and essential~~ly~~ metadata required
             by processes that generate versions of the module
             or shell in other schema syntaxes (DTD, XSD, etc.)

             Note that this schema does define a DITA topic type,
             however, it is in a namespace, which is not
             strictly conforming per the DITA 1.x spec (DITA
             has no way to handle specialized elements that are
             in a namespace because of limitations in the @class
             value syntax). RelaxNG requires that foreign
             elements be in a namespace.

             However, as long as the tags used for the header
             markup do not have a prefix, the header can
             be processed as a non-namespaced document by simply
             omitting the default namespace declaration on
             the root <moduleDesc> element.

             RNG documents can refer to this schema using this
             processing instruction:


 ORIGINAL CREATION DATE:
             September, 2013

             (C) Copyright OASIS Open 2013
             All Rights Reserved.

 =============================================================
 ]]></headerComment>
   <moduleMetadata>
```

```
      <moduleType>topic</moduleType>
      <moduleShortName>vocabularyModule</moduleShortName>
      <modulePublicIds>
        <dtdMod>-//OASIS//ELEMENTS DITA Vocabulary Module Description//EN</dtdMod>
        <dtdEnt>-//OASIS//ENTITIES DITA Vocabulary Module Description//EN</dtdEnt>
        <xsdMod>urn:oasis:names:tc:dita:xsd:vocabularyModuleDescMod.xsd</xsdMod>
        <rncMod>urn:oasis:names:tc:dita:rnc:vocabularyModuleDescMod.rnc</rncMod>
        <rngMod>urn:oasis:names:tc:dita:rng:vocabularyModuleDescMod.rng</rngMod>
      </modulePublicIds>
      <domainsContribution>(topic moduleDesc)</domainsContribution>
    </moduleMetadata>
  </moduleDesc>

   <define name="moduleDesc-info-types">
     <empty/>
   </define>

   <define name="moduleDesc">
     <ref name="moduleDesc.element"/>
   </define>
   <define name="moduleTitle">
     <ref name="moduleTitle.element"/>
   </define>
   <define name="headerComment">
     <ref name="headerComment.element"/>
   </define>
   <define name="moduleMetadata">
     <ref name="moduleMetadata.element"/>
   </define>
   <define name="moduleShortName">
     <ref name="moduleShortName.element"/>
   </define>
   <define name="domainsContribution">
     <ref name="domainsContribution.element"/>
   </define>
   <define name="modulePublicIds">
     <ref name="modulePublicIds.element"/>
   </define>
   <define name="moduleType">
     <ref name="moduleType.element"/>
   </define>
   <define name="shellPublicIds">
     <ref name="shellPublicIds.element"/>
   </define>
   <define name="dtdMod">
     <ref name="dtdMod.element"/>
   </define>
   <define name="dtdEnt">
     <ref name="dtdEnt.element"/>
   </define>
   <define name="dtdShell">
     <ref name="dtdShell.element"/>
   </define>
   <define name="rncMod">
     <ref name="rncMod.element"/>
   </define>
   <define name="rncShell">
     <ref name="rncShell.element"/>
   </define>
   <define name="rngMod">
     <ref name="rngMod.element"/>
   </define>
   <define name="rngShell">
     <ref name="rngShell.element"/>
   </define>
   <define name="xsdMod">
     <ref name="xsdMod.element"/>
   </define>
   <define name="xsdShell">
     <ref name="xsdShell.element"/>
   </define>
   <define name="moduleDesc.content">
     <group>
       <ref name="moduleTitle"/>
       <ref name="headerComment"/>
       <ref name="moduleMetadata"/>
     </group>
   </define>
   <define name="moduleDesc.attributes">
     <empty/>
   </define>
   <define name="moduleDesc.element">
     <a:documentation>Describes a DITA vocabulary or constraint module or a DITA
  document type shell</a:documentation>
     <element name="dita:moduleDesc" ditaarch:longName="Module Description">
       <ref name="moduleDesc.content"/>
       <ref name="moduleDesc.attributes"/>
     </element>
   </define>
   <define name="moduleDesc.attlist" combine="interleave">
     <ref name="moduleDesc.attributes"/>
     <ref name="arch-atts"/>
     <optional>
       <attribute name="domains" a:defaultValue="(topic moduleDesc)"/>
     </optional>
   </define>

   <define name="moduleTitle.content">
     <text/>
   </define>
   <define name="moduleTitle.attributes">
     <empty/>
   </define>
   <define name="moduleTitle.element">
     <a:documentation>The descriptive title of the module</a:documentation>
     <element name="dita:moduleTitle" ditaarch:longName="Module Title">
       <ref name="moduleTitle.content"/>
       <ref name="moduleTitle.attributes"/>
     </element>
   </define>

   <define name="headerComment.content">
     <text/>
   </define>
   <define name="headerComment.attributes">
     <optional>
       <attribute name="xml:space" a:defaultValue="preserve"/>
     </optional>
   </define>
   <define name="headerComment.element">
     <a:documentation>Holds the header comment for the module.
       Spaces and newlines are preserved.</a:documentation>
     <element name="dita:headerComment" ditaarch:longName="Header Comment">
       <ref name="headerComment.content"/>
       <ref name="headerComment.attributes"/>
     </element>
   </define>

   <define name="moduleMetadata.content">
     <group>
       <ref name="moduleType"/>
       <ref name="moduleShortName"/>
       <choice>
         <group>
           <ref name="modulePublicIds"/>
           <ref name="domainsContribution"/>
         </group>
         <ref name="shellPublicIds"/>
       </choice>
     </group>
   </define>
   <define name="moduleMetadata.attributes">
     <empty/>
   </define>
   <define name="moduleMetadata.element">
     <a:documentation>Holds metadata for the module.
     This metadata is used primarily to support the needs
     of transforms that generate other syntax versions
     of the schema, including DTD-syntax modules, XSD-syntax modules,
     and RelaxNG compact syntax modules.</a:documentation>
     <element name="dita:moduleMetadata" ditaarch:longName="Module Metadata">
       <ref name="moduleMetadata.content"/>
       <ref name="moduleMetadata.attributes"/>
     </element>
   </define>

   <define name="moduleType.content">
     <choice>
       <value>attributedomain</value>
       <value>base</value>
       <value>constraint</value>
       <value>elementdomain</value>
       <value>map</value><!-- Map type module -->
       <value>mapshell</value><!-- Topic document type shell -->
       <value>topic</value><!-- Topic type module -->
       <value>topicshell</value><!-- Topic document type shell -->
     </choice>
   </define>
   <define name="moduleType.attributes">
     <optional>
       <attribute name="name" a:defaultValue="moduleType"/>
     </optional>
   </define>
   <define name="moduleType.element">
     <a:documentation>Indicates the type of the module. The types are:
 attributeDomain - An attribute domain module (e.g., deliveryTargetAttDomain)

 base - Part of the base DITA vocabulary definition. These modules are
        special cases. Only OASIS-defined modules may be of type base,
        e.g. commonElements, metaDecl.

 constraint - A constraint module (e.g., strictTaskbodyConstraint)

 elementDomain - An element domain module (e.g., highlightDomain)

 map - A map-type module (e.g., basemap, bookmapMod)

 shell - A document type shell (e.g., concept, basemap, map)

 topic - A topic-type module (e.g., topicMod, conceptMod)
     </a:documentation>
     <element name="dita:moduleType" ditaarch:longName="Module type indicator">
       <ref name="moduleType.content"/>
       <ref name="moduleType.attributes"/>
     </element>
   </define>

   <define name="moduleShortName.content">
     <text/>
   </define>
   <define name="moduleShortName.attributes">
     <optional>
       <attribute name="name" a:defaultValue="moduleShortName"/>
     </optional>
   </define>
   <define name="moduleShortName.element">
     <a:documentation>Indicates the type of the module. The types are:
 attributeDomain - An attribute domain module (e.g., deliveryTargetAttDomain)

 base - Part of the base DITA vocabulary definition. These modules are
        special cases. Only OASIS-defined modules may be of type base,
        e.g. commonElements, metaDecl.
```

```
constraint - A constraint module (e.g., strictTaskbodyConstraint)

elementDomain - An element domain module (e.g., highlightDomain)

map - A map-type module (e.g., basemap, bookmapMod)

shell - A document type shell (e.g., concept, basemap, map)

topic - A topic-type module (e.g., topicMod, conceptMod)
    </a:documentation>
    <element name="dita:moduleShortName" ditaarch:longName="Module Short Name">
      <ref name="moduleShortName.content"/>
      <ref name="moduleShortName.attributes"/>
    </element>
  </define>

  <define name="modulePublicIds.content">
    <interleave>
      <ref name="dtdMod"/>
      <optional>
        <a:documentation>As attribute domains and constraint
          modules do not have .ent
        files, dtdEnt can be omitted for those module types. It must be specified
        for other module types and for document type shells.</a:documentation>
        <ref name="dtdEnt"/>
      </optional>
      <ref name="rncMod"/>
      <ref name="rngMod"/>
      <ref name="xsdMod"/>
    </interleave>
  </define>
  <define name="modulePublicIds.attributes">
    <empty/>
  </define>
  <define name="modulePublicIds.element">
    <a:documentation>Defines the public IDs to be associated
    with each different syntax form of the module. These
    values serve as documentation for how the module
    should be referenced and also enables generation of
    references to the modules from document type shells.

    The elements may occur in any order.
    </a:documentation>
    <element name="dita:modulePublicIds" ditaarch:longName="Module and Shell Public
 IDs">
      <ref name="modulePublicIds.content"/>
      <ref name="modulePublicIds.attributes"/>
    </element>
  </define>

  <define name="shellPublicIds.content">
    <interleave>
      <ref name="dtdShell"/>
      <ref name="rncShell"/>
      <ref name="rngShell"/>
      <ref name="xsdShell"/>
    </interleave>
  </define>
  <define name="shellPublicIds.attributes">
    <empty/>
  </define>
  <define name="shellPublicIds.element">
    <a:documentation>Defines the public IDs to be associated
    with each different syntax form of a document type shell. These
    values serve as documentation for how the shell
    should be referenced and also enable generation of references
    to the modules from document type shells or other modules.

    The elements may occur in any order.
    </a:documentation>
    <element name="dita:shellPublicIds" ditaarch:longName="Shell Public IDs">
      <ref name="shellPublicIds.content"/>
      <ref name="shellPublicIds.attributes"/>
    </element>
  </define>

  <define name="domainsContribution.content">
    <text/>
  </define>
  <define name="domainsContribution.attributes">
    <empty/>
  </define>
  <define name="domainsContribution.element">
    <a:documentation>The module's contribution to the @domains attribute</
a:documentation>
    <element name="dita:domainsContribution"
      ditaarch:longName="Domains attribute contribution">
      <ref name="domainsContribution.content"/>
      <ref name="domainsContribution.attributes"/>
    </element>
  </define>

  <define name="dtdMod.content">
    <text/>
  </define>
  <define name="dtdMod.attributes">
    <empty/>
  </define>
  <define name="dtdMod.element">
    <a:documentation>Defines the public identifier to use for the
      DTD .mod file for this module.</a:documentation>
    <element name="dita:dtdMod" ditaarch:longName="DTD module file public ID">
      <ref name="dtdMod.content"/>
      <ref name="dtdMod.attributes"/>
    </element>
  </define>

  <define name="dtdEnt.content">
    <text/>
  </define>
  <define name="dtdEnt.attributes">
    <empty/>
  </define>
  <define name="dtdEnt.element">
    <a:documentation>Defines the public identifier to use for the
      DTD .ent file for this module.</a:documentation>
    <element name="dita:dtdEnt" ditaarch:longName="DTD entity file public ID">
      <ref name="dtdEnt.content"/>
      <ref name="dtdEnt.attributes"/>
    </element>
  </define>

  <define name="dtdShell.content">
    <text/>
  </define>
  <define name="dtdShell.attributes">
    <empty/>
  </define>
  <define name="dtdShell.element">
    <a:documentation>Defines the public identifier to use for the
      DTD shell file.</a:documentation>
    <element name="dita:dtdShell" ditaarch:longName="DTD shell public ID">
      <ref name="dtdShell.content"/>
      <ref name="dtdShell.attributes"/>
    </element>
  </define>

  <define name="rncMod.content">
    <text/>
  </define>
  <define name="rncMod.attributes">
```

```
    <empty/>
  </define>
  <define name="rncMod.element">
    <a:documentation>Defines the public identifier to use for the
      DTD .mod file for this module.</a:documentation>
    <element name="dita:rncMod" ditaarch:longName="RNC module public ID">
      <ref name="rncMod.content"/>
      <ref name="rncMod.attributes"/>
    </element>
  </define>

  <define name="rncShell.content">
    <text/>
  </define>
  <define name="rncShell.attributes">
    <empty/>
  </define>
  <define name="rncShell.element">
    <a:documentation>Defines the public identifier to use for the
      RNC shell file.</a:documentation>
    <element name="dita:rncShell" ditaarch:longName="RNC shell public ID">
      <ref name="rncShell.content"/>
      <ref name="rncShell.attributes"/>
    </element>
  </define>

  <define name="rngMod.content">
    <text/>
  </define>
  <define name="rngMod.attributes">
    <empty/>
  </define>
  <define name="rngMod.element">
    <a:documentation>Defines the public identifier to use for the
      DTD .mod file for this module.</a:documentation>
    <element name="dita:rngMod" ditaarch:longName="RNG module public ID">
      <ref name="rngMod.content"/>
      <ref name="rngMod.attributes"/>
    </element>
  </define>

  <define name="rngShell.content">
    <text/>
  </define>
  <define name="rngShell.attributes">
    <empty/>
  </define>
  <define name="rngShell.element">
    <a:documentation>Defines the public identifier to use for the
      RNG shell file.</a:documentation>
    <element name="dita:rngShell" ditaarch:longName="RNG shell public ID">
      <ref name="rngShell.content"/>
      <ref name="rngShell.attributes"/>
    </element>
  </define>

  <define name="xsdMod.content">
    <text/>
  </define>
  <define name="xsdMod.attributes">
    <empty/>
  </define>
  <define name="xsdMod.element">
    <a:documentation>Defines the public identifier to use for the
      DTD .mod file for this module.</a:documentation>
    <element name="dita:xsdMod" ditaarch:longName="XSD module public ID">
      <ref name="xsdMod.content"/>
      <ref name="xsdMod.attributes"/>
    </element>
  </define>
  <define name="xsdShell.content">
    <text/>
  </define>
  <define name="xsdShell.attributes">
    <empty/>
  </define>
  <define name="xsdShell.element">
    <a:documentation>Defines the public identifier to use for the XSD shell file.</
a:documentation>
    <element name="dita:xsdShell" ditaarch:longName="XSD shell public ID">
      <ref name="xsdShell.content"/>
      <ref name="xsdShell.attributes"/>
    </element>
  </define>

    <!-- Specialization attributes. Global attributes and class defaults -->

  <define name="moduleDesc.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/topic moduleDesc/moduleDesc "/>
    </optional>
  </define>
  <define name="moduleTitle.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/title moduleDesc/moduleTitle "/
>
    </optional>
  </define>
  <define name="headerComment.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/shortdesc moduleDesc/
headerComment "/>
    </optional>
  </define>
  <define name="moduleMetadata.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/prolog moduleDesc/
moduleMetadata "/>
    </optional>
  </define>
  <define name="moduleType.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/moduleType "/>
    </optional>
  </define>
  <define name="moduleShortName.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/moduleShortName
 "/>
    </optional>
  </define>
  <define name="modulePublicIds.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/modulePublicIds
 "/>
    </optional>
  </define>
  <define name="shellPublicIds.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/shellPublicIds
 "/>
    </optional>
  </define>
  <define name="domainsContribution.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/
domainsContribution "/>
    </optional>
  </define>
  <define name="dtdMod.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/dtdMod "/>
    </optional>
  </define>
  <define name="dtdEnt.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/dtdEnt "/>
    </optional>
  </define>
  <define name="dtdShell.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/dtdShell "/>
    </optional>
  </define>
  <define name="rncMod.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/rncMod "/>
    </optional>
  </define>
  <define name="rncShell.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/rncShell "/>
    </optional>
  </define>
  <define name="rngMod.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/rngMod "/>
    </optional>
  </define>
  <define name="rngShell.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/rngShell "/>
    </optional>
  </define>
  <define name="xsdMod.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/xsdMod "/>
    </optional>
  </define>
  <define name="xsdShell.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="- topic/data moduleDesc/xsdShell "/>
    </optional>
  </define>

</grammar>
```

Vocabulary module topic document type shell:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="vocabularyModuleDesc.rng"
            schematypens="http://relaxng.org/ns/structure/1.0"?>
<grammar xmlns:dita="http://dita.oasis-open.org/architecture/2005/"
```

```
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
   <moduleTitle>DITA Module Description Module</moduleTitle>
   <headerComment><![CDATA[
=============================================================
                    HEADER
 =============================================================
  MODULE:    DITA Module Description Topic Type Shell
  VERSION:   1.3
  DATE:      September 2013

 =============================================================

 =============================================================
 SYSTEM:     Darwin Information Typing Architecture (DITA)

 PURPOSE:    This shell is for validating DITA RNG XML-syntax
             document type shells, vocabulary modules, and
             constraint modules.

             Defines markup that must occur at the start of a
             DITA RelaxNG XML-syntax vocabulary module,
             constraint module, or document type shell. Provides
             both documentation and essential~~ly~~ metadata required
             by processes that generate versions of the module
             or shell in other schema syntaxes (DTD, XSD, etc.)

             Note that this schema does define a DITA topic type,
             however, it is in a namespace, which is not
             strictly conforming per the DITA 1.x spec (DITA
             has no way to handle specialized elements that are
             in a namespace because of limitations in the @class
             value syntax). RelaxNG requires that foreign
             elements be in a namespace.

             However, as long as the tags used for the header
             markup do not have a prefix, the header can
             be processed as a non-namespaced document by simply
             omitting the default namespace declaration on
             the root <moduleDesc> element.

             RNG documents can refer to this schema using this
             processing instruction:

             <?xml-model href="urn:oasis:names:tc:dita:rng:vocabularyModuleDesc.rng"
                         schematypens="http://relaxng.org/ns/structure/1.0"?>

             Parlor trick: This schema validates itself.

 ORIGINAL CREATION DATE:
             September, 2013

             (C) Copyright OASIS Open 2013
             All Rights Reserved.

 =============================================================
 ]]></headerComment>
   <moduleMetadata>
     <moduleType>topicshell</moduleType>
     <moduleShortName>vocabularyModule</moduleShortName>
     <shellPublicIds>
       <dtdShell>-//OASIS//DTD DITA Vocabulary Module Description//EN</dtdShell>
       <xsdShell>urn:oasis:names:tc:dita:xsd:vocabularyModuleDesc.xsd</xsdShell>
       <rncShell>urn:oasis:names:tc:dita:rnc:vocabularyModuleDesc.rnc</rncShell>
       <rngShell>urn:oasis:names:tc:dita:rng:vocabularyModuleDesc.rng</rngShell>
     </shellPublicIds>
   </moduleMetadata>
 </moduleDesc>
  <start>
```

```
    <ref name="grammar.element"/>
  </start>

  <define name="grammar.element">
    <element name="grammar" ns="http://relaxng.org/ns/structure/1.0">
      <group>
        <ref name="moduleDesc"/>
        <zeroOrMore>
          <ref name="anyElementNotDita"/>
        </zeroOrMore>
      </group>
      <zeroOrMore>
        <attribute>
          <anyName/>
        </attribute>
      </zeroOrMore>
    </element>
  </define>

  <define name="anyElementNotDita">
    <element>
      <anyName>
        <except>
           <nsName ns="http://dita.oasis-open.org/architecture/2005/"/>
        </except>
      </anyName>
      <zeroOrMore>
       <choice>
         <attribute>
           <anyName/>
         </attribute>
         <text/>
         <ref name="anyElementNotDita"/>
       </choice>
      </zeroOrMore>
    </element>
  </define>

  <define name="domains-att" combine="interleave">
    <optional>
      <attribute name="domains"
        a:defaultValue="(topic moduleDesc)"/>
    </optional>
  </define>

  <include href="../../base/rng/topicMod.rng"/>

  <include href="vocabularyModuleDescMod.rng"/>

    <!-- Define the any pattern to exclude elements with ID attributes
    from the wildcard and refer them expliceitely  -->
  <define name="any">
    <zeroOrMore>
      <choice>
        <ref name="moduleDesc"/>
        <element>
          <anyName>
            <except>
              <name ns="http://dita.oasis-open.org/architecture/2005/">moduleDesc</
name>
            </except>
          </anyName>
          <zeroOrMore>
            <attribute>
              <anyName/>
            </attribute>
          </zeroOrMore>
          <ref name="any"/>
        </element>
        <text/>
      </choice>
    </zeroOrMore>
  </define>

</grammar>
```

Modified specification documentation

Table 1: Modified Topics

Topic to Be Modified	DITA 1.2 Text	Proposed 1.3 Text
archSpec/base/dita-terminology.dita	**DITA document-type shell** A set of DTD or XSD declarations that implement a DITA document type...	**DITA document-type shell** A set of DTD, XSD, or RELAX NG declarations that implement a DITA document type...
archSpec/base/fileext.dita	**Document-type shell files** • *typename*.dtd • *typename*.xsd	**Document-type shell files** • *typename*.dtd • *typename*.xsd • *typename*.rnc • *typename*.rng
	Add to end of definition list	**RELAX NG structural module files** • *typename*Mod.rnc • *typename*Mod.rng **RELAX NG domain module files** • *typename*Domain.rnc • *typename*Domain.rng **RELAX NG constraint module files** • *constraintname*ConstraintMod.rnc • *constraintname*ConstraintMod.rng
archSpec/base/recognizedconstraintmechanisms.dita	The DITA standard currently recognizes two XML document grammar mechanisms by which conforming DITA vocabulary modules and document types may be constructed: document type declarations (DTDs) and XML Schema declarations (XSDs). This specification defines implementation requirements for both of these document constraint mechanisms.	The DITA specification currently recognizes three XML document grammar mechanisms by which conforming DITA vocabulary modules and document types may be constructed: document type declarations (DTDs), XML Schema declarations (XSDs), and RELAX NG grammars. This specification defines implementation requirements for all of these document constraint mechanisms. The OASIS DITA Technical Committee recognizes that other XML grammar languages might provide similar modularity and extensibility mechanisms. However, the Technical Committee has not yet defined implementation requirements for those languages so their conformance cannot be determined.

Topic to Be Modified	DITA 1.2 Text	Proposed 1.3 Text
	The OASIS DITA Technical Committee recognizes that other XML grammar languages might provide similar modularity and extensibility mechanisms. However, the Technical Committee has not yet defined implementation requirements for those languages so their conformance cannot be determined.	Of these three document constraint mechanisms, RELAX NG grammars offer the easiest-to-use syntax and the most expressive constraints. For this reason, the RELAX NG definitions of the standard DITA vocabularies are the authoritative versions from which the DTD and XSD versions are automatically generated. Open-source tools are available for generating conforming DTD-syntax and XSD-syntax document type shells, vocabulary modules, and constraint modules from RELAX NG grammars that conform to the coding requirements defined in this specification.
New topic: Using RELAX NG for DITA document types, vocabulary modules, and constraint modules		*Using RELAX NG for DITA document type shells, vocabulary modules, and constraint modules* on page 19 Insert as second child of archSpec/base/ditaspecialization.dita.
New topic, generation of DTD and XSD schemas from RELAX NG		*Generating DTD and XSD Schemas from RELAX NG* on page 20 Insert as third child of archSpec/base/ditaspecialization.dita.
archSpec/base/createCustomDocType.dita	Thus, DITA DOES NOT require that conforming DITA documents have an associated DTD, XSD, or other formal document type definition as long as all required attributes are explicit in document instances. However, most DITA documents have an associated DTD or XML schema document by which the documents can be validated using normal XML processors and that can provide default values for the @domains and @class attributes, in particular. In addition, while the DITA specification only defines coding requirements for	Thus, DITA DOES NOT require that conforming DITA documents have an associated DTD, XSD, RELAX NG, or other formal document type definition as long as all required attributes are explicit in document instances. However, most DITA documents have an associated DTD, RELAX NG grammar, or XML schema document by which the documents can be validated using normal XML processors and that can provide default values for the @domains and @class attributes, in particular. In addition, while the DITA specification only defines coding requirements for DTDs, RELAX NG grammars, and XML schema documents, conforming DITA documents MAY use other document type constraint languages, such as Schematron.

Topic to Be Modified	**DITA 1.2 Text**	**Proposed 1.3 Text**
	DTDs and XML schema documents, conforming DITA documents MAY use other document type constraint languages, such as RELAX NG or Schematron.	
	For example, a shell document type that is an unmodified copy of the OASIS-provided topic document type shell (topic.dtd or topic.xsd)	For example, a shell document type that is an unmodified copy of the OASIS-provided topic document type shell (topic.rng, topic.rnc, topic.dtd, or topic.xsd)
New topic		*RELAX NG document-type shell: Coding requirements* on page 23
archSpec/base/classatt.dita	When the @class attribute is declared in a DTD or XSD	When the @class attribute is declared in a DTD, XSD, or RELAX NG grammar,
archSpec/base/foreigncontentspec.dita	New example	Add new example of RELAX NG foreign specialization: *Specializing foreign or unknown content* on page 37
New topic		*RELAX NG grammar specialization module coding requirements* on page 28
New topic		*Constraint module RELAX NG coding requirements* on page 36
For the Technical Content section, all the topics have trailing "Modules" sections that must be updated to reflect the RELAX NG modules. I have not bothered to show the original except for Concept, which establishes the pattern.		
archSpec/technicalContent/ dita_concept_topic.dita	In section titled "Modules": The following DITA modules are provided for the concept topic: concept.mod, concept.ent (DTD) conceptMod.xsd, conceptGrp.xsd (Schema)	The following DITA modules are provided for the concept topic: concept.mod, concept.ent (DTD), conceptMod.xsd, conceptGrp.xsd (Schema), conceptMod.rnc (RELAX NG compact syntax), conceptMod.rng (RELAX NG XML syntax)
archSpec/technicalContent/ dita_reference_topic.dita		reference.mod, reference.ent (DTD), referenceMod.rnc (RELAX NG compact syntax) referenceMod.rng (RELAX NG XML syntax) referenceMod.xsd, referenceGrp.xsd (Schema)

Topic to Be Modified	DITA 1.2 Text	Proposed 1.3 Text
archSpec/technicalContent/ dita_generic_task_topic.dita		task.mod, task.ent(DTD), taskMod.rnc (RELAX NG compact syntax) taskMod.rng (RELAX NG XML syntax) taskMod.xsd, taskGrp.xsd (Schema)
archSpec/technicalContent/ dita_task_topic.dita		task.mod. task.ent, strictTaskbody constraint (DTD) taskMod.rnc, strictTaskBodyConstraint.rnc (RELAX NG compact syntax) taskMod.rng, strictTaskBodyConstraint.rng (RELAX NG XML syntax) taskMod.xsd, taskGrp.xsd, strictTaskbodyConstraintMod.xsd (Schema)
archSpec/technicalContent/ dita_machinerytask_topic.dita		machineryTask.dtd (DTD), machineryTaskbodyConstraint.mod machineryTask.rnc, machineryTaskbodyConstraint.rnc (RELAX NG compact syntax) machineryTask.rng, machineryTaskbodyConstraint.rng (RELAX NG XML syntax) machineryTask.xsd, machineryTaskbodyConstraintMod.xsd, machineryTaskbodyConstraintIntMod.xsd (Schema)
archSpec/technicalContent/ dita_glossary_topic.dita		glossentry.dtd, glossentry.ent, glossentry.mod (DTD) glossentry.rnc, glossentryMod.rnc (RELAX NG compact syntax) glossentry.rng, glossentryMod.rng (RELAX NG XML syntax) glossentryMod.xsd, glossentryGrp.xsd (Schema)
archSpec/technicalContent/ dita_glossarygroup_topic.dita		glossgroup.dtd, glossgroup.ent, glossgroup.mod (DTD) glossgroup.rnc, glossgroupMod.rnc (RELAX NG compact syntax) glossgroup.rng, glossgroupMod.rng (RELAX NG XML syntax) glossgroup.xsd, (Schema)
archSpec/technicalContent/ dita_spec_intro_bookmap.dita		bookmap.dtd, bookmap.ent, bookmap.mod (DTD) bookmap.rnc, bookmapMod.rnc (RELAX NG compact syntax)

Topic to Be Modified	DITA 1.2 Text	Proposed 1.3 Text
		bookmap.rng, bookmapMod.rng (RELAX NG XML syntax) bookmap.xsd, bookmapGrp.xsd, bookmapMod.xsd (Schema)

Using RELAX NG for DITA document type shells, vocabulary modules, and constraint modules

The RELAX NG specification defines two syntaxes for RELAX NG grammars: the XML syntax and the compact syntax. The two syntaxes are functionally equivalent and either syntax may be reliably converted into the other (e.g., using the open-source jing tool). However, the XML syntax, because it is XML based, allows the use of foreign markup within RELAX NG grammars. The DITA coding requirements depend on this feature of RELAX NG in order to capture metadata about document type shells and modules required in order to generate DTD- and XSD-syntax versions of the files that themselves conform to the DITA coding requirements. In addition, the foreign vocabulary feature can be used to include Schematron rules directly in RELAX NG grammars. Schematron rules can check rules that are not expressible with RELAX NG directly or that would be inconvenient to express.

In addition, the RELAX NG XML syntax provides a general grouping element type, <div>, that allows for arbitrary organization and grouping of patterns within grammar documents. Such grouping tends to make the grammar documents easier to work with, especially in XML-aware editors. The use or non-use of the RELAX NG <div> element cannot affect the meaning of the patterns defined in a RELAX NG schema.

For these reasons, the authoritative version of all OASIS-defined DITA document type shells, vocabulary modules, and constraint modules use the RELAX NG XML syntax. The DITA RELAX NG coding requirements are defined for the RELAX NG XML syntax. Conforming RELAX NG compact syntax modules MUST reflect the same file naming and organization requirements as the RELAX NG XML syntax requirements, substituting ".rnc" for ".rng" as the filename extension. By definition, RELAX NG compact syntax modules generated from conforming RELAX NG XML syntax modules such that there is a one-to-one correspondence between XML files and compact syntax files are conforming. Likewise, generation of XML syntax modules from conforming compact syntax modules should result in minimally-conforming XML syntax modules (such modules will lack the DITA-specific <moduleDesc> element for which there is no defined compact syntax equivalent, see *Generating DTD and XSD Schemas from RELAX NG* on page 20).

The DITA use of RELAX NG depends on the companion *RELAX NG DTD Compatibility* specification, which provides a mechanism for defining default attribute values, as well as embedded documentation. Processors that use RELAX NG to get the full infoset for DITA documents for which not all required attributes are present in document instances must implement the DTD compatibility specification in order to get default attribute values (e.g., values for @domains and @class). The DTD compatibility specification defines the namespace name "http://relaxng.org/ns/compatibility/annotations/1.0" which is bound to the prefix "a" by convention~~, e.g.~~ "<a:documentation>" refers to the <documentation> element type from the "http://relaxng.org/ns/compatibility/annotations/1.0" namespace as defined by the DTD compatibility specification.

RELAX NG grammars for DITA document type shells, vocabulary modules, and constraint modules ("DITA grammars") MAY use the <a:documentation> element anywhere foreign elements are allowed by RELAX NG. DITA grammars MAY use <div> to group pattern declarations without restriction.

RELAX NG grammars for DITA markup MAY include embedded Schematron rules or any other foreign vocabulary. Processors MAY ignore any foreign vocabularies within DITA grammars not in the "http://relaxng.org/ns/compatibility/annotations/1.0" or "http://dita.oasis-open.org/architecture/2005/" namespaces.

Each section of a DITA grammar document is either introduced by a comment or represented by a <div> element with child <a:documentation> element containing the section header. Grammars SHOULD use these divs or comments to identify each section of the grammar. Sections SHOULD occur in the order specified in these coding requirements. Grammars SHOULD have an initial set of comments or a <moduleMetadata> element that describes the grammar document and indicates the URNs or absolute URLs by which the grammar should be referenced in

RELAX NG grammar references (see *Module Description Markup for use in DITA-specific RELAX NG grammars* on page 20).

Section titles using <div> SHOULD have the form:

```
<div>
  <a:documentation>SECTION TITLE</a:documentation>
 ...
</div>
```

Header comments SHOULD have the form:

```
<!-- ============================================================= -->
<!--                  SECTION TITLE                                -->
<!-- ============================================================= -->
```

RELAX NG is capable of expressing constraints that are more precise than is possible with either DTDs or XSDs. For example, RELAX NG patterns can be context specific such that the same element type may allow different content or attributes in different contexts. Where generation of DTD or XSD schema modules from RELAX NG modules is a requirement care must be taken to avoid RELAX NG features that cannot be translated into DTD or XSD constructs. When RELAX NG is used directly for DITA document validation, the document type shells for those documents MAY integrate constraint modules that use the full power of RELAX NG to enforce constraints that cannot be enforced by DTDs or XSD schemas.

Generating DTD and XSD Schemas from RELAX NG

RELAX NG XML syntax document type shells and modules can be used to generate DTD- and XSD-syntax document type shells and modules that conform to the DITA coding requirements when the RELAX NG grammar document contains DITA-specific metadata that defines the module type, module header comment, and public identifiers for the components to be generated. All RELAX NG grammars defined by the DITA specification include this metadata.

Of the three DITA-recognized document constraint languages, RELAX NG provides the simplest syntax for specifying the integration of vocabulary and constraint modules into document type shells. This makes RELAX NG ideally suited as the base format for defining DITA vocabulary. However, many, if not most DITA-aware tools require the use of DTDs or XSD schemas as of version 1.3 of the DITA specification. Thus there is a practical requirement to generate DTDs and XSD schemas from RELAX NG.

The generation of DTD and XSD modules requires additional metadata that is not inherent in the grammar definition:

- The module type (document type shell, vocabulary module, or constraint module)
- The header comment for the file
- The module short name for vocabulary and constraint modules
- The public identifiers to use for each component file to be generated
- The @domains attribute contribution for vocabulary and constraint modules

This metadata can be defined using the <moduleDesc> element within a RELAX NG <grammar> element.

Module Description Markup for use in DITA-specific RELAX NG grammars

The <moduleDesc> element enables specification of the metadata required in order to generate from RELAX NG grammars DTD and XSD components that conform to the DITA coding requirements.

The use of <moduleDesc> is not required. However, when it is used, the <moduleDesc> element MUST be specified as a direct child of the RELAX NG <grammar> element and SHOULD be the first child of <grammar>. The <moduleDesc> element functions as a foreign element as defined in the RELAX NG specification. As a foreign element, it is required by RELAX NG rules to be in a namespace that is different from any RELAX NG namespaces. The <moduleDesc> element is defined as a specialization of <topic>. Because it must be in a namespace, it cannot be a strictly-conforming DITA topic as DITA elements may not be in a namespace. However, if the <moduleDesc> element is processed in terms of the @class attributes it will be recognized as a DITA topic and otherwise conforms to all rules for conforming topic specializations.

The <moduleDesc> element type is defined in the vocabulary module file `vocabularyModuleDescMod.rng`.

Namespace

The <moduleDesc> element is in the DITA architecture namespace, "http://dita.oasis-open.org/architecture/2005/"

<moduleDesc> element type

<moduleDesc>

Contains the module description metadata. Subelements:

<moduleTitle>

Provides a descriptive title for the module or document type shell. Content is text.

<headerComment>

Contains the header comment for the module. The content is text with whitespace preserved (@xml:space value of "preserve"). The header comment is used as the header for all generated components.

<moduleMetadata>

Contains additional metadata for the module or document type shell. Subelements:

<moduleType>

Indicates the module type. Content is one of the following keywords:

attributedomain

The grammar defines an attribute domain

base

The grammar is one of the DITA base modules. Base modules may only be defined by the DITA Technical Committee.

constraint

The grammar defines a constraint module.

elementdomain

The grammar defines an element domain.

map

The grammar defines a map type.

mapshell

The grammar is a document type shell for a map type.

topic

The grammar defines a topic type.

topicshell

The grammar is a document type shell for a topic type.

<modulePublicIds>

For vocabulary and constraint modules, defines the public IDs for files generated from the grammar. Each subelement specifies the public identifier to use for a specific generated file. Subelements:

<dtdMod>

The public identifier for the DTD-syntax .mod file to be generated from the grammar.

<dtdEnt>

The public identifier for the DTD-syntax .ent file to be generated from the grammar. Required for element domain modules, not used for attribute domain modules.

<rncMod>

The public identifier for the RELAX NG compact syntax (RNC) module file to be generated from the grammar.

<rngMod>

The public identifier for the RELAX NG XML syntax (RNG) module file (the file that contains the <moduleMetadata> element).

<xsdMod>

The public identifier for the XSD schema module file to be generated from the grammar.

<domainsContribution>

Required for modules, not used for shells. Defines the @domains attribute contribution for the module, e.g. "(topic hi-d)". This is the value that MUST be added to the @domains value for root map or topic types that integrate the module.

<shellPublicIds>

For document type shells, defines the public IDs for files generated from the grammar. Each subelement specifies the public identifier to use for a specific generated file. Subelements:

<dtdShell>

The public identifier for the DTD document type shell to be generated from the grammar.

<rncShell>

The public identifier for the RELAX NG compact syntax (RNC) document type shell to be generated from the grammar.

<rngShell>

The public identifier for the RNG document type shell (the file that contains the <moduleMetadata> element)

<xsdShell>

The public identifier for the XSD schema document type shell to be generated from the grammar.

Example RELAX NG grammars with module metadata

A typical vocabulary module, in this case, a topic type:

```
<grammar xmlns:dita="http://dita.oasis-open.org/architecture/2005/"
  xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  >
 <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
   <moduleTitle>DITA Module Description Module</moduleTitle>
   <headerComment><![CDATA[
=============================================================
                    HEADER
 =============================================================
  MODULE:    DITA Module Description
  VERSION:   1.3
  DATE:      September 2013

  ...
]]></headerComment>
   <moduleMetadata>
     <moduleType>topic</moduleType>
     <moduleShortName>vocabularyModule</moduleShortName>
     <modulePublicIds>
       <dtdMod>-//OASIS//ELEMENTS DITA Vocabulary Module Description//EN</dtdMod>
       <dtdEnt>-//OASIS//ENTITIES DITA Vocabulary Module Description//EN</dtdEnt>
       <xsdMod>urn:oasis:names:tc:dita:xsd:vocabularyModuleDescMod.xsd</xsdMod>
       <rncMod>urn:oasis:names:tc:dita:rnc:vocabularyModuleDescMod.rnc</rncMod>
       <rngMod>urn:oasis:names:tc:dita:rng:vocabularyModuleDescMod.rng</rngMod>
     </modulePublicIds>
```

```
      <domainsContribution>(topic myTopicType)</domainsContribution>
    </moduleMetadata>
  </moduleDesc>

  ...

</grammar>
```

A typical document type shell, in this case, a topic shell:

```
<grammar xmlns:dita="http://dita.oasis-open.org/architecture/2005/"
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>DITA Module Description Module</moduleTitle>
    <headerComment><![CDATA[
=============================================================
                    HEADER
 =============================================================
  MODULE:    DITA Module Description Topic Type Shell
  VERSION:   1.3
  DATE:      September 2013

  ...

]]></headerComment>
    <moduleMetadata>
      <moduleType>topicshell</moduleType>
      <moduleShortName>vocabularyModule</moduleShortName>
      <shellPublicIds>
        <dtdShell>-//OASIS//DTD DITA Vocabulary Module Description//EN</dtdShell>
        <xsdShell>urn:oasis:names:tc:dita:xsd:vocabularyModuleDesc.xsd</xsdShell>
        <rncShell>urn:oasis:names:tc:dita:rnc:vocabularyModuleDesc.rnc</rncShell>
        <rngShell>urn:oasis:names:tc:dita:rng:vocabularyModuleDesc.rng</rngShell>
      </shellPublicIds>
    </moduleMetadata>
  </moduleDesc>

  ...

</grammar>
```

RELAX NG document-type shell: Coding requirements

A document type shell integrates one or more topic type or map type modules, zero or more domain modules, and zero or more constraint modules.

RELAX NG document type shells MAY NOT directly declare element types or attributes. A RELAX NG document type shell MUST conform to the following coding requirements as well as the requirements defined in *Using RELAX NG for DITA document type shells, vocabulary modules, and constraint modules* on page 19.

RELAX NG modules are self-integrating such that document type shells need only include vocabulary modules, there is no separate specification required to integrate domain and nested topic elements into base content models. Likewise, constraint modules simply override the patterns they constrain directly in the constraint module itself. However, the @domains attribute default value cannot be automatically constructed using RELAX NG facilities. Thus, the @domains attribute value must be directly specified in the document type shell.

Note: For modules that use the <moduleDesc> element, the module's @domains attribute contribution should be in the <domainsContribution> element. When this is the case, it should be possible for tools to automatically construct the @domains value in document type shells as an aid to document type shell authoring or automatic generation.

Constraint modules are used by importing the constraint module rather than the module the constraint modifies. Constraint modules refer to the base module to be constrained and redefine patterns as needed to implement the constraint. In addition, you can disallow base types extended by domains by overriding the base type's pattern in the document type shell within the reference to the domain module for the domain. In this case, the constraint represented by the pattern redefinition MUST be declared in the @domains attribute. The @domains contribution for the constraint may be documented using a <domainsContribution> element from the DITA namespace. For example:

```
...
<div>
  <a:documentation>MODULE INCLUSIONS</a:documentation>
  <include href="topicMod.rng"/>
  <include href="hazardstatementDomainMod.rng"/>
  <include href="highlightDomainMod.rng">
    <domainsContribution xmlns="http://dita.oasis-open.org/architecture/2005/"
      >(topic hi-d-noUnderline-c)</domainsContribution>
    <define name="u">
      <notAllowed></notAllowed>
    </define>
  </include>
  <include href="indexingDomainMod.rng"/>
  <include href="utilitiesDomainMod.rng"/>
</div>
...
```

Root element declaration

Document type shells MUST use the RELAX NG start declaration to specify the allowed root element defined by the document type, either the root topic type or root map type defined by the document type shell. The start declaration MUST specify exactly one allowed root element as a reference to the `*tagname*.element` pattern for the root element. The root element declaration SHOULD start with the header "ROOT ELEMENT DECLARATION".

For example:

```
<div>
  <a:documentation>ROOT ELEMENT DECLARATION</a:documentation>
  <start combine="choice">
    <ref name="topic.element"/>
  </start>
</div>
```

Module inclusions

The document type shell must include at least the module for the map or topic type the shell is configuring. The module inclusion section SHOULD start with the header "MODULE INCLUSIONS".

For example:

```
<div>
  <a:documentation>MODULE INCLUSIONS</a:documentation>
  <include href="topicMod.rng"/>
  <include href="highlightDomainMod.rng"/>
  <include href="utilitiesDomainMod.rng"/>
  <include href="indexingDomainMod.rng"/>
  <include href="hazardstatementDomainMod.rng"/>
</div>
```

ID-defining-elements override

In order to support patterns for elements that declare the @id attribute directly, rather than through the universal attributes or ID attributes patterns (pattern "univ-atts" or pattern "id-atts"), the document type shell grammar must define a pattern named "any" that configures the attribute list pattern for those elements. This section SHOULD start with the header "ID-DEFINING-ELEMENT OVERRIDES".

The general structure of this pattern is:

```
<div>
  <a:documentation>ID-DEFINING-ELEMENT OVERRIDES</a:documentation>
  <define name="any">
    <zeroOrMore>
      <choice>
        <ref name="tagname.element"/>
        <element>
          <anyName>
            <except>
              <name>tagname</name>
            </except>
          </anyName>
          <zeroOrMore>
            <attribute>
              <anyName/>
            </attribute>
          </zeroOrMore>
          <ref name="any"/>
        </element>
        <text/>
      </choice>
    </zeroOrMore>
  </define>
</div>
```

Where the <ref> and <name> elements are repeated for each element type used from the shell that defines the @id attribute directly. For topic types this will be any topic types used from the shell (the root topic type and any allowed nested topic types). For maps this will be the map element type (<map> or specializations of <map>) and the <anchor> element type or any specialization of <anchor>. Domains and constraint modules may also define patterns that define the @id attribute directly.

Sample topic type shell

Strict task document type shell:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="../../../checkShell.sch" schematypens="http://purl.oclc.org/dsdl/
schematron"?>
<!-- =============================================================
  MODULE:    DITA Task RNG
  VERSION:   1.2
  DATE:      November 2010
  ============================================================= -->
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <headerComment>
</headerComment>
   <moduleMetadata>
     <moduleType>topicshell</moduleType>
     <moduleShortName>vocabularyModule</moduleShortName>
     <shellPublicIds>
       <dtdShell>-//OASIS//DTD DITA Topic//EN</dtdShell>
       <xsdShell>urn:oasis:names:tc:dita:xsd:topic.xsd</xsdShell>
       <rncShell>urn:oasis:names:tc:dita:rnc:topic.rnc</rncShell>
       <rngShell>urn:oasis:names:tc:dita:rng:topic.rng</rngShell>
     </shellPublicIds>
   </moduleMetadata>
 </moduleDesc>

<!-- ============================================================= -->
<!--                  ROOT ELEMENT DECLARATION                     -->
<!-- ============================================================= -->
  <start>
    <ref name="task.element"/>
  </start>
```

```
<!-- ============================================================= -->
<!--                  DITA DOMAINS ATTRIBUTE                       -->
<!-- ============================================================= -->
  <define name="domains-att" combine="interleave">
    <optional>
      <attribute name="domains"
        a:defaultValue="
        (topic task strictTaskbody-c)
        (topic task)
        (topic abbrev-d)
        (topic hazard-d)
        (topic hi-d)
        (topic indexing-d)
        (topic pr-d)
        (topic sw-d)
        (topic ui-d)
        (topic ut-d)
        "/>
    </optional>
  </define>

<!-- ============================================================= -->
<!--                  MODULE INCLUSIONS                            -->
<!-- ============================================================= -->
  <include href="strictTaskbodyConstraintMod.rng"/>
  <include href="abbreviateDomainMod.rng"/>
  <include href="../../base/rng/hazardstatementDomainMod.rng"/>
  <include href="../../base/rng/highlightDomainMod.rng"/>
  <include href="../../base/rng/indexingDomainMod.rng"/>
  <include href="programmingDomainMod.rng"/>
  <include href="softwareDomainMod.rng"/>
  <include href="uiDomainMod.rng"/>
  <include href="../../base/rng/utilitiesDomainMod.rng"/>

<!-- ============================================================= -->
<!--                  ID-DEFINING-ELEMENT OVERRIDES                -->
<!-- ============================================================= -->
  <define name="any">
    <zeroOrMore>
      <choice>
        <ref name="task.element"/>
        <element>
          <anyName>
            <except>
              <name>topic</name>
              <name>task</name>
            </except>
          </anyName>
          <zeroOrMore>
            <attribute>
              <anyName/>
            </attribute>
          </zeroOrMore>
          <ref name="any"/>
        </element>
        <text/>
      </choice>
    </zeroOrMore>
  </define>
</grammar>
```

Sample map type shell

Base map document type shell:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?xml-model href="../../../checkShell.sch" schematypens="http://purl.oclc.org/dsdl/
schematron"?>
<?xml-model href="../../vocabularyModuleDesc/rng/vocabularyModuleDesc.rng"
                         schematypens="http://relaxng.org/ns/structure/1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0" xmlns:a="http://relaxng.org/
ns/compatibility/annotations/1.0">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>DITA Highlight Domain</moduleTitle>
    <headerComment
 xml:space="preserve">=================================================================
                              HEADER
=============================================================
 MODULE:    DITA Base MAP DTD (only base domains)
 VERSION:   1.2
 DATE:      April 2010

=============================================================

=============================================================
                   PUBLIC DOCUMENT TYPE DEFINITION
                       TYPICAL INVOCATION

Refer to this file by the following public identifier or an
     appropriate system identifier:
PUBLIC "-//OASIS//DTD DITA Base Map//EN"
     Delivered as file "basemap.dtd"

The public ID above refers to the latest version of this DTD.
     To refer to this specific version, you may use this value:
PUBLIC "-//OASIS//DTD DITA 1.2 Base Map//EN"

=============================================================
SYSTEM:     Darwin Information Typing Architecture (DITA)

PURPOSE:    DTD to describe DITA maps

ORIGINAL CREATION DATE:
            April 2010

            (C) Copyright OASIS Open 2010
            All Rights Reserved.

 UPDATES:
   2010.09.20 RDA: Add topic-based domains
=============================================================
</headerComment>
    <moduleMetadata>
      <moduleType>mapshell</moduleType>
      <moduleShortName>basemap</moduleShortName>
      <shellPublicIds>
        <dtdShell>-//OASIS//DTD DITA Base Map//EN</dtdShell>
        <xsdShell>urn:oasis:names:tc:dita:xsd:basemap.xsd</xsdShell>
        <rncShell>urn:oasis:names:tc:dita:rng:basemap.rnc</rncShell>
        <rngShell>urn:oasis:names:tc:dita:rng:basemap.rng</rngShell>
      </shellPublicIds>
    </moduleMetadata>
  </moduleDesc>

  <div>
    <a:documentation>ROOT ELEMENT DECLARATION</a:documentation>
    <start>
      <ref name="map.element"/>
    </start>
  </div>
  <div>
    <a:documentation>DITA DOMAINS ATTRIBUTE</a:documentation>

    <define name="domains-att">
      <optional>
        <attribute name="domains"
          a:defaultValue="
```

```
         (topic delay-d)
         (topic hazard-d)
         (topic hi-d)
         (topic indexing-d)
         (map mapgroup-d)
         (topic ut-d)
         "/>
       </optional>
     </define>


   </div>
   <div>
     <a:documentation>MODULE INCLUSIONS</a:documentation>

     <include href="mapMod.rng"/>
     <include href="delayResolutionDomainMod.rng"/>
     <include href="indexingDomainMod.rng"/>
     <include href="mapGroupMod.rng"/>
     <include href="highlightDomainMod.rng"/>
     <include href="utilitiesDomainMod.rng"/>
     <include href="hazardstatementDomainMod.rng"/>
   </div>
   <div>
     <a:documentation>ID-DEFINING-ELEMENT OVERRIDES</a:documentation>

     <define name="any">
       <zeroOrMore>
         <choice>
           <ref name="map.element"/>
           <ref name="anchor.element"/>
           <element>
             <anyName>
               <except>
                 <name>map</name>
                 <name>anchor</name>
               </except>
             </anyName>
             <zeroOrMore>
               <attribute>
                 <anyName/>
               </attribute>
             </zeroOrMore>
             <ref name="any"/>
           </element>
           <text/>
         </choice>
       </zeroOrMore>
     </define>
   </div>
 </grammar>
```

RELAX NG grammar specialization module coding requirements

To be extensible and backward compatible, DITA requires that RELAX NG implementations of structural and domain specialization modules conform to well-defined implementation (coding) requirements.

These coding requirements implement the specialization architecture with the capabilities and within the limitations of the RELAX NG grammar. Structural specializations, element domain specializations, and attribute domain specializations MUST conform to these requirements as well as the requirements defined in *Using RELAX NG for DITA document type shells, vocabulary modules, and constraint modules* on page 19.

Unlike DTD and XSD-schema modules, RELAX NG modules are self integrating such that document type shells need only include the module and include the module's @domains contribution in the declaration of the @domains attribute default value.

All vocabulary and constraint modules must document their @domains attribute contribution. This MAY be done using the <domainsContribution> element within a <moduleDesc> element or using a normal XML comment or <a:documentation> element. Grammars that are intended to be used for automatic generation of DTD- and XSD-syntax modules MUST use the <domainsContribution> element.

General element type declaration requirements

Structural and element domain vocabulary modules MUST reflect the same coding requirements for element type declarations.

Module names

Each vocabulary module has a short name that is used to construct file names, pattern names, and other names used in associated declarations. Modules MAY also have abbreviated names that further shorten the short name, for example "sw" for the "software" domain, where "software" is the short name and "sw" is the abbreviated name.

For structural modules, the module name MUST be the element type name of the top-level topic or map type defined by the module, such as "concept" or "bookmap".

For element domain modules, the module name MUST be a name that reflects the subject domain to which the domain applies, such as "highlight" or "software". Domain module names should be sufficiently unique that they are unlikely to conflict with any other domains.

Module files

A RELAX NG vocabulary module consists of a single module file.

For structural modules, the file name is the module name plus "Mod" plus an extension of either ".rng" for RELAX NG XML syntax modules or ".rnc" for RELAX NG compact syntax modules, e.g., "conceptMod.rng", "glossentryMod.rnc".

For domain modules, the file name is the domain name plus `DomainMod` plus an extension of either ".rng" for RELAX NG XML syntax modules or ".rnc" for RELAX NG compact syntax modules, e.g. `highlightDomainMod.rng`, `programmingDomainMod.rnc`.

Element definitions

A structural or element domain vocabulary module MUST contain a declaration for each specialized element type named by the module. While the XML standard allows content models to refer to undeclared element types, all element types named in content models or attribute list declarations within a vocabulary module MUST have a RELAX NG element declaration, in one of:

- The vocabulary module
- A base module of which the vocabulary module is a direct or indirect specialization
- A required domain module (if the vocabulary module is a structural module).

The specialized elements MUST follow the rules of the architecture in defining content models and attributes.

The element type patterns are organized into the following sections:

Element type name patterns

This section SHOULD have the section heading "ELEMENT TYPE NAME PATTERNS"

```
<div>
  <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
  ...
</div>
```

For each element type declared in the vocabulary module there MUST be a pattern whose name is the element type name and whose content is a reference to the element type's ".element" pattern. For example:

```
<div>
  <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
  <define name="b">
    <ref name="b.element"/>
  </define>
  ...
</div>
```

The element type name pattern provides a layer of abstraction that facilitates redefinition. The element type name patterns are referenced from content model and domain extension patterns. Specialization modules can redeclare the patterns to include specializations of the type, allowing the specialized types in all contexts where the base type is allowed.

The declarations MAY occur in any order. By convention, they are usually ordered alphabetically or grouped logically.

Common content model patterns

Structural and element domain modules MAY include a "common content model patterns" section that defines patterns that contribute to the content models of the element types defined in the module. This section SHOULD use the section header "COMMON CONTENT MODEL PATTERNS":

```
<div>
  <a:documentation>COMMON CONTENT MODEL PATTERNS</a:documentation>
  ...
</div>
```

Common content model patterns SHOULD use names that end with ".cnt", e.g. "body.cnt".

Common attribute sets

Structural and element domain modules MAY include a "common attribute sets" section that defines patterns for attribute sets common to one or more of the element types defined in the module. This section SHOULD use the section header "COMMON ATTRIBUTE SETS":

```
<div>
  <a:documentation>COMMON ATTRIBUTE SETS</a:documentation>
  ...
</div>
```

Common attribute set patterns SHOULD use names that with "-atts", e.g. "topicref-atts".

Element type declarations

For each element type declared in the vocabulary module there MUST be a set of patterns that define the content model and attributes for the element type. Each set of patterns SHOULD be grouped within a <div> element and should start with a descriptive comment of the form:

```
<div>
  <a:documentation>LONG NAME: Long Name</a:documentation>
  ...
</div>
```

Each element type MUST have a content model pattern named `tagname.content`. The value of the pattern must be the complete content model definition. For example:

```
  <define name="topichead.content">
    <optional>
      <ref name="topicmeta"/>
    </optional>
    <zeroOrMore>
      <choice>
        <ref name="anchor"/>
        <ref name="data.elements.incl"/>
        <ref name="navref"/>
        <ref name="topicref"/>
      </choice>
    </zeroOrMore>
  </define>
```

The content model pattern MAY be overridden in constraint modules to further constrain the content model for the element type.

Each element type MUST have an attribute list pattern named `*tagname*.attributes`. The pattern entity must declare all attributes used by the element type (except for the attributes provided by the `global-atts` pattern, which is always referenced as part of the attribute list declaration for an element's class attribute). For example:

```
  <define name="topichead.attributes">
    <optional>
      <attribute name="navtitle"/>
    </optional>
    <optional>
      <attribute name="outputclass"/>
    </optional>
    <optional>
      <attribute name="keys"/>
    </optional>
    <optional>
      <attribute name="copy-to"/>
    </optional>
    <ref name="topicref-atts"/>
    <ref name="univ-atts"/>
  </define>
```

The attribute list declaration MAY be overridden in constraint modules to further constraint the attribute list for the element type.

Each element type MUST have a pattern named `*tagname*.element` containing an <element> element whose @name value is the element type name and whose content is a reference to the `*tagname*.attlist` and `*tagname*.content` patterns. The <element> element SHOULD include a <a:documentation> element containing a short description of the element type. The description should not include any unescaped XML markup. For grammars intended for use in generating DTD- and XSD-syntax modules, the <element> element SHOULD specify the attribute "longName" in the DITA architecture name ("http://dita.oasis-open.org/architecture/2005/", by convention bound to the prefix "ditaarch"). For example:

```
  <define name="topichead.element">
    <element name="topichead" ditarch:longName="Topic Head">
      <a:documentation>The &lt;topichead> element provides a title-only entry in a 
 navigation map,
        as an alternative to the fully-linked title provided by the &lt;topicref> 
 element.
        Category: Mapgroup elements</a:documentation>
      <ref name="topichead.attlist"/>
      <ref name="topichead.content"/>
    </element>
  </define>
```

Each element type must have a pattern named `*tagname*.attlist` with a @combine value of "interleave" containing only a reference to the `*tagname*.attributes` pattern. For example:

```
  <define name="topichead.attlist" combine="interleave">
    <ref name="topichead.attributes"/>
  </define>
```

example:

```
<div>
  <a:documentation>LONG NAME: Topic Head</a:documentation>
  <define name="topichead.content">
    <optional>
      <ref name="topicmeta"/>
    </optional>
    <zeroOrMore>
      <choice>
        <ref name="anchor"/>
        <ref name="data.elements.incl"/>
        <ref name="navref"/>
        <ref name="topicref"/>
      </choice>
    </zeroOrMore>
  </define>
  <define name="topichead.attributes">
    <optional>
      <attribute name="navtitle"/>
    </optional>
    <optional>
      <attribute name="outputclass"/>
    </optional>
    <optional>
      <attribute name="keys"/>
    </optional>
    <optional>
      <attribute name="copy-to"/>
    </optional>
    <ref name="topicref-atts"/>
    <ref name="univ-atts"/>
  </define>
  <define name="topichead.element">
    <element name="topichead">
      <a:documentation>The &lt;topichead> element provides a title-only entry in a
 navigation map,
        as an alternative to the fully-linked title provided by the &lt;topicref>
 element.
        Category: Mapgroup elements</a:documentation>
      <ref name="topichead.attlist"/>
      <ref name="topichead.content"/>
    </element>
  </define>
  <define name="topichead.attlist" combine="interleave">
    <ref name="topichead.attributes"/>
  </define>
</div>
```

Specialization attribute declarations

A vocabulary module MUST define a @class attribute for every specialized element declared in the module.

This section declares the @class attributes for the element types defined in the module. This section SHOULD use the section header "SPECIALIZATION ATTRIBUTE DECLARATIONS":

```
<div>
  <a:documentation>SPECIALIZATION ATTRIBUTE DECLARATIONS</a:documentation>
  ...
</div>
```

For each element type defined in the module there MUST be a pattern named `*tagname*.attlist` that contains a reference to the attribute list pattern "global-atts" and defines an optional attribute named "class". The @class attribute default value MUST include the value of the @class attribute of the base element, and append to it the element name qualified by the topic element name with at least one leading and trailing space. The @class attribute for an element introduced by a structural specialization MUST start with a minus sign ("-"). The @class attribute for a domain specialization MUST start with a plus sign ("+"). The initial minus or plus sign MUST be followed by one or more spaces. The attribute value MUST end with one or more trailing space.

The attribute default value is declared using the @defaultValue attribute from the RELAX NG DTD compatibility namespace "http://relaxng.org/ns/compatibility/annotations/1.0" (by convention bound to the prefix "a").

For example:

```
<define name="anchorref.attlist" combine="interleave">
  <ref name="global-atts"/>
  <optional>
    <attribute name="class"
      a:defaultValue="+ map/topicref mapgroup-d/anchorref "
    />
  </optional>
</define>
```

The @class attribute declarations for a module MUST be grouped together at the end of the module after any other declarations. The declarations may occur in any order. By convention they are often ordered alphabetically or grouped logically.

See *classatt.xml#classatt* for complete details on the @class attribute.

Structural module coding requirements

A structural vocabulary module defines a new topic or map type as a specialization of a base topic or map type. The purpose is usually to enhance the user's interaction by adapting the topic or map type to its particular purposes.

A structural type module MUST conform to the following coding requirements in addition to the general module coding requirements and the rules defined in *Using RELAX NG for DITA document type shells, vocabulary modules, and constraint modules* on page 19.

Architecture attributes

The topic or map element type MUST declare the @DITAArchVersion attribute and set its value to "1.3". This attribute gives processors a reliable way to check the architecture version. In addition, because the @DITAArchVersion attribute is in a DITA-defined namespace, it serves as an unambiguous signal that the element is a DITA element.

The architectural attributes section SHOULD use the section header "ARCHITECTURE ATTRIBUTES":

```
<div>
  <a:documentation>ARCHITECTURE ATTRIBUTES</a:documentation>
  ...
</div>
```

The architectural attributes pattern MUST have the name "arch-atts" and MUST declare the @DITAArchVersion attribute in the DITA architecture namespace (by convention bound to the prefix "ditaarch"):

```
  <define name="arch-atts">
    <optional>
      <attribute name="ditaarch:DITAArchVersion" a:defaultValue="1.2"/>
    </optional>
  </define>
```

Topic type module coding requirements

Topic type vocabulary modules MUST conform to additional coding requirements for defining default topic nesting.

Default nested topics pattern

A topic type module MUST define a pattern to specify default subordinate topics. The pattern name MUST be the topic element name plus the suffix `-info-types`. For example, the info-types pattern for the concept topic type is `concept-info-types`. The info type pattern section should use the section header "INFO TYPES PATTERNS":

```
<div>
  <a:documentation>INFO TYPES PATTERNS</a:documentation>
  ...
</div>
```

If the topic has default subordinate topics, this pattern MAY refer to a list of topic element type name patterns. If not, the pattern SHOULD refer to the `info-types` pattern as in the following example:

```
<div>
  <a:documentation>INFO TYPES PATTERNS</a:documentation>
  <define name="concept-info-types">
    <ref name="info-types"/>
  </define>
  ...
</div>
```

A document type shell can then control how topics are allowed to nest by redefining the *`topictype`*`-info-types` pattern for each topic type, or it can efficiently create common nesting rules by redefining the main info-types pattern.

In the declaration of the root element of a topic type, the last position in the content model MUST be the *`topictype`*`-info-types` nested topics pattern, as in the following example:

```
<div>
  <a:documentation>LONG NAME: Concept</a:documentation>
  <define name="concept.content">
    <ref name="title"/>
    <optional>
      <ref name="titlealts"/>
    </optional>
    <optional>
      <choice>
        <ref name="abstract"/>
        <ref name="shortdesc"/>
      </choice>
    </optional>
    <optional>
      <ref name="prolog"/>
    </optional>
    <optional>
      <ref name="conbody"/>
    </optional>
    <optional>
      <ref name="related-links"/>
    </optional>
    <zeroOrMore>
      <ref name="concept-info-types"/>
    </zeroOrMore>
  </define>
  ...
</div>
```

Element domain module coding requirements

An element domain vocabulary module defines element types that are appropriate for the subject-matter or application domain for which they are designed. The purpose is usually to enhance the user's interaction by providing “semantic” elements whose names more accurately denote their content, making that content easier to search and retrieve.

Domain extension patterns

The domain extension patterns serve to integrate domain modules into the base content models of the element types the module extends when the module is included in a document type shell. This section SHOULD use the section header "DOMAIN EXTENSION PATTERNS":

```
<div>
  <a:documentation>DOMAIN EXTENSION PATTERNS</a:documentation>
  ...
</div>
```

For each element type that is extended by the element domain module, the module MUST define a domain extension pattern consisting of a choice group of references to the element type name patterns of the domain-provided extensions of the extended element type. The name of the pattern MUST be constructed from the short name for the domain, "-d-", and the element type being extended. For example, this pattern extends the <ph> element to add the highlight-domain-defined specializations of <ph>:

```
  <define name="hi-d-ph">
    <choice>
      <ref name="b"/>
      <ref name="i"/>
      <ref name="sup"/>
      <ref name="sub"/>
      <ref name="tt"/>
      <ref name="u"/>
    </choice>
  </define>
```

For each element type that is extended by the element domain module, the module MUST redefine the element type's pattern with a @combine value of "choice" containing a reference to the domain extension pattern. For example, this pattern adds the highlight domain specializations of <ph> to the <ph> content model:

```
  <define name="ph" combine="choice">
    <ref name="hi-d-ph"/>
  </define>
```

Because the redefinition of the extended element type's pattern uses a @combine value of "choice", the effect is that the domain-provided elements are automatically added to the effective content model of the extended element in any grammar that includes the domain module.

Attribute domain module coding requirements

An attribute domain vocabulary module declares a new attribute specialized from either the @props or @base attribute. An attribute domain module defines exactly one new attribute type.

An attribute domain's name is the name of the attribute plus "Att" to distinguish the attribute domain from any element domains with the same name. For example, for an attribute named "new" the attribute domain name would be "newAtt". The attribute domain name is used to construct filenames and pattern names for the domain.

An attribute domain MUST consist of one file, whose name consists of the module name plus `DomainMod` plus the `rng` or `rnc` extension. For example: `newAttDomainMod.rng` for an attribute named "new".

The file MUST have three sections:

Domains attribute contribution

The @domains contribution for attribute domains uses the pattern "a(*baseAttributeName specializedAttributeNames*)" where *baseAttributeName* is either "base" or "props" and *specializedAttributeNames* is a space-separated list of attribute names reflecting the specialization hierarchy of the specialized attribute and ending with the name of the specialized attribute defined in the module.

example:

An attribute specialized directly from @props:

```
<moduleDesc>
  ...
  <moduleMetadata>
    ...
    <domainsContribution>a(props deliveryTarget)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

An attribute "myDeliveryTarget" specialized from @deliveryTarget:

```
<moduleDesc>
  ...
  <moduleMetadata>
    ...
    <domainsContribution>a(props deliveryTarget myDeliveryTarget)</
domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

Attribute extension pattern

The attribute extension pattern extends either the @props or @base attribute set pattern to include the attribute specialization.

For specializations of @props the pattern MUST be named "props-attribute-extensions" and MUST specify a @combine value of "interleave". The content of the pattern MUST be a reference to the specialized attribute declaration pattern. For example:

```
  <define name="props-attribute-extensions" combine="interleave">
    <ref name="deliveryTargetAtt-d-attribute"/>
  </define>
```

For specializations of @base the pattern MUST be named "base-attribute-extensions" and MUST specify a @combine value of "interleave" The content of the pattern MUST be a reference to the specialized attribute declaration pattern. For example:

```
  <define name="base-attribute-extensions" combine="interleave">
    <ref name="myBaseSpecializationAtt-d-attribute"/>
  </define>
```

Attribute declaration pattern

The specialized attribute MUST be declared in a pattern named `*domainShortName*-d-attribute`. The attribute must be defined as optional. For example:

```
  <define name="deliveryTargetAtt-d-attribute">
    <optional>
      <attribute name="deliveryTarget"/>
    </optional>
  </define>
```

Constraint module RELAX NG coding requirements

A structural constraint module defines the constraints for exactly one map or topic element type. A domain constraint module defines the constraints for exactly one element or attribute domain.

Requirements for structural constraint modules

Constraint modules SHOULD be named "*qualifiertagname*Constraints.mod", where *qualifier* is specific to the constraints module and characterizes it, e.g. "strict", "requiredTitle", etc. and *tagname* is the name of the element type to which the constraints apply, e.g. "topic", "p", "myNewTopicType", etc.

The constraint module's @domains contribution pattern MUST be of the form "(*tagname qualifierTagname*-c)", where *tagname* is the name of the element type to which the constraints apply, *qualifier* is as for the module filename (e.g., "strict"), and *Tagname* is the element type name with an initial capital (e.g. "Topic"). The literal "-c" indicates that the name is the name of a constraints domain. For example:

```
<moduleDesc>
  ...
  <moduleMetadata>
    ...
    <domainsContribution>(topic task strictTaskbody-c)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

Requirements for domain constraint modules

A domain constraint module defines the constraints for exactly one element or attribute domain module.

Domain constraint modules SHOULD be named "*qualifierdomain*DomainConstraintsMod" with an extension of ".rng" or ".rnc", where *qualifier* is specific to the constraints module and characterizes it, e.g. "strict", "requiredTitle", etc. and *domain* is the name of the domain to which the constraints apply, e.g. "hi-d", "pr-d", "mydomain-d", "deliveryTargetAtt-d", etc.

The constraint module's @domains contribution pattern MUST be of the form "(*domain qualifierDomain*-c)", where *domain* is the name of the domain to which the constraints apply, *qualifier* is as for the module filename (e.g., "strict"), and *Domain* is the domain name with an initial capital (e.g. "Taskreq"). The literal "-c" indicates that the name is the name of a constraints domain. For example:

```
<moduleDesc>
  ...
  <moduleMetadata>
    ...
    <domainsContribution>(topic task taskreq-d)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

Requirements for shell document types

Constraint modules are integrated into document type shells by simply including the module into the shell document type. See *RELAX NG document-type shell: Coding requirements* on page 23. The @domains contribution of the constraint module MUST be reflected in the @domains value for the root map or topic type configured by the document type shell.

Specializing foreign or unknown content

Example of specializing foreign content using RELAX NG

The sample below describes how to create a domain declaration for the RELAX NG grammar element and integrate it into a document type shell.

Domain modules that include foreign grammars should use the <externalRef> element to refer to the foreign grammar when the grammar specifies a <start> element. When using the <externalRef> element, it is possible to safely include foreign vocabularies that are not in a namespace because the externally-referenced grammar's patterns are not merged with those of the referencing grammar.

The following specialization domain specializes <foreign> in order to contain RELAX NG <grammar> elements, for example, to document document types by rendering the grammar markup as diagrams.

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
  xmlns="http://relaxng.org/ns/structure/1.0">
<moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
   <moduleTitle>RELAX NG Foreign Domain</moduleTitle>
   <headerComment>
=============================================================
Define the RELAX NG XML syntax as a DITA foreign vocabulary
=============================================================
</headerComment>
   <moduleMetadata>
     <moduleType>elementdomain</moduleType>
     <moduleShortName>rnggrammar-d</moduleShortName>
     <modulePublicIds>
       <dtdMod>urn:oasis.org:elements:dita:dtd:rnggrammarDomain.mod</dtdMod>
       <dtdEnt>urn:oasis.org:entities:dita:dtd:rnggrammarDomain.ent</dtdEnt>
       <xsdMod>urn:oasis.org:names:dita:xsd:rnggrammarDomain.xsd</xsdMod>
       <rncMod>urn:oasis.org:names:dita:rnc:rnggrammarDomain.rnc</rncMod>
       <rngMod>urn:oasis.org:names:dita:rng:rnggrammarDomain.rng</rngMod>
     </modulePublicIds>
     <domainsContribution>(topic rnggrammar-d)</domainsContribution>
   </moduleMetadata>
 </moduleDesc>

 <div>
   <a:documentation>DOMAIN EXTENSION PATTERNS</a:documentation>

  <define name="rnggrammar-d-foreign">
    <choice>
      <ref name="rng_grammar"/>
    </choice>
  </define>

  <define name="foreign" combine="choice">
    <ref name="rnggrammar-d-foreign"/>
  </define>
 </div>
 <div>
   <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
  <define name="rng_grammar">
    <ref name="rng_grammar.element"/>
  </define>
 </div>
 <div>
   <a:documentation>ELEMENT TYPE DECLARATIONS</a:documentation>
    <div><a:documentation>LONG NAME: RNG Grammar Container</a:documentation>
      <define name="rng_grammar.content">
        <zeroOrMore>
          <choice>
            <externalRef href="rngGrammar.rng"/>
            <ref name="data"/>
            <ref name="data-about"/>
          </choice>
        </zeroOrMore>
      </define>
      <define
        name="rng_grammar.attributes">
        <ref name="univ-atts"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
      </define>
      <define name="rng_grammar.element">
        <element name="rng_grammar" ditaarch:longName="Bold">
          <a:documentation></a:documentation>
          <ref name="rng_grammar.attlist"/>
          <ref name="rng_grammar.content"/>
        </element>
      </define>
      <define name="rng_grammar.attlist"
        combine="interleave">
        <ref name="rng_grammar.attributes"/>
      </define>
    </div>
 </div>
 <div>
   <a:documentation>SPECIALIZATION ATTRIBUTE DECLARATIONS</a:documentation>
  <define name="rng_grammar.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="+ topic/foreign rnggrammar-d/
rng_grammar "/>
    </optional>
  </define>
 </div>
</grammar>
```