Elkera Pty Limited

Elkera BNML Schema

Guide to BNML Schema configuration

Version 1.00, 27 July 2005

Elkera Pty Limited ABN 68 092 447 428
Suite 701, 10 Help Street, Chatswood, NSW 2067, Australia
PO Box 5280, West Chatswood, NSW 1515
Telephone +61 2 8440 6999
Facsimile +61 2 8440 6988
www.elkera.com.au

Contents

1 Introduction 1

1.1 Purpose of this document 1

1.2 Background to the BNML Schema 1

1.3 What is the BNML Schema? 2

2 Description of the BNML files 2

2.1 Overview 2

2.2 bnml-standard.rnc 3

2.3 bnml-core.rnc 3

2.4 bnml-structure.rnc 3

2.5 bnml-document.rnc 3

2.6 bnml-contract.rnc 4

2.7 bnml-correspondence.rnc 4

2.8 dc-metadata.rnc 4

2.9 xi-include.rnc 4

3 Customization of BNML Standard 4

3.1 Overview 4

3.2 Classes of customization 4

3.3 High level approach to customization 5

3.4 Adding or removing document types 5

3.5 Changing hierarchy structure models 6

3.6 Adding new elements 6

3.7 Adding or redefining attributes 7

Guide to BNML Schema configuration

1 Introduction

1.1 Purpose of this document

This document is an introductory guide to configuration and customization of the Elkera® Business Narrative Markup Language (BNML™) Schema, version 1.05. A more complete specification for the BNML Schema is under development.

1.2 Background to the BNML Schema

The BNML Schema is developed and maintained by Elkera (http://www.elkera.com). The principal developer of the BNML Schema is Elkera's senior consultant and XML applications architect, Andrew Squire. Other Elkera consultants have contributed to its development over many years. Elkera proposes to release the BNML Schema under an open source licence once it is satisfied of the best approach for so doing.

The basic objectives, features and expected benefits of the Elkera BNML Schema are described in Elkera's Introduction to the Elkera BNML Schema (http://www.elkera.com/cms/index.php?id=8). A brief description of each element and attribute in the BNML Schema is provided in the schema source files.

BNML Standard is defined in RELAX NG Compact syntax. A good tutorial for this syntax can be found at: http://www.relaxng.org/compact-tutorial-20030326.html#id2814005.

Currently, TRANG is used to produce XML Schema (.xsd) versions of BNML Standard. This produces a schema layout that may not be ideally suited to customizing the XSD format. Elkera proposes to prepare a native xsd format that better supports customization.

Elkera has not yet produced a DTD version of BNML Standard. The item element is re-defined when contained by the block element. This is not supported by DTD syntax, thus preventing automatic creation of a DTD version. Elkera proposes to provide a DTD version of BNML Standard in the future.

1.3 What is the BNML Schema?

The BNML Schema has a different architecture to most other general purpose narrative document schema. One of its principal objectives is to avoid the large number of elements and loose content models of those schema. The BNML Schema is a foundation for the construction of application specific schema. It requires a modest amount of development to meet individual application requirements before it can be used.

The BNML Schema, can be thought of as a family of schema, sharing a common core that enables them to take advantage of supporting applications with minimal effort and expense. It is not intended that all schema members of the BNML Schema family are fully compatible so that arbitrary document interchange can occur. Such compatibility can be achieved only by the methods already explored by many other schema. As explained in Elkera's Introduction to the Elkera BNML Schema (http://www.elkera.com/index.php?id=8), that approach has many limitations.

The foundation of the BNML Schema is the file bnml-core.rnc. That file defines the basic, low level components common to most narrative document types. BNML Core is the essence of the BNML Schema family.

The BNML-Standard defines a base schema with three document types, document, contract and correspondence. However, principally because it deliberately lacks defined values for most element attributes, BNML Standard is not proposed as a working schema. Some working schema will be BNML subsets, as explained in section 3.2, and will be valid against BNML Standard. To that extent, BNML Standard can function as an interchange schema.

Elkera has created XML-2-Go as a working application of BNML Standard. This can be downloaded by approved users from Elkera's web site. (http://www.elkera.com).

Approaches to modification and extension of the BNML Schema are described in section 3.2, thereby defining whether particular schema may retain the root designation “BNML”.

2 Description of the BNML files

2.1 Overview

BNML Standard consists of 6 BNML specific files and 2 files that incorporate external features into the application:

- bnml-standard.rnc
- bnml-core.rnc

- bnml-structure.rnc
- bnml-document.rnc
- bnml-contract.rnc
- bnml-correspondence.rnc
- dc-metadata.rnc
- xi-include.rnc

The purpose of each of these files is described in the following sections.

2.2 bnml-standard.rnc

This is the *umbrella* file that defines the BNML Standard schema. This file includes the default namespace definition for BNML Standard: http://www.elkera.com/ns/2003/bnml-standard.

That file contains statements to include all other files that make up the schema.

The bnml-standard.rnc file can be copied and re-named to create a new application of the BNML Schema. Customization of BNML standard is described in section 3.

2.3 bnml-core.rnc

This file contains the core patterns and elements that define the BNML Schema family which are used as the base of BNML Standard. These core patterns can be used to create the basic structures that occur in many narrative documents. These patterns are used in all BNML document types.

2.4 bnml-structure.rnc

This file contains a number of shared elements that are not considered part of BNML core. These elements are based on the patterns found in BNML core and are used to create specific types of structures that can be found in any of the 3 basic BNML Standard document types.

2.5 bnml-document.rnc

This file defines a BNML document type of document, as well as a other elements that are specific to this document type.

2.6 bnml-contract.rnc

This file defines a BNML document type of contract, as well as a other elements that are specific to this document type.

2.7 bnml-correspondence.rnc

This file defines a BNML document type of contract, as well as a other elements that are specific to this document type.

2.8 dc-metadata.rnc

This file provides elements from the Dublin Core metadata set. This is used to incorporate a number of basic metadata elements into BNML Standard.

2.9 xi-include.rnc

This file provides the xi:include element, which is defined by the World Wide Web Consortium XML Inclusions recommendation. This is used in BNML Standard to provide for content re-use applications.

A full description of this element can be found at: http://www.w3.org/TR/2004/PR-xinclude-20040930.

3 Customization of BNML Standard

3.1 Overview

BNML Standard is not intended as an end user schema. It is intended to be the foundation upon which organization or application specific schema are built. This section describes the different classes of customization, the recommended approach to customization, and a general introduction to how the customize the schema.

3.2 Classes of customization

There are 2 main classes of customization of BNML Standard:

BNML Subset — This is a customization of BNML Standard that can be validated against BNML standard. This class of customization can only add attribute value enumerations, redefine existing attributes, perhaps to provide default values or to change the data type of attributes. Changed data types must remain valid when validating against BNML Standard. It is also possible to remove BNML document types.

BNML Variant — This is a customization of BNML that cannot be validated against BNML Standard. There are no restrictions on this type of customization, however the integrity of the core patterns found in bnml-core.rnc must be retained if the schema is to retain designation as part of the BNML Schema family. This class of customization may add new attributes to existing elements, add new elements and new document types.

The type of customization best suited to a particular application is left to the application developer. It is envisaged that a variant customization would be a typical approach. However, a subset customization may be useful for simple cases or where document interchange is a requirement.

3.3 High level approach to customization

BNML Standard is structured to allow easy customization. It is intended that all customization of BNML Standard will take place in an application customization layer. This should be a single file that acts as an *umbrella* file and includes all other files that are part of BNML Standard, as well as the extensions to BNML that make up the customization. The existing BNML Standard files must not be changed unless they are renamed.

The file bnml-standard.rnc can be used as a template for the customization layer file. A typical customization would begin by making a copy of bnml-standard.rnc and renaming this file using an appropriate file name.

For a BNML Subset it is recommended that the name be in the form:

`bnml-s-<application name>.rnc`

A BNML Subset must validate against BNML Standard.

For a BNML Variant it is recommended that the name be in the form:

`bnml-<application name>.rnc`

In each case, `<application name>` is a name that identifies the customization. The application name may also include the organization name if desired.

The advantage of the customization layer approach is that it allows the application to be easily updated when new versions of BNML Standard are released by Elkera because the base BNML Standard files are unchanged.

3.4 Adding or removing document types

BNML Standard provides three main document types:

- contract

- correspondence
- document.

Some applications may require different document types and some applications may not need all of these document types.

A new document type can be added by creating a new file to contain the main element and any other type specific elements that are required. This is then included into the customization layer using the "include" statement. See bnml-document.rnc and bnml-standard.rnc for an example of this approach.

An existing BNML document type may be removed from an application simply by removing the "include" statement in the customization layer that adds the document type to the application.

3.5 Changing hierarchy structure models

BNML Standard is designed to allow the content models of the major structural container elements to change the basic pattern used create the document hierarchy.

BNML Standard provides threes basic patterns that are defined in bnml-core.rnc:

- `tight.structure.model` – This permits either block or item elements but does not allow both at the same level.
- `standard.structure.model` – This permits block elements before the first item but not otherwise at the same level with item.
- `loose.structure.model` – This permits item and block elements to be mixed in any order at the same level.

Most structural elements have a pattern that can be set to one of these structure models. This pattern is always in the form:

<element name>.structure.model.

All elements are defined with a default that best suits their intended use. The content model can be tightened or loosened according to application requirements in the customization layer.

The hierarchy structure models may also be used when creating new elements.

3.6 Adding new elements

BNML Standard is designed to allow application developers to add elements to the content models for block level elements and inline elements only. If

elements are to be added to other content models, the element may need to be redefined in the customization layer.

There are two main patterns in which new elements may be added:

- `block.level.elements` – Adding new elements to this pattern makes the element available within the block element.
- `inline.content.inner` – Adding elements to this pattern adds inline elements to most contexts where text (#PCDATA) is allowed.

These patterns are defined in bnml-core.rnc. New elements must be combined into the existing pattern using the RNC "|=" operator. This should be done inside the bnml-core.rnc include statement in the customization layer.

3.7 Adding or redefining attributes

3.7.1 Overview

BNML Standard is designed to allow application developers easily add new attributes to existing elements. Usually, this will be the first type of customization an application developer will need to make.

All element attributes in BNML are defined in a similar way. This section describes the common features of each element attribute definition and provides some background so this information does not have to be repeated for each element attribute definition.

3.7.2 Attribute list definitions

All attributes for an element are defined in a single RELAX NG pattern:

`<elements-name>.attlist`

This method allows all attributes of an element to be removed or replaced by a customization.

3.7.3 Common attribute patterns

All attribute list definitions contain two patterns that can be used by a developer to customize attribute lists:

- `common.attributes` – This pattern is used for attributes that should occur on all elements. By default, id is the only common attribute.
- `<element-name>.attlist.extensions` – This pattern is used to add new attributes to the element for a particular customization.

3.7.4 Class attributes

Many elements also have a class attribute. This will be defined in the attribute pattern:

`<element-name>.class.attribute`

The class attribute is used to define additional semantics for the element. These semantics can be used to control processing applications and rendering applications.

By default all class attributes have a standard definition of xsd:string. It is intended that a customization may provide a list of enumerated values for this attribute.

3.7.5 Numbering attributes

Some elements have attributes that are intended to control automatic numbering processors. These will be defined in the attribute pattern:

`<element-name>.numbering.attributes`

There may be many different attributes defined in this pattern.

Separating attributes into these different patterns provides a way for an application developer to remove attributes from an element if they are not required for a customization or do redefine these attributes if different numbering attributes are required.