Akoma Ntoso 1.0

Release Notes

Strengthening Parliaments' Information Systems in Africa

UNITED NATIONS Department of Economic and Social Affairs

Release 02/05/2006

Fabio Vitali - University of Bologna - Italy

Table of content

1 Introduction

This document introduces and explains the schema for AKOMA NTOSO 1.0, an XML-based document format for legislative documents in African countries. More details about the background and rationale of this project and of the design methodology can be found in the AKOMA NTOSO framework document [1].

In this document technical details and assumptions about the AKOMA NTOSO 1.0 document structure and elements are provided, as well as some hints for document markup using this schema. The final document describes two different but connected families of schema. The first is the AKOMA NTOSO General Schema, a vocabulary and a minimal set of constraints that all AKOMA NTOSO documents must comply to. A set of stricter schemas, the AKOMA NTOSO Detailed Schemas, provides more constraints over the same vocabulary of elements to enforce the rules of specific document types in specific African countries. It is a requirement of AKOMA NTOSO that all documents satisfying one of the Detailed Schemas also satisfy the General Schema.

In this release only the General Schema is described in full. Thus, except when explicitly mentioned, all rules are expected to refer to the General Schema (and thus to **all** AKOMA NTOSO documents).

2 Namespaces

AKOMA NTOSO 1.0 documents are completely qualified, i.e., namespaces are used throughout. Even though some elements use the same name as HTML elements, and in fact are directly drawn out of the HTML vocabulary, out of simplicity it has been decided to use one namespace only, so that all elements are similarly qualified. The net result is that it is possible to specify the AKOMA NTOSO namespace as the default namespace and have no prefixes in the instance document, while maintaining full qualification of the documents.

The namespace for this release of AKOMA NTOSO is “`http://www.akomantoso.org/1.0`”.

3 Global overview of the schema

All AKOMA NTOSO documents share the same root element `<akomantoso>`, under which the specific document type is selected. The single root element follows a specific design pattern “Universal root” aimed at better identification of the root and separation of namespace and schema declaration (available in the root) and meaningful attributes (available in the document type element).

The schema starts with a few `<group>`s and `<attributeGroup>`s used throughout the schema for content models and types. They are followed by common simple types (mostly enumerations of string values) and complex types. Complex types in this sections include those supporting four of the five main content model patterns used throughout this schema: `hierarchy` (a hierarchy of nested elements with number and titles), `blocks` (a sequence of block elements - e.g., paragraphs) used within containers either with required or optional identifiers, `inline` (the content model for all mixed model elements such as paragraphs), and `marker` (zero length elements characterized by their attributes) either with required or optional identifiers. The fifth content model pattern, `container`, has no common form, but lists different elements in different orders, and individual container-like complex patterns are spread throughout the schema. Content model patterns are described in section 4.2.

Elements are organized in meaningful sections:

- The root element `<akomantoso>`
- The document elements, one for each document type (`<act>`, `<bill>`, `<doc>`, `<report>` and `<minutes>`), that share one of the three document formats: `&HierarchicalStructure;` (that has an explicit hierarchy inside), `&OpenStructure;`, that allows basically everything inside, and `&DebateStructure;`, a slightly hierarchical structure for minutes and reports.
- The container elements, one for each main part of the abovementioned structures, except for clauses, described next, and meta, described in the apposite section.
- The hierarchical elements, listing the main elements that are used in the full hierarchy of nested structures of acts and bills, as well as `<title>`s, `<num>`s and `<subtitle>`s.
- Elements for parliamentary debates, particularly `<subdivision>`, `<speech>`, `<question>` and elements for open structures, particularly `<item>`.
- AKOMA NTOSO specific block and inline elements, including the table of content (`<TOC>`), the normative reference (`<ref>`), the defined term in a definition (`<def>`) the note marker (`<noteref>`) pointing to an editorial note placed out of line (in the meta section), the recorded time of aa spoken remark (`<recordedTime>`), the container for amendments (`<mod>`) and of two types of amendment quoted fragments: simple text fragments (such as a few words inside quotes) or full structures (such as an entire clause or article).
- Generic elements: the list of available generic elements (one for each of the five main patterns for content models), explained in detail in section 6.
- HTML elements: the list of elements, directly derived from HTML, used to provide for presentation-oriented, rather than semantic-oriented, markup within AKOMA NTOSO documents. They form a very strict simplification of the HTML language, but allow for many useful structures inside an act. HTML elements and how to use them in AKOMA NTOSO are described in section 7.
- Metadata elements provide a location for all relevant information about a AKOMA NTOSO document that *does not belong to its actual content*. Metadata thus are all, by definition, editorial additions to the text as originally composing the document. Metadata are described in section 8.

In this release the AKOMA NTOSO 1.0 schema contains a full total of 83 elements, of which 41 specific to the AKOMA NTOSO vocabulary, 5 generic elements, 14 HTML elements, and 23 metadata elements.

4 Patterns

Patterns are the abstraction and distillation of past experiences in designing and resolving design problems. They are general and widely applicable guidelines for approaching and justifying design issues that often occur in XML-based projects.

We distinguish between patterns in content models (a restriction of content models to the ones that are actually useful) and patterns in schema design (guidelines on how to make a schema more modular, flexible and understandable by users). Both patterns are well known and well established in the literature, although by different experts in different ways. For patterns in content models we used our own resource, [2], while the authoritative resource for patterns in schema design is [3].

4.1 Patterns in content model

The AKOMA NTOSO 1.0 Schema uses systematically five of the 7 patterns described in [2]. This means that all content models and complex types used in the schema follow precisely the form of the relevant pattern, and all elements can be simply described and treated according to their pattern rather than individually.

These patterns are:

- The *markers*: markers are content-less elements that are scattered here and there in the document and are meaningful for their names as well as their attributes. Markers are also known in literature as *empty elements* or *milestones*. There are two main families of markers in the AKOMA NTOSO schema: placeholders in the text content (e.g., note references) that can appear in any position that also has text, and metadata elements that only appear in some subsection of the `<meta>` section. As discussed in section 8, all metadata elements are markers so that metadata values are not part of the text content of a document, but rather are attribute values.
- The *inlines*: an inline element is an element placed within a mixed model element that identifies some text fragment as relevant for some reason. There are both semantically relevant inlines and presentation oriented inlines. There is but one content model using inlines (and markers), which means that all mixed model elements (i.e., those that allow both text and elements) also allow the a repeatable selection of all inline elements. For a discussion of why this is only a trade-off decision, and not the ideal solution, see at the end of this section.
- The *blocks*: a block is a container of text or structures that is organized vertically on the display (i.e., has paragraph breaks) and can contain either substructures or text. Most blocks in AKOMA NTOSO are based on the HTML language. There is only one content models using blocks, that allow a repeatable selection of all available blocks. This means that wherever a block is allowed (e.g., a paragraph), a table or a list is also allowed.
- The *containers*: containers are sequences of specific elements, some of which can be optional. The corresponding pattern in [2] is the *record*. Containers are all different from each other (as the actual list of contained elements vary), and so there is no single container content model, but rather a number of content models that share the record pattern.
- The *hierarchy*: a hierarchy is a set of arbitrarily deep nested sections with title and numbering. Each level of the nesting can contain either more nested sections or blocks. The corresponding pattern in [2] is the table. No text is allowed directly inside the hierarchy, but only within the appropriate block element (or, of course, titles and numbering). AKOMA NTOSO 1.0 uses only one hierarchy, with predefined names and no constraints on their order or systematic layering.

There are three exceptions to the systematic use of patterns:

- The `<li>` element allows both inlines and other nested lists (`<ul>` and `<ol>`). The pattern would require `<li>` elements to contain only text, and nested lists to be direct child of the main list (`<ul>`s within `<ul>`). Since this goes against universal HTML practice, we have decided against full pattern adherence and in favor of HTML tradition.
- The `<mod>` element allows quoted text and structures within its content. No problems for quoted texts, but when an amendment clause specifies in full a new structure (such as an

article) within the main discourse, the full structure needs to be described, and it is thus possible to have an article within a paragraph within a clause, which is against the inline pattern. There is no simple way out of this issue.

- There is six inline elements that only make sense in the preface and preamble parts of the document: these are <ActTitle>, <ActNumber>, <ActType>, <ActProponent>, <ActDate> and <ActPurpose>. They are in fact part of the one inline content model and thus are available everywhere in the document. There is no simple way to define blocks within <preamble> and <preface> to allow these elements and paragraphs elsewhere to not allow them, so it has been decided that it was better to allow them everywhere rather than uselessly complicating the schema. In [2] a direct solution to this issues is proposed (*additive context*, also known as *inclusions*), but in the current XML technology such constraint would require a different or additional validation language such as Schematron or SchemaPath, which constitute a possible evolution of the AKOMA NTOSO project, but certainly not an immediate one.

4.2 Patterns in schema design

Design patterns are distillation of common wisdom in organizing the parts and the constraints of a schema. They are listed in [3]. Whenever there has been a design choice to be made that was not immediately obvious and naturally acceptable, a relevant pattern has been sought and properly used. In fact, [3] also contain immediately obvious and naturally acceptable pattern that have been used in AKOMA NTOSO, but only the not-so-obvious and not-so-natural ones have been explicitly mentioned and referred to. You can find the relevant mentions within the schema itself, in comments and documentation.

5 Generic Elements

AKOMA NTOSO 1.0 strongly supports the idea of using semantically rich terms whenever a semantically justifiable text fragment exist in the document. This means that it is possible that users of AKOMA NTOSO in daily work will find the need for more elements than currently provided.

Generic elements come to aid in this respect. Whenever a new semantic is needed to describe a text fragment, a generic element of the appropriate content model is used instead, and the correct label is specified in the name attribute.

It is strongly discouraged to use presentation-oriented elements (such as b, i, etc.) elements to emphasize fragments that do have a semantic justification for being emphasized. Also, each text fragment need to be enclosed within the appropriate generic element according to its position and content model, which is the reason for there being five generic elements (one for each content model pattern).

Finally, an explicit equivalence is provided between named elements and generic elements: all named elements are just generic elements in disguise, the value of the name attribute having been upgraded to being the full element name. Therefore, for instance, <section> is absolutely equivalent to <hcontainer name=”section”>, or <noteref> is equivalent to <marker name=”noteref”>.

This is turn means that it is possible to reverse the approach, and, after a revision process, officially enrich the AKOMA NTOSO language with new elements that have been used in the past as values for the name attribute of generic elements.

6 HTML elements and CSS rules

AKOMA NTOSO uses a number of HTML elements for text fragments whose purpose is mainly presentation-oriented. These include paragraphs, lists, images, tables, and so on. Furthermore, as mentioned, even HTML elements have been made into the AKOMA NTOSO namespace, so as to simplify the namespace management.

Only a strict subset of the HTML language has been chosen, and no additional element can be added. In particular, headings (`<H1>, <H2>` and so forth) **cannot** be used in AKOMA NTOSO document, since they enforce a flat organization of sections, which is against the fundamentally hierarchical nature of AKOMA NTOSO documents. This is compatible with future developments of the HTML language, in particular considering that XHTML 2.0 will include nested hierarchies with `<section>` and `<h>` elements closely resembling AKOMA NTOSO `<hcontainer>` and `<title>` respectively.

All HTML elements have exactly the same nature and role as they have in HTML documents, with one exception: `<div>` is a generic container rather than a generic block as in HTML. This is due to the fact that a generic block already exist (`<p>`), and that in many automatically produced HTML documents (e.g., Open Office and MS Word), the `<div>` element is in fact used as a section separator (i.e., a container) rather than a paragraph.

The `<div>`, `<p>` and `<span>` elements can be considered as additional generic elements for the container, block and inline content models, and are in fact to be considered absolutely equivalent to `<container>`, `<block>` and `<inline>` elements, using the class attribute instead of the name attribute.

All HTML elements (and, in fact, all AKOMA NTOSO elements as well) can be optionally enriched with standard HTML core attributes allowing CSS styles with precise presentation instructions to be associated to them. The class and style attributes can be used as in HTML for external or internal CSS rules, liberally and without limitations on both HTML and AKOMA NTOSO elements.

7 Metadata elements

The meta section contains all the meta-information that needs or can be added to the actual content of the document. As a rule, all editorial content (i.e. content added by the editorial process out of Parliament rooms) need to be placed in the meta section, except for markup and note references. Vice versa, all actual content of the document need to have a place outside of the meta section in the appropriate content sections.

Meta elements are divided in four subsections:

- *Descriptors*: i.e., a set of metainformation providing info about the document and its publication and edition details, including its official promulgation date, its official URI, and so on.

- *Lifecycle*: the lifecycle element provides information about the events that the document has undergone, and references to the documents that have caused these events. Lifecycle is explained in section 10.
- *Notes*: this subsection contains the text of the editorial notes that might be produced to comment and expand the actual text of the document. Note references inside the text point to notes contained here.
- *Proprietary*: this subsection allows any additional metadata to be specified in any order and vocabulary (provided it uses a different namespace than AKOMA NTOSO). Proprietary metadata can be used within a specific document management system to specify additional information useful for internal search and document management that is not worth standardizing and imposing across all AKOMA NTOSO implementations.

The development of the meta section is not finished yet. For instance, support for Dublin Core metadata is currently imperfect (there are semantic equivalences between Dublin Core elements and AKOMA NTOSO elements, but they are not complete nor officially described as equivalent).

8 Identifiers

Identifiers are systematically used in AKOMA NTOSO. All AKOMA NTOSO elements allow an identifier. Many relevant elements and sections *require* it. Identifiers are the main way to identify fragments and parts of the document in an unambiguous form. They can be used in document references (e.g. links and amendment commands) as a precise pointer to the actual part of the document mentioned (as opposed to simply referring to a document as a whole). Also internal links need to use identifiers.

The schema does not explicitly provide a syntax for identifiers, which is described here in human readable format. Identifiers are composed by juxtaposing subidentifiers of the path needed to access them. Legal documents provide explicit global numbering for sections and articles, and local numbering for hierarchical subparts of them. For instance, all parts in different sections are numbered starting each time from 1, so “part 1” is not sufficient to clearly identify the actual part, while “article 12” clearly points to a single and well-specified element.

Each subidentifier is composed of the three letters, plus a number identifying its position within the overall list of similarly named elements. Blocks are all named “blkXX”, regardless of their actual names, and inlines are all named “inlXX”, regardless of their actual names. An exception to this are references “refXX” and amendments “modXX”.

The following is a table with some examples of identifiers:

Element	Identifier	Example	Identifier of example
`<section>`	`secXX`	Section 2 of this act	`sec02`
`<part>`	`prtXX`	Part 1 of section 2 of this act	`sec02-prt01`
`<paragraph>`	`parXX`	Paragraph 3 of part 1 of section 2 of this act	`sec02-prt01-par03`

`<chapter>`	`chpXX`	Chapter 5 of paragraph 3 of part 1 of section 2 of this act	`sec02-prt01-par03-chp05`
`<article>`	`artXX`	Article 12	`art12`
`<clause>`	`claXX`	Clause 3 of article 12	`art12-cla03`
`<li>`	`itmXX`	item “c” of clause 3 of article 12	`art12-cla03-itm03`
`<p>`	`blkXX`	Third paragraph of clause 3 of article 12	`art12-cla03-blk03`

Identifiers **never change** even if and when the elements get officially renumbered. Insertions may add a character at the end of the identifier. So if an amendment creates an article “12/a” or “12 bis” between articles 12 and 13, then the relevant identifies will be “art12a” in both cases.

Structures within the `<quotedStructure>` elements add the relevant mod identifier before their “natural” identifiers. So for instance if clause 3 of article 15 has an amendment that adds article 4/a to a different act, the identifier of the `<quotedStructure>` element that contains the new article will be “art15-cla03-mod01-art04a”. Of course, automatic systems that create current versions of texts will remove the prefix belonging to the amendment law and will only keep the “art04a” identifier in the final result.

9 Amendments, versions and document lifecycle

AKOMA NTOSO 1.0 include a sophisticated mechanism to keep track of the life cycle and evolution of a legislative document. This is particularly useful for acts that are amended and modified in time, while maintaining their fundamental nature.

The management of evolution of a document makes two very important assumptions:

- Amendments and events in the life cycle of a document (including original approval, final repeal and any other event affecting its presence in the law system or its content) happen in precise moments in time that can be determined objectively (albeit with difficulty) and attributed a specific date.
- Amendments and events in the life cycle are due to the enactment of a specific, individual document that can be objectively traced back and identified with an URI. If two different documents affect the same act on the same date, then these must be counted as two different and separate event on the amended act.

Handling events in AKOMA NTOSO centers around the `<lifecycle>` element in the meta section. This contains two containers, `<events>` and `<references>`, used to list the dates of all the events affecting a document, and the references to the URIs of all the documents generating these events. Each reference is provided with a required identifier, which is used by the event list to specify which document is responsible for which events. These elements must appear in all documents that have undergone two or more events (i.e., all acts except the ones that still have no amendments).

Documents in AKOMA NTOSO are organized in three main categories, as specified in the contain attribute of the document type element:

- OriginalVersion: this value reflects the fact that the content on the document is exactly the content that has been formally and explicitly approved by the relevant authority, with no amendments applied.

- SingleVersion: this value reflects the fact that the content of the document is an editorially modified version of the original act, according to one or more subsequent amendment acts. These amendment acts and the enactment dates of the amendments **must** be all present in the <lifecycle> element. Individual additions and deletions are not necessarily marked in the content.
- MultipleVersions: this value reflects the fact that the content of the document is the juxtaposition of fragments belonging to two or more different versions of the same act, each fragment marked as belonging to one or many of these versions. Thus in a MultipleVersions act there could be two or more copies of article 2, each associated to the date it started enactment and ended enactment.

The `<lifecycle>` element is a required element for all SingleVersion and MultipleVersion documents, and must be complete up to the enactment date of the latest document referenced in the `<lifecycle>` element (i.e., there can potentially be subsequent amendments non included in a SingleVersion or MultiVersion document, but all intermediate amendments must be correctly listed and referenced, even if they play no part to the displayed content). OriginalVersion documents need not have the `<lifecycle>` element, but surely can have it if the editors decide so.

In case a MultipleVersions document is being generated, each element and text fragment may be associated an enactment specification through the means of the three enactment attributes: start, end and status. Each fragment (a whole element if appropriate, otherwise a newly inserted `<span>` or `<inline>` element if no exact containing element exists) use these attribute to specify their nature.

The start and end attributes contain an IDREF to the ID of the event that has marked the beginning or the end of the enactment of the fragment. A start attribute with no end attribute marks a fragment that has appeared in an amendment and still exist in the latest recorded version of the document. An end attribute with no start attribute mark a fragment that was part of the original document but has been repealed before or at the latest recorded version of the document. The status attribute records the type of amendment of the fragment. The value “omissis” can only be used by private editors that want to display only part of the whole document. In this case, the structure must be complete anyway, but the actual content can be removed is the status=”omissis” attribute is present.

10 Document URIs

A resource is identified by a unique name. The name consists of the following pieces:

- A three letter country code, followed by,
- A detail area (contents differ across document types), followed by,
- Issuing Authorities (optional), followed by,
- Language coding information, followed by,
- Unique number .

URI	Description
`sle_act_2_2004_en`	Sierra Leone enacted Legislation. Act number 2 of 2004. English version.

URI	Description
`nam_bill_19_2003_1_en`	Namibia Bill number 19 of 2003, first stage, English version
`mdg_act_2002_003_fr_mlg`	Madagascar. Act 3 from 2003 in French and Malagasy.
`zaf_bill_12_2005_2_MinistryOfAgriculture_en`	South Africa. Bill 12 of 2005 (second stage), issued by the Ministry Of Agriculture. English version.
`dza_pdr_21122004_en`	Algerian parliamentary debate record, 21st December 2004. English version.

11 Element Synopsis

In the following table, a complete list of elements defined in this release of AKOMA NTOSO and their expected usage is provided.

Element name	Description
`AKOMA NTOSO`	*Content Model:* DocumentTypes (a list of either act, bill, doc, report, or minutes) *Description:* This is the root element, which contains all other elements. It is always present and required. *Use in:* all document types
`act`	*Content Model:* HierarchicalStructure *Description:* This is the element to use within `AKOMA NTOSO` for describing an act *Use in:* acts only
`bill`	*Content Model:* HierarchicalStructure *Description:* This is the element to use within `AKOMA NTOSO` for describing a bill *Use in:* bills only
`doc`	*Content Model:* OpenStructure *Description:* This is the element to use within `AKOMA NTOSO` for describing all kinds of documents for which there is no specific document type. *Use in:* all document types except acts, bills, reports, and minutes.
`report`	*Content Model:* DebateStructure *Description:* This is the element to use within `AKOMA NTOSO` for describing a parliamentary report *Use in:* reports only
`minutes`	*Content Model:* DebateStructure *Description:* This is the element to use within `AKOMA NTOSO` for describing the minutes of a parliamentary session (also known as Hansards) *Use in:* minutes only
`preface`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* The preface contains all information that happen before the preamble, and constitute the heading of the document presented to the public. This may include the title, the number, the date, the emanating body, and every other piece of information that precedes the preamble. The content is organized in blocks within which semantically loaded terms can be specified (in particular, ActType, ActTitle, ActNumber, ActProponent, ActDate and ActPurpose). *Use in:* all document types *Example:* `<preface>`

Element name	Description

	`<p>REPUBLIC OF SOUTH AFRICA</p>`
	`<p>`
	`<ActTitle>TRADITIONAL LEADERSHIP AND GOVERNANCE FRAMEWORK BILL</ActTitle>`
	`</p>`
	`<p>(As introduced in the National Assembly as a section 76 Bill; explanatory summary of`
	`Bill published in Government Gazette No. 25437 of 4 September 2003)</p>`
	`<p>(The English text is the offıcial text of the Bill)</p>`
	`<p>(<ActProponent>MINISTER FOR PROVINCIAL AND LOCAL GOVERNMENT</ActProponent>)</p>`
	`<p>`
	`<ActType>BILL</ActType>`
	`</p>`
	`<p>To <ActPurpose>provide for the recognition of traditional communities; to provide for`
	`the establishment and recognition of traditional councils; to provide for the`
	`functions and roles of traditional leaders; to provide a statutory framework for`
	`leadership positions within the institution of traditional leadership, the`
	`recognition of traditional leaders and the removal from office of traditional`
	`leaders; to provide for houses of traditional leaders; to provide for dispute`
	`resolution and the establishment of the Commission on Traditional Leadership`
	`Disputes and Claims; and to provide for matters connected`
	`therewith.</ActPurpose>`
	`</p>`
	`</preface>`

`preamble`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* The `preamble` element contains everything between the preface and the enacting terms of the act (included), such as the citations, the recitals and the solemn forms which precede and follow them. *Use in:* all document types that have preambles
`maincontent`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* The `maincontent` element contains all parts of the document that are between the preamble and the conclusions of the document. Note that `maincontent` is to be used with the `doc` and `report` element, and as such only for the MISC class of documents. *Use in:* Document doc only *Example:*
`conclusions`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* The `conclusions` element contains all parts of the document that conclude it, such as final forms, signatures, dates, and all other information follow the actual content of the document

Element name	**Description**
	Use in: all document types
attachments	*Content Model:* a list of attachment elements *Description:* The attachments element introduces a number of attachments associated to the document. It is a simple container, and all relevant information is contained in as many attachment elements as there are attachments. *Use in:* all document types *Example:* <attachments> <attachment href=”urn:AKOMA NTOSO:sa:act:117:1998:att1”/> <attachment href=”urn:AKOMA NTOSO:sa:act:117:1998:att2”/> <attachment href=”urn:AKOMA NTOSO:sa:act:117:1998:att3”/> </attachments>
attachment	*Content Model:* empty with *href* attribute *Description:* The attachment element refers to another AKOMA NTOSO document containing the content of the attachment. *Use in:* all document types *Example: see element* attachments

clauses	*Content Model:* a hierarchy of containment elements such as section, part, paragraph, chapter, article, etc.
	Description: The clauses element is the containing element for clauses in acts and bills and all other documents that rely on a hierarchical structure.
	Use in: acts and bills
	Example:
	<clauses>
	  <chapter id="chap01">
	    <num>CHAPTER 1</num>
	    <title>INTERPRETATION AND APPLICATION</title>
	    <article id="art01">
	      <num>1.</num>
	      <title>Definitions and application</title>
	      <clause id="art01-cla01">
	        <num>(1)</num>
	        <p>...</p>
	      </clause>
	      <clause id="art01-cla02">
	        <num>(2)</num>
	        <p>...</p>
	      </clause>
	    </article>
	  </chapter>
	  <chapter id="chap02">
	    <num>CHAPTER 2</num>
	    <title>TRADITIONAL COMMUNITIES AND TRADITIONAL COUNCILS</title>
	    <article id="art02">

Element name	Description
	<num>2.</num> <title>Recognition of traditional communities</title> <clause id="art02-cla01"> <num>(1)</num> <p>...</p> </clause> <clause id="art02-cla02"> <num>(2)</num> <p>...</p> </clause> <clause id="art02-cla03"> <num>(3)</num> <p>...</p> </clause> </article> </chapter> </clauses>
section	*Content Model:* a hierarchy of containment elements such as section, part, paragraph, chapter, article, etc. , preceded by, if needed, number and title. *Description:* a hierarchical element called section. Use it whenever appropriate within a clauses element. *Use in:* all hierarchical document types *Example: see element* clauses
part	*Content Model:* a hierarchy of containment elements such as section, part, paragraph, chapter, article, etc. , preceded by, if needed, number and title. *Description:* a hierarchical element called part. Use it whenever appropriate within a clauses element. *Use in:* all hierarchical document types *Example: see element* clauses
paragraph	*Content Model:* a hierarchy of containment elements such as section, part, paragraph, chapter, article, etc., preceded by, if needed, number and title. *Description:* a hierarchical element called paragraph. Use it whenever appropriate within a clauses element. *Use in:* all hierarchical document types *Example: see element* clauses
chapter	*Content Model:* a hierarchy of containment elements such as section, part, paragraph, chapter, article, etc. , preceded by, if needed, number and title. *Description:* a hierarchical element called section. Use it whenever appropriate within a clauses element. *Use in:* all hierarchical document types *Example: see element* clauses
article	*Content Model:* A list of clauses preceded by, if needed, number and title. *Description:* the article element is the basic unit of the hierarchy of the clauses part. *Use in:* all hierarchical document types

Element name	Description
	Example: see element `clauses`
`clause`	*Content Model:* A list of blocks preceded by, if needed, number and title. *Description:* The clause is the largest subdivision of the article. It contains the actual text of the acts and bills, organized in blocks. *Use in:* all hierarchical document types *Example: see element* `clauses`
`num`	*Content Model:* text and inline elements such as `b, i, span` *Description:* *Use in:* all hierarchical document types *Example: see element* `clauses`
`title`	*Content Model:* text and inline elements such as `b, i, span` *Description:* *Use in:* all hierarchical document types *Example: see element* `clauses`

`debate`	*Content Model:* a list of `subdivision` elements
	Description: The debate element contains all speeches, questions and discussion items that are present in reports and minutes.
	Use in: debate document types
	Example:
	`<debate>`
	`  <subdivision id="prayers">`
	`    <title>PRAYERS</title>`
	`  </subdivision>`
	`  <subdivision id="corrections">`
	`    <title>CORRECTION OF VOTES AND PROCEEDINGS AND THE OFFICIAL REPORT </title>`
	`    <speech id="spe01" by="per01">`
	`      <p>...</p>`
	`    </speech>`
	`    <speech id="spe02" by="per02">`
	`      <p>...</p>`
	`    </speech>`
	`  </subdivision>`
	`  <subdivision id="answers">`
	`    <title>ORAL ANSWERS TO QUESTIONS</title>`
	`    <subdivision id="minsport">`
	`      <title>MINISTRY OF EDUCATION AND SPORTS</title>`
	`      <subdivision id="que41">`
	`        <title>Professional and Trade Union Rights between GNAT and NAGRAT</title>`
	`        <question id="spe16" by="per07">`
	`          <p>...</p>`
	`        </question>`
	`        <speech by="per08" id="spe17">`

Element name	Description

	`<p>...</p>`
	`</speech>`
	`</subdivision>`
	`</subdivision>`
	`</subdivision>`
	`</debate>`

subdivision	*Content Model:* either a list of further `subdivision` elements, or a list of `speech` or `question` elements *Description:* The subdivision is a hierarchical organization of elements composing a debate. They can be either speeches, or questions to the assembly or to the government. They can be further organized hierarchically (for instance, to agree with the sections of an agenda). *Use in:* debate document types *Example: see element* `debate`
speech	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* A remark by a member of the assembly. The author of the remark is identified through the `by` attribute, which points to a listed person (see element `persons`). *Use in:* debate document types *Example: see element* `debate`
question	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* A question posed to the assembly or the government by an individual. The author of the question is identified through the `by` attribute, which points to a listed person (see element `persons`). *Use in:* debate document types *Example: see element* `debate`
item	*Content Model:* optional num, title and subtitle, followed by item elements and/or blocks (such as paragraphs and lists, or TOCs) *Description:* A discussion item in a parliamentary report or a daily tabling.. *Use in:* open document types

toc	*Content Model:* a list of `tocitem` elements
	Description: A table of content (TOC), typically provided in the preamble or the preface of a document. It contains a list of TOC items.
	Use in: all document types
	Example:
	`<toc>`
	`<tocitem idref="chap01">CHAPTER 1 INTERPRETATION AND APPLICATION</item>`
	`<tocitem idref="art01">1. Definitions and application</item>`
	`<tocitem idref="chap02">CHAPTER 2 TRADITIONAL COMMUNITIES AND TRADITIONAL COUNCILS</item>`
	`<tocitem idref="art02">2. Recognition of traditional communities</item>`
	`</toc>`

tocitem	*Content Model:* text and inline elements such as `b, i, span` *Description:* Each individual element of a table of content. The `idref` attribute points to the `id` value of the corresponding part in the actual document content. *Use in:* all document types

Element name	Description
	Example: see element `toc`
`ActType`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the type of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ActTitle`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the title of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ActNumber`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the number of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ActProponent`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the proponent of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ActDate`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the title of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ActPurpose`	*Content Model:* text and inline elements such as `b, i, span` *Description:* An inline element wrapping the purpose of the act. Use it typically in the preface. *Use in:* all document types *Example: see element* `preface`
`ref`	*Content Model:* text and inline elements such as `b, i, span` *Description:* A legislative reference (i.e., to another AKOMA NTOSO-maintained document). Use `ref` for legislative references, and use `a` for references to World Wide Web documents. *Use in:* all document types *Example:* `<li><def>district municipality</def> means a district municipality as defined in <ref href="urn:AKOMA NTOSO:sa:act:117:1998">section 1 of the Local Government: Municipal Structures Act, 1998 (Act No. 117 of 1998);</ref></li>`
`def`	*Content Model:* text and inline elements such as `b, i, span` *Description:* the defined term in a definition. This is usually presented in bold, and it could be useful to differentiate for searches and cataloguing operations. *Use in:* all document types *Example: see element* `ref`
`noteref`	*Content Model:* empty with a `href` attribute *Description:* a reference to an endnote or a footnote. The content of the note is actually in the notes subsection of the meta section. The `num` attribute contains the note marker that is shown in the main text. The `href` attribute provides a pointer to it.

Element name	Description
	Use in: all document types *Example: see element* `notes`
`recordedTime`	*Content Model:* empty with a `time` attribute *Description:* the specification of the time in which a remark, discussion item or question was proposed. The `time` attribute contains the time itself in format `hh:mm`. *Use in:* all document types *Example:* `<p><recordedTime time="14:04"/> The Chief Whip of the Majority Party moved: That the House resolves that—</p>`
`mod`	*Content Model:* text, inline elements such as `b, i, span`, plus two special elements for quotations: `quotedText` and `quotedStructure`. *Description:* The `mod` element contains the part of an act that provides explicit instruction for an amendment to another act. *Use in:* all document types
`quotedText`	*Content Model:* text and inline elements such as `b, i, span` *Description:* a fragment of text being either the substituted text or the substitution text of an amendment. Only subparts of a whole structure (e.g., one or a few words) are placed in a `quotedText` element. Use `quotedStructure` for whole structures. *Use in:* all document types
`quotedStructure`	*Content Model:* all types of structures: hierarchical elements such as whole parts, articles, clauses, etc. plus any kind of blocks or containers. *Description:* a substructure of the act that is the part being substituted or the substitution itself of an amendment. Only whole structure (e.g., an article, a clause, a block) are placed in a `quotedStructure` element. Use `quotedText` for individual words. *Use in:* all document types
`hcontainer`	*Content Model:* a hierarchy of containment elements such as `section, part, paragraph, chapter, article`, etc. , preceded by, if needed, number and title. *Description:* A generic element to be used as a placeholder of any hierarchical element within a clause element. *Use in:* all hierarchical document types *Example: see discussion in sect 6 of this document*
`container`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* A generic element to be used as a placeholder of any container element wherever containers are allowed. *Use in:* all document types *Example: see discussion in sect 6 of this document*
`block`	*Content Model:* text and inline elements such as `b, i, span` *Description:* A generic element to be used as a placeholder of any block element wherever blocks are allowed. *Use in:* all document types *Example: see discussion in sect 6 of this document*
`inline`	*Content Model:* text and inline elements such as `b, i, span` *Description:* A generic element to be used as a placeholder of any inline element wherever inlines are allowed. *Use in:* all document types

Element name	Description
	Example: see discussion in sect 6 of this document:
marker	*Content Model:* markerreq *Description:* A generic element to be used as a placeholder of any marker element wherever markers are allowed. *Use in:* all document types *Example: see discussion in sect 6 of this document*
div	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* A generic element to be used as a placeholder of any container element wherever containers are allowed. Note: use as a container, not as a block (as HTML would recommend). *Use in:* all document types *Example: see discussion in sect 7 of this document*
p	*Content Model:* text and inline elements such as b, i, span *Description:* as in HTML *Use in:* all document types
li	*Content Model:* text and inline elements such as b, i, span, plus other ul and ol elements. *Description:* as in HTML *Use in:* all document types
span	*Content Model:* text and inline elements such as b, i, span *Description:* as in HTML *Use in:* all document types
b	*Content Model:* text and inline elements such as b, i, span *Description:* as in HTML *Use in:* all document types
i	*Content Model:* text and inline elements such as b, i, span *Description:* as in HTML *Use in:* all document types
a	*Content Model:* text and inline elements such as b, i, span *Description:* as in HTML *Use in:* all document types
img	*Content Model:* empty plus the src and alt attributes. *Description:* as in HTML *Use in:* all document types
ul	*Content Model:* a list of li elements *Description:* as in HTML *Use in:* all document types
ol	*Content Model:* a list of li elements *Description:* as in HTML *Use in:* all document types
table	*Content Model:* a list of tr elements *Description:* as in HTML *Use in:* all document types
tr	*Content Model:* a list of either th or td elements *Description:* as in HTML

Element name	Description
	Use in: all document types
`th`	*Content Model:* text and inline elements such as `b, i, span` *Description:* as in HTML *Use in:* all document types
`td`	*Content Model:* text and inline elements such as `b, i, span` *Description:* as in HTML *Use in:* all document types
`meta`	*Content Model:* a `descriptor` elements, a `lifecycle` element (optional), a `persons` element (optional), zero or more `notes` sections, zero or more `proprietary` sections. *Description:* The meta element contains the whole set of metadata defined for each AKOMA NTOSO document. All metadata are organized in subsections according to purpose and use. *Use in:* all document types
`descriptor`	*Content Model:* `publication, otherpublications` (optional), `enactmentDate, editors, uri` (repeatable), `alias` (repeatable and optional), `keywords` (optional) *Description:* The descriptor container collects all official and objective information *about* the document that does not actually *belong* to the document. This include naming and publications details, as well as keywords. *Use in:* all document types
`publication`	*Content Model:* empty with `type, num` and `date` attributes *Description:* Details about the official publication where the act or document first appeared. It is assumed that there is **one** authoritative publication, and optionally a few more additional publication venues. The type of publication, number (if the publication has issues) and date are required attributes. *Use in:* all document types *Example:*
`otherpublications`	*Content Model:* one or more of `ripublication` and `errata` elements *Description:* Sometimes official documents are published several times, possibly in different venues, possibly with slightly different content. Sometimes a corrected version is published, other times just a list of errata is provided. Information about these additional publications must be described here. *Use in:* all document types
`ripublication`	*Content Model:* empty with `type, num` and `date` attributes *Description:* Non-authoritative secondary publication of the act. The type of publication, number (if the publication has issues) and date are required attributes *Use in:* all document types
`errata`	*Content Model:* empty with `type, num` and `date` attributes *Description:* Publication of errata on the authoritative publication of the act. Use ripublication for additional publication of the *corrected* text. The type of publication, number (if the publication has issues) and date are required attributes *Use in:* all document types
`enactmentDate`	*Content Model:* empty with `date` attribute *Description:* The date from which the act is actually enacted. This date is only relevant when it is different from the publication date, and therefore for acts only. In all other cases, it is basically the publication date. Nonetheless, it is a reliable source of information about the starting date of validity of the document. *Use in:* all document types
`editors`	*Content Model:* (editor+) *Description:* The list of human editors that have worked on the AKOMA NTOSO version of the

Element name	Description

	document.
	Use in: all document types
	Example:
	`<editors>`
	`<editor id="ed01" date="2005-09-17" name="FV" contribution="markup"/>`
	`<editor id="ed02" date="2006-01-12" name="FV" contribution="editing"/>`
	`</editors>`

`editor`	*Content Model:* required `id, date, name` attribute, plus an optional `contribution` attribute. *Description:* Each individual editor that has contributed to the current AKOMA NTOSO document. The contribute is an arbitrary string detailing what kind of contribution the editor has provided. Possible values include: markup, editing, proof reading, etc. *Use in:* all document types *Example: see element* `editors`
`uri`	*Content Model:* empty with `href` attribute. *Description:* The official URI of the document. *Use in:* all document types *Example:* `<uri href="urn:AKOMA NTOSO:sa_bill_76_20030904_en"/>`
`alias`	*Content Model:* empty with `value` attribute. *Description:* Alternative names for the document. *Use in:* all document types
`keywords`	*Content Model:* (keyword+) *Description:* A list of keyword elements. The `dictionary` attribute controls the source dictionary of keywords that is being used. *Use in:* all document types
`keyword`	*Content Model:* empty with `value` attribute. *Description:* Each individual keyword according to the dictionary specified. *Use in:* all document types

`lifecycle`	*Content Model:* (events?,references)
	Description: The `<lifecycle>` element contains two main structure to determine the life cycle of the document: the list of events that the current document has undergone, and the list of references that have generated these events.
	Use in: all document types
	Example:
	`<lifecycle>`
	`<events>`
	`<event id="evn1" source="org01" date="2005-08-29" type="Generation"/>`
	`<event id="evn2" source="pas01" date="2005-12-31" type="Repeal"/>`
	`</events>`
	`<references>`
	`<reference type="Original" href="urn:AKOMA NTOSO:sa_act_76_20030904_en" id="org01"/>`
	`<reference type="Passive" href="urn:AKOMA`

Element name	Description

	`NTOSO:sa_act_97_20051231_en" id="pas01"/>`
	`  </references>`
	`</lifecycle>`

`events`	*Content Model:* (event+) *Description:* A list of events *Use in:* all document types *Example: see element* `lifecycle`
`event`	*Content Model:* empty with required `source, date` and `type` attributes. *Description:* Each event that affects the history and lifecycle of the document. Each event is caused by one document, which is listed in the references list and is referred to by the `source` attribute. The `date` attribute specifies the date from which the event takes place, and the `type` attribute specifies the type of event. *Use in:* all document types *Example: see element* `lifecycle`
`references`	*Content Model:* (reference+) *Description:* A list of references. *Use in:* all document types *Example: see element* `lifecycle`
`reference`	*Content Model:* referenceType *Description:* A reference to a document that affects for some reason the current document. These documents may be the original document, or referred to by the document (active references) or refer to the current document (passive document), or referred to by some jurisprudence. *Use in:* all document types *Example: see element* `lifecycle`

`persons`	*Content Model:* (person+)
	Description: All persons described in a debate are fully represented here. This list allows to use much shorter references to individuals in the actual debate minutes, and furthermore allow debates to avoid spelling differences in the name of the same person. The `by` attribute in `<question>` and `<speech>` elements refer to individuals whose full and short name are provided here.
	Use in: all document types
	Example:
	`<persons>`
	`  <person id="per01" short="Mr. Speaker" long="Mr. Speaker"/>`
	`    <person id="per02" short="Mr. Opare-Hammond" long="Mr. Kwadjo Opare-Hammond"/>`
	`   <person id="per07" short="Mr. J.K. Gidisu" long="Mr. Joe Kwashie Gidisu"/>`
	`</persons>`

`person`	*Content Model:* empty with `short` and `long` attributes *Description:* The short and full name of individuals whose speeches and questions are reported in the debate document. *Use in:* all document types *Example: see element* `persons`
`notes`	*Content Model:* (note)+ *Description:* A list of editorial notes (footnotes, end-notes, etc.)

Element name	Description

	Use in: all document types
	Example:
	`<p> THE REPUBLIC OF GHANA<noteref href="#not01" num="1"/></p>`
	`...`
	`<notes>`
	`    <note id="not01"><p>A test footnote</p></note>`
	`</notes>`

`note`	*Content Model:* blocks (such as paragraphs and lists, plus TOCs) *Description:* An editor node connected via id to the corresponding note marker expressed by the `<noteref>` element. *Use in:* all document types *Example: see element* `notes`
`proprietary`	*Content Model:* ANY *Description:* Any arbitrary number of elements (not belonging to the AKOMA NTOSO namespace) that individual editors feel the need to add to AKOMA NTOSO documents. Applications are free to either correctly consider these elements, or just completely ignore their existence. *Use in:* all document types

12 Attribute Synopsis

In the following table, a complete list of attributes defined in this release of AKOMA NTOSO and their expected usage is provided. ***TBD***

Attribute	Type	Appears in	Description
`alt`	string		As in HTML
`border`	integer		As in HTML
`by`	idref		A reference to the `<person>` element who is providing the speech or question.
`cellpadding`	integer		As in HTML
`cellspacing`	integer		As in HTML
`class`	string		As in HTML
`colspan`	integer		As in HTML
`contains`	VersionType		One of the values: OriginalVersion, SingleVersion or MultipleVersions. See section 10 for details.
`contribution`	string		An arbitrary string detailing the contribution of an editor.
`date`	date		A date expressed in syntax YYYY-MM-DD.
`dictionary`	string		The name pf the dictionary being used for a set of keywords.
`end`	idref		A reference to an event that is the end of the enactment period of a fragment of a document (such as an article being repealed)
`height`	integer		As in HTML
`href`	anyURI		A reference to an external document, as in HTML. The href attribute is also used for internal references. In this case prefix the id being referenced with the # character.
`id`	id		An identifier for the section or element. See section 9 for details.
`idref`	idref		A reference to a section or an element. See section 9 for details.

long	string		The long name of a person
name	string		The actual name of a generic element. See section 6 for details.
num	string		The visible part of a note reference. It can be a number, a bracketed number (e.g. [2]) or in fact any sequence of characters.
rowspan	integer		As in HTML
short	string		The short name of a person
source	idref		The document originating an event or the editor originating a set of proprietary metadata
src	anyURI		As in HTML
start	idref		A reference to an event that is the start of the enactment period of a fragment of a document (such as an article being inserted)
status	statusType		The current status of a section or element in a modified act. The values repealed and suspended refer to actual, official enactment of the element, while omissis refers to the fact that, for editorial reasons, the content of the element has been omitted from this edition. The omissis value is **never** to be used in official editions of the document.
style	string		As in HTML
target	string		As in HTML
time	time		The time of an event as recorded. The value should be expressed in the syntax: HH:MM:SS.
title	string		As in HTML
type	EventType		The type of event, reference, or publication.
uri	string		An optional uri of the organization for which the editor has done some editing.
value	string		A value for a metadata element.
width	integer		As in HTML

13 The akomantoso.xsl and style.css stylesheets

The current release of AKOMA NTOSO 1.0 also includes an example stylesheet for generating XHTML files out of AKOMA NTOSO XML documents. The stylesheet can be used both server-side (within a Perl, PHP, ASP, Java, or Python application) and client-side (delivering to the requesting browsers the XML file containing the stylesheet URL, as in the examples proposed here). The stylesheet has been successfully tested on the following browsers:

- Internet Explorer v. 6.0.2 SP1 for Windows XP
- Mozilla Firefox v. 1.0.7 for Windows XP
- Mozilla Firefox v. 1.5 for Macintosh OS X
- Apple Safari v. 1.3.1 for Macintosh OS X.

The XSLT stylesheet proposed here is an initial example, incomplete and graphically non sophisticated. Yet, it shows a few approaches that can be used fruitfully for final stylesheets, as they automatically manage the variability in element names and local habits that will be found in real-life African-wide implementations.

The proposed stylesheet exploits content model patterns systematically, radically reducing the length (and therefore the complexity) of the stylesheet. Thus, rather than having a template for each individual AKOMA NTOSO element, we have a template for every content model pattern of the schema: hierarchical structures, containers, blocks, inlines, markers. A special template is also provided for elements drawn from the HTML language, that simply converts the element into its HTML correspondent.

Variability in presentation is handled by systematically adding a specific CSS class that can be exploited for presentational differences. Variability in structuring is handled by means of DTD entities within the XSLT stylesheet itself. This is a sufficiently unusual approach as to deserve a more detailed explanation.

Some elements may require transformations that depend on the actual African country that uses it. For instance, the same element may have a different hierarchical level in different countries, and thus require different presentation characteristics. We have identified a few examples of such variability:

- The **window title**: what part of the document is to be used in the window title (and therefore is stored by bookmarks and search engines).
- The **overall layout**: document with a strong hierarchical structure may require a navigation aid in the form of a table of content always available on a side bar. Other documents can just be drawn as a long and continuous single column.
- The **presentation of the hierarchy**: the beginning of each subsection depends on the relative importance: for instance, a major section will be shown by separating in different lines the number, the title, and the content (which we term the N-T-C presentation, where N, C and T represent number, title and content, respectively, and the dash refers to the existence of a line break), or place on the same line the number and the title, and on a further line the content (the NT-C presentation), other may lack the number and present the number close to the content (the NC presentation). And other may lack numbers and just have on separated lines the title and the content (the T-C presentation).
- The **content of the table of content**: usually, the table of content only contains the first few levels of the hierarchy. Not all hierarchy levels thus need to end in the automatic table of content.

Since the list of element names that impact on the above-mentioned may vary from country to country, and need to be detailed in several places of the XSLT stylesheet, the DTD entity approach has been taken: the templates actually just contains a reference to an entity, and at the beginning of the stylesheet the definition of such entities is provided. By just changing these definition, it is possible to immediately and automatically update the whole presentation according to the local presentation rules, without modifying the whole stylesheet.

Customizing the `styles.css` CSS stylesheet is also appropriate to obtain a fully customized presentation of AKOMA NTOSO documents.

14 Differences from previous releases

14.1 Differences between release 25/04/2006 and release 15/01/2006

- All references to PAPI have been removed and substituted with Akoma Ntoso, both in the schema documents and the documentation (in the following forms: `akomantoso`, AKOMA NTOSO, and AN).
- After registering the proper domain name, the namespace for Akoma Ntoso documents is now `http://www.akomantoso.org/1.0`.
- All references to equivalences, both in the schema documents and documentation, have been removed. Schema is now fully and exclusively in English.

14.2 Differences between release 15/01/2006 and release 15/11/2005

- PAPI is now **really** specified as version 1.0 instead of 2.0. Correspondingly, the namespace for this document class is now **really** defined as `http://www.parliaments.info/PAPI/1.0`.
- The `MISC` category of document is now called simply `<doc>`. `<doc>` elements are to be used to specify documents that are neither acts (or having an act-like structure) nor debates (or having a debate-like structure). The previously existing document class `<doc>`, has been completely reorganized and restructured, by modifying the underlying content model, `&OpenStructure;`. Furthermore, existing document class `<report>` has been moved into the `&OpenStructure;` content model from `&DebateStructure;`.
- Element `<item>` has completely changed role and content model, being now a hierarchical element providing support for a hierarchy of items. This is the main structure for hierarchies that are not legislative and thus are contained in generic `<doc>` elements.
- Debates (as specified with the `<subdivision>` element) can now only contain just `<speech>` and `<question>` elements, since the `<item>` element has been reorganized for a different purpose and a different hierarchy.
- The element `<maincontent>`, the backbone of the `&OpenStructure;` content model, has been completely redesigned. Instead of containing just block elements, it can now contain block elements, juridical hierarchical elements, and debate subdivisions and item hierarchies.
- A new element `<subtitle>` has been added for hierarchical structures that contain subtitles in addition to number and title.
- A new attribute has been added, `numbering`, to elements `<maincontent>` and `<item>`, for requesting that elements of a hierarchical structure are numbered by the displaying application, rather than carry their own numbers in the XML source.
- Element `<tocitem>`, containing individual items of a table of content, now has an additional required `level` attribute to specify the hierarchical level of the `<tocitem>` element.
- Metadata elements `<uri>` and `<alias>`, that in the previous versions had a text content model, now are markers, and have the corresponding value expressed in the `value` attribute. This definitely and completely aligns all metadata elements to the marker pattern, in order to avoid improper display of their values by unsuspecting XSLT stylesheets.

14.3 Differences between release 15/11/2005 and release 15/09/2005

- PAPI is now specified as version 1.0 instead of 2.0 (references to previous attempts at PAPI have been removed). Correspondingly, the namespace for this document class is now defined as http://www.parliaments.info/PAPI/1.0.

- Two new document classes have been added, <report> and <minutes>, to handle, respectively, *Official Reports* (or *Hansards*) and *Official Minutes* (or *Votes and Proceedings*). These two document classes use a new document structure, DebateStructure, that is added to HierarchicalStructure and OpenStructure.
- Three new special elements have been added to handle the content of reports and minutes: speech, question and item. They are collected in a hierarchical structure of subdivisions, that provide nesting for such elements.
- A new marker element has been added, recordedTime, to handle the specification, anywhere in the text, of the moment in which the remark, agenda item or question was proposed.
- A new section of meta elements has been added, persons, to list all the people whose remarks have been recorded in the minutes or reports.
- Element item, within TOC (Table Of Content), has been renamed tocitem to avoid clashes with debates' items. Also, element TOC has been converted into lowercase for consistency with other element names.

15 References

[1] Pan African Parliamentary Interoperability (AKOMA NTOSO) Report and Documentation http://www.parliaments.info/AKOMA NTOSO/docs/AKOMA NTOSO%20Draft%20Proposal%20V%202.2.pdf

[2] F. Vitali, A. Di Iorio, D. Gubellini, *Design patterns for document substructures*, Extreme Markup 2005 Conference, Montreal, 1-5 August 2005, http://www.mulberrytech.com/Extreme/Proceedings/xslfo-pdf/2005/Vitali01/EML2005Vitali01.pdf

[3] http://www.xmlpatterns.com/