eContracts Structure Markup – Preliminary Report

eContracts Structure sub-committee

Editor: Jason Harrop, jharrop@speedlegal.com

Sub-committee members:
Jason Harrop, jharrop@speedlegal.com
Peter Meyer, pmeyer@elkera.com.au
Mary McRae, mmcrae@dmsi-world.com
John McClure, jmcclure@hypergrove.com
Dr Laurence Leff, D-Leff@wiu.edu

last revised: 19 July 2004

Summary of changes: The major changes since 12 April reflect Peter Meyer and Jason Harrop's suggestions regarding the handling of the top level structure, which includes things such as:

- Date
- Parties
- Recitals/Background
- Operative Clauses
- Signatures
- Schedules/Annexures

Proposed next steps:

- TC to review work to date, and if appropriate, endorse. If necessary, consider separately:
  - section model (part A1 herein)
  - paragraph level (part B)
  - high-level structure (parts D1, D2)

TC to consider and decide on process for resolution of issues affecting these areas:

  - table (part A4) and style attribute set (part C4)
  - party detail markup
  - signatures (D3)

  - numbering and cross references (A2 and B6), including hypertext attribute set (C6)

  - headers/footers (D8) and page breaks (D9)

- cover page (D7)
- choice of definitive grammar (DTD, XML Schema, or Relax NG)

Background

This subcommittee was asked by the TC to develop a structural makrup model based on the proposed XHTML 2.0 model under development by the W3C.

The structural markup model was expected to cover those issues set out in the clause model requirements document dated 27 May 2003 and subsequently adopted by the TC.

To achieve this, the subcommittee set out to adapt the XHTML 2 model to simplify it, remove unnecessary elements and attributes and add to it where considered necessary to provide a structural representation of contract documents.

Broadly, the proposed aims of the structural markup are to:

- Avoid dependency on any particular style language or rendering applications.
- Define the hierarchical relationship of generic components of contract documents, as understood by lawyers, without imposing legal meaning.
- Keep the markup as simple as possible to facilitate use of XML markup by contract drafters where this is desired by user organizations. Even if this is not common today, it is proposed that the markup should not create a barrier to this development.
- Allow the widest range of users to create contract documents.
- Provide sufficient markup so that a user organization can define style sheets in a chosen styling language to render all conforming documents in an identical layout, without inclusion of explicit styling information in the markup (except for possible allowance of emphasis, table layouts and similar).
- Provide structure for the reliable application of automatic numbering to contract components.
- Provide context for the reliable insertion or extraction of contract information by processing applications (but not define metadata or inline markup as these are out of scope for the Subcommittee).

Our discussions to date have been based on the XHTML 2.0 working draft of May 2003. We had understood there was to be a new public draft around the first week of April, which addresses all issues identified in the last Working Draft, but introduces new issues of its own. As of 17 July, this document is not yet available, but we now understand a new public draft is expected in the week of 19 July.

A. "Clause" model

1. <section> model changes

Our model is:

```
section (nr?, h?, p*, section*)
```

Notes:

- <section> is the container element for any heading and paragraphs of text which make up the clause. So ordinarily, a clause looks like:

```
<section>
        <h>
```

```
            Optional heading
        </h>
        <p>
            Clause content ...
        </p>
        ...
    </section>
```

Section can be used recursively to create subject groupings (parts, divisions, etc) within the document.

See the New Orleans (April 2004) presentation for further examples.

- #PCDATA is not allowed directly in <section>. Contrast this with XHTML 2.0 WD, which uses #PCDATA and the Flow entity in the section content model.
- The list elements are not allowed in <section> - they are only allowed in <p>. This is tighter than XHTML 2.0 WD.
- remove <h1>-<h6>, <blockcode> and <pre> from the definition of the Block entity
- The model is looser than (h?, (section+|p+)), but tighter than (h?, (p|section)*). Specifically, the model does not allow a trailing paragraph. TODO: for the case where that trailing paragraph is a note, consider how it might be represented.
- <nr> is for the section number.

Issues:

- John would like to revisit whether nr is optional or mandatory.

2. Numbering generally

- TODO: What are our requirements? It is agreed that explicit numbering must be available (but optional) for the markup of legacy content, and also for consistent numbering when markup is exchanged between different eContract applications.We need:
  - section numbering,
  - schedule numbers
  - schedule item numbering
  - list item numbering (block and inline)
  - table numbering?
  - image numbering?
  - table cell numbering? Paragraph numbering in a table cell?
- We have introduced <nr> as our numbering element. TODO: Its content model is yet to be discussed.
- Agreed that in general, number is better as an element than as an attribute. Leaves the way open to put inline markup in the number text, if we decide we want that.
- TODO: [Interested parties to write proposals as to how this – and the related issue of cross references - might work, as basis for discussion]

3. Block quote

- TODO: what are our requirements: multiple paragraphs? trailing citation? Need some samples to consider
- TODO: Content model for <blockquote>?

- Assume <blockquote> will live in <p>, and not <section>. It could then also go in <li> only indirectly via <p>.

4. Tables

Note:
- <table> will live in <p>, and not in <section>
- the WD content model for a table cell (<td>) is (PCDATA | Flow)*. TODO: Will we make it (section | p)* ? If we include section, the content of table cells can be numbered.
- TODO: how is the table to be styled (eg cell borders)? Can we avoid a CSS dependency?

  According to the W3C representative we spoke to, the W3C likes to maintain the fiction that there are no dependencies on CSS. But user agents are expected to behave as described by CSS.

  W3C Rep says "great" if we want to support multiple styling languages.

- Proposed that we need to be able to number tables.

5. Images

The WD says: "The Object Module provides elements for general-purpose object inclusion; this includes images and other media, as well as executable content. .. When this module is used, it adds the object element to the Inline content set of the Inline Text module."

- it uses CSS for height and width, and placement.
- The Image module from XHTML 1 is gone – there is no image element anymore. (It used to add the image to inline text.)
- TODO: decide whether to use <object>, and if so, how.
- image (whatever its element is) will live in <p>, and not in <section>
- Proposed that we need to be able to number images.

6. <Div>

Section 8.4 of the WD says: "The div element, in conjunction with the id and class attributes, offers a generic mechanism for adding extra structure to documents. This element defines no presentational idioms on the content. Thus, authors may use this element in conjunction with style sheets , the xml:lang attribute, etc., to tailor XHTML to their own needs and tastes."

Like <section>, the content model for <Div> is (#PCDATA | Flow)*

TODO: We need to talk about what <Div> would be used for in the context of contracts, where it should be allowed, and what its content model should be. One possibility is for organisations to use it to extend the structure to meet any specific needs they may have.

B. Paragraph level considerations

1. Working model for <p>

```
p (#PCDATA | table | img | blockquote  | %Inline; | %List;)
```

where %Inline includes <l>. See further B2 and B4 below.

where %List includes <ol>, and an inline list element.

2. <l> in <p>

Agreed that <l> is to be optional in <p>

Note:
- There is no
 element, so one must use <l> or the Unicode line separator (should we say this is (good or) bad practice?).
- TODO: What is the difference in appearance (if anything) between the following:

```
<p>     Some text
        <l>some more text</l>
        Even more text
</p>

<p>     <l>Some text</l>
        <l>some more text</l>
        <l>Even more text</l>
</p>
```

3. Block lists

```
ol (li*)
li (nr?, h?, p*)
```

Notes:
- remove dl, nl, and ul.
- Don't allow PCDATA directly in

4. Inline Lists

Since we don't want <p> in an inline list, we need either a local definition for li, or a different element for an inline list. We decided on a different element.

Content model:

```
inlinelist (ili)*

ili (#PCDATA | inlinelist)*
```

- TODO: choose element names ("ilist"?)
- TODO: consider whether we need an inlinelist element, or whether ol (li* | ili*) would suffice (ie just an inline list **item**). Do we need a container for inline list items at all?
- TODO: list item numbering (note that if we say:

```
inlinelist (nr?, ili)*
```

then nr is outside the list item (which is different to how it is for section). It would be possible to use an attribute for the number, but we're all agreed that in general, number is better as an element than as an attribute.

- TODO: confirm headings aren't required

5. Irrelevent elements in the %Inline; content set

- remove <code>, <kbd>, <samp> and <var> from the definition of the Inline entity
- keep <abbr>, <sub> and <sup>
- keep cite | dfn | em | quote | span | strong

TODO: revisit suitability of cite and dfn.

6. Cross references (in conjunction with numbering above)

- It is proposed that we need to be able to do cross references to images and/or tables.

C. Attributes

The common attributes (numbering some 29 in all) are attached to each and every element, even though they are largely unecessary in contracts. Those that aren't necessary to represent the contract are to be removed.

1. Property attribute

We understand this attribute will be introduced in the next Working Draft. Its purpose is to help the RDF world work in concert with the HTML world. TODO: We will consider this further when we see the next Working Draft.

2. Events attribute set

Agreed that we will omit the events attribute set

3. map attribute set

Agreed that we will omit the map attribute set

4. style attribute set

<!ENTITY % style "style #CDATA #IMPLIED">

WD 6.9 says: "The syntax of the value of the style attribute is determined by the default style sheet language. For example, for [CSS2] inline style, use the declaration block syntax described in the Style Sheet Module (without curly brace delimiters).
This CSS example sets color and font size information for the text in a specific paragraph.

Aren't style sheets wonderful?
:

Note that the Style collection is only defined when the Style Attribute Module is selected. Otherwise, the Style collection is empty."

- TO DO – FURTHER DISCUSSION: Do we need this attribute? Or is it sufficient to define styles externally in a stylesheet (relying if necessary on @class or even @ID)?

5. core attribute collection: class, id and title attributes

"class = NMTOKENS
This attribute assigns one or more class names to an element; the element may be said to belong to these classes. A class name may be shared by several element instances.

The class attribute can be used for different purposes in XHTML, for instance as a style sheet selector (when an author wishes to assign style information to a set of elements), and for general purpose processing by user agents."

- Agreed we will keep @class. TODO: discuss whether this can take arbitrary values, or whether there is a list of available choices.
- What elements do we need it on?

"id = ID
The id attribute assigns an identifier to an element. The id of an element must be unique within a document.

The id attribute has several roles in XHTML:

- As a style sheet selector.
- As a target anchor for hypertext links."

- Agreed we will keep @id #IMPLIED.

"title = Text
This attribute offers advisory information about the element for which it is set."

- Agreed that we will omit @title. Wherever it might otherwise be useful, there is likely to be an "h" element, or metainformation.

6. hypertext attribute set

Do we need the hypertext attributes, when hypertext will really only be used for internal cross references and external citations?

```
<!ENTITY % hypertext        "href          %URI;          #IMPLIED
                             cite          %URI;          #IMPLIED
                             target  %HrefTarget;  #IMPLIED
                             rel           %LinkTypes;    #IMPLIED
                             rev           %LinkTypes;    #IMPLIED
                             accesskey     %Character;    #IMPLIED
                             navindex      %Number;       #IMPLIED
                             xml:base      %URI;          #IMPLIED">
```

TODO: revisit the hypertext attribute set in the context of cross references. Leave it in for the moment.

7. Query the TC's stance on languages other than English:

```
<!ENTITY % i18n       "xml:lang      %LanguageCode;     #IMPLIED">
<!ENTITY % bi-direct "dir            (ltr | rtl
                                     | lro | rlo)       ltr">
```

Agreed to leave in for now.

8. Query the TC's stance on change tracking:

```
<!ENTITY % edit       "edit          (inserted
```

```
                    | deleted
                    | changed
                    | moved)        #IMPLIED
      datetime      %Datetime;     #IMPLIED">
```

Agreed that we will omit these attributes. Either this is outside the scope of the TC, or if it is in scope, these attributes aren't a sufficient solution.

9. Embedding attribute collection (6.6)

This is a mechanism for content reuse by reference.

Agreed to omit, in part because of the Mischief of "content processed instead"

D. Highest Level Content Additions

1. Root element

We have agreed to introduce an “instrument” element to represent documents included in schedules etc.

We have agreed to re-use this element name for the eContract itself. We have chosen a name which reflects the fact that this document type has wider applicability than just contracts, but at the same time, we didn’t want something as wide a “business document”.

2. top level contract structures

It is necessary to be able to represent the high level structure of a contract. Typically, this looks something like:

- Date
- Parties
- Recitals/Background
- Operative Clauses
- Signatures
- Schedules/Annexures

But clearly other arrangements are possible.

As far as Date and Parties are concerned, these appear to be laid out in different patterns, depending on jurisdiction:

- USA

  On the evidence available to the sub-committee, in American contracts, the date and parties information is typically all contained in a single paragraph.

- UK, ANZ

  In English and Australian/New Zealand contracts, the date and party information is typically presented in separate paragraphs. See the samples in appendix 2.

It was noted that this high level structure is “real” to lawyers and others who work with contracts, and that XML representations of them serve a variety of purposes, including:

- navigation of the document
- control of clause numbers
- generation of cross references
- document styling, including page breaks
- generation of content of running headers/footers
- context for contents generation

Peter Meyer and Jason Harrop propose "named containers", as follows:

```
instrument (h*, extra?, ( p* | (datearea, parties) ),
                background?,  operative, (signatures | extras)* )
```

Notes:

- h*, so that it is possible to have a heading and a sub-heading
- The single <extra> at the beginning of the instrument content model is for the reference schedule sometimes found at the front of a contract.
- The alternative:

  (p* | (datearea, parties))

  is necessary to accommodate the difference between the USA and UK approaches to date/parties.

  US users would likely simplify the structure to:

  instrument (h*, extra?, p*, background?, operative, (signatures | extras)*)

  and put the content in a , for example:

  ** **This License Agreement is made this 12th day of January 2001, by and between the National Association of Realtors, an Illinois not-for-profit corporation which has its principal place of business at 430 North Michigan Avenue, Chicago, Illinois 60611 ("NAR") and Larry Liquour ("Licensee").** **

  UK/ANZ users would likely simplify it to:

  instrument (h*, extra?, datearea, parties, background?, operative, (signatures | extras)*)>

  with:

  ```
  datearea (h?, p*)

  parties (h?, (party | p)* )

  party ( l | #PCDATA)*
  ```

  See the sample in appendix 3. Note that is included in <parties> for the case where "BETWEEN" and "AND" appear on lines by themselves.

  As shown in appendix 2, the presentation options vary greatly in the way that the connecting words “BETWEEN” and “AND” are handled. A complete structural markup model might require one or more specific elements for these words so that a rendering application could create all the possible layout arrangements. Alternatively, tables could be used.

  Jason Harrop and Peter Meyer consider that it is not desirable to try to define a generic element for these components because of their limited function. Rather, they propose that these connecting words can be generated in layouts by the rendering application, if so desired by the user. Under such an approach, the parties element might only contain a series of party elements. The words “BETWEEN” and “AND” or similar words would not be present in the markup.

To illustrate how the complete model might look, possible inline markup for the party names and addresses is sketched out in that appendix. Party related markup is different from most other semantic markup, in that it is immediately and obviously useful in an authoring environment.

- <background> is for the recitals. Its content model is:

```
background (nr?, h?, section*)
```

The recitals are made up of sections. There are typically no opening paragraphs between the recitals heading and the recitals.

- The content model for the operative clauses allows for one or more opening paragraphs before the clauses proper:

```
operative (nr?, h?, p*, section*)
```

- <extras> is for schedules/annexures etc. The content model for extras is simply:

```
extras (h?, extra*)
```

The extras container carries an attribute which says whether its <extra> elements are

```
(schedules | attachments | annexures | appendices | exhibits)
```

"schedules" is proposed as the default.

The content model for extra is:

```
extra (nr?, h?, p*, (section* | instrument*) )
```

By means of <instrument>, a document can be attached in a schedule.

- The model allows the signatures to come before the schedules and annexures, after them, or between them.

Peter Meyer and Jason Harrop favour this "named container" approach since it makes it easy for authors to create contracts in XML using standard XML authoring tools.

The alternative the sub-committee considered is a generic container:

instrument (h*, struct*)

struct (nr?, h?, p*, (section* | struct* | instrument*))

We considered: instrument (h*, p*, section*, struct*), but decided it was better not to allow p* or section* directly within <instrument>

The generic struct container would have to be recursive to represent both a schedules element and the schedule elements contained within it.

The generic container would need to carry an attribute identifying whether it contains for example, operative clauses or a schedule or the recitals.

The critical problem with this approach is that it is not easy to stop someone doing this:

```
<struct class="schedules">
        <struct class="operativeclauses">
        :
        </struct>
</struct>
```

(ie putting the main operative clauses inside the schedules container, which should only contain schedule elements)

or, for example, from putting schedules or recitals inside their operative clauses.

A content model which permits this:

- makes life difficult for our TC, since we need to document the various possible forms of markup and how they are to be treated by processing tools
- makes life difficult for tool providers, since ease-of-use demands that they program around the unintended possibilities
- makes it hard for authors to use generic tools, since their desired alternatives are mixed up with non-sensical ones
- makes life difficult for stylesheet developers, since they need to handle the unintended combinations authors will create when they use a generic tool.

For these reasons, the structural model should where practical restrict valid content to exclude non-sensical constructs.

One could use a single generic container and avoid the problems mentioned above, if RELAX NG was employed (rather than a DTD or XML Schema). However, the authoring tool support which would be necessary for widespread adoption of our standard is not yet available.

For these reasons, the named container approach is to be preferred. The named container approach works well with current schema languages and XML toolsets.

In addition the named containers can readily be transformed to vanilla standards-compliant XHTML should one wish to do so for any reason.

3. Signature/Execution blocks

- photocopied examples of signatures we need to be able to represent have been circulated
- several possible models are in circulation for comment and review
- the signature model may accommodate an opening line before the signatures (possibly including a date); alternatively, that line could be regarded as outside the signatures element.

4. Appendices/Annexures

These are to be handled using our extras container (see above).

The extras container allows annexures to be grouped together, for example:

```
<extras class="annexures">
        <extra>
                <nr>Annexure 1</nr>
                <h>Pro forma NDA</h>
                <instrument>
                        <h>Nondisclosure Agreement</h>
                        :
                </instrument>
        </extra>
        <extra>
                <nr>Annexure 2</nr>
                        :
        </extra>
        :
</extras>
```

5. Annotations:

Handling of note/example as last child of <section>

- Can/should the Ruby module be used for this?

6.Extension mechanism

via foreign namespace elements? or use Div?

7. Cover page?

ISSUE: Does our structural XML need to be able to represent this explicitly, or is this purely the domain of some separate styling mechanism which can operate on our structural XML?

The question is whether this is required for an adequate "exchange" of XML data between organisations. It is noted that any document to be exchanged for the purposes of signature is likely to be PDF, Word, SVG etc, not eContracts XML. But if it is eContracts XML, it would need to be accompanied by everything necessary to style/render it in the manner intended by the sender.

8. Header/footer?

ISSUE: Does our structural XML need to be able to represent this explicitly, or is this purely the domain of some separate styling mechanism which can operate on our structural XML?

Why can't this be left purely to the styling mechanism?

- XSLT can handle it
- See CSS3 Paged Media Module (W3C Candidate Recommendation of 25 Feb): margin boxes

Even so, is it more properly the province of the author?

9. Page-set?

ISSUE: Does our structural XML need to be able to represent **page breaks** explicitly, or is this purely the domain of some separate styling mechanism which can operate on our structural XML?

Why can't this be left purely to the styling mechanism?

- XSLT can handle it
- See CSS3 Paged Media Module (W3C Candidate Recommendation of 25 Feb): page-break-before, page-break-after etc

E. Module packaging

1. Module use cases

Is the grammar designed to be free-standing, or to slot into an XHTML page, or both?

It is agreed that the <instrument> element could be the root of a standalone document (ie there is no “html”, “body” etc elements.

We will also look at packaging our work so that it can be used within an XHTML document. Note: in this context, we wouldn’t be able to maintain our pruned XHTML element re-definitions if the module we define is to be an XHTML “Family Module”.

2. Choice of grammar

Recommendations about which of W3C schema, DTD, Relax NG will be used etc

3. Conformance considerations

The issue of conformance is yet to be fully assessed.

Preliminary assessment is as follows:

- In the case where Instrument is the root element, we are not Host Language conformant. The reason for this is that an “XHTML Host Language” must include the Structure module (whereas an “Integration Set” need not)
- In any case, since we have pruned unnecessary elements, strictly speaking, we do not conform at either level.

If particular conformance objectives are identified and regarded as mandatory, we may need to revise our model to accommodate them.

F. Other possible outstanding issues

1. Forms module

Contract documents within the scope of the TC are not, and do not include, HTML forms which can be submitted to a web server.

In the view of Peter Meyer and Jason Harrop, there is no place for HTML forms nor for XForms in the structural markup per se.

2. Party references

See appendix 3 for possible markup.

3. Definitions and their usage

4. Symbols

Out of scope (ie Unicode), or mechanism to use Wingdings etc?

5. Content re-use mechanism

Consider after an initial report back to TC?

6. Metainformation

Out of scope of sub-committee

7. Formulae / Equations

8. Notary Certificate / Attorney Certificate

Out of scope or Liaise with Notary TC about these?

Appendix 1 – indicative DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE instrument [
    <!ENTITY % primaryattributes "class  NMTOKENS  #IMPLIED
   id   ID   #IMPLIED
">

    <!ENTITY % Inline " abbr| cite | dfn | em | I | quote | span | strong | sub | sup ">
    <!ENTITY % List " ol | inlinelist ">

    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                HIGH LEVEL STRUCTURE

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->


    <!ELEMENT instrument (h*, extra?, ( p* | (datearea, parties) ), background?,  operative,
(signatures | extras)* )>

    <!ELEMENT h (#PCDATA | %Inline;)*>

    <!-- For UK style -->
    <!ELEMENT datearea (h?, p*) >
    <!ELEMENT parties (h?, (party | p)* ) >
    <!ELEMENT party (#PCDATA | I)* >
    <!-- / For UK style -->

    <!ELEMENT background (nr?, h?, section*)>

    <!ELEMENT operative (nr?, h?, p*, section*)>

    <!ELEMENT extras (h?, extra*)>

    <!ATTLIST extras
        type (schedules | attachments | annexures | appendices | exhibits)  "schedules">

    <!ELEMENT extra (nr?, h?, p*, (section* | instrument*) )>
    <!ATTLIST extra %primaryattributes; >

    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                CLAUSE MODEL

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

    <!ELEMENT section (nr?, h?, p*, section*)>
    <!ATTLIST section
    %primaryattributes;
>
    <!ELEMENT nr (#PCDATA)>
    <!-- nr content model to be completed -->

    <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                GRAMMATICAL PARAGRAPH
```

```
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->


    <!ELEMENT p (#PCDATA | table | img | blockquote | %Inline; | %List;)*>

    <!ELEMENT ol (li* | ili*)>
    <!ELEMENT li (nr?, h?, p*)>
    <!ATTLIST li
    %primaryattributes;
>
    <!ELEMENT inlinelist (nr?, ili)*>
    <!-- just a placeholder

      For example, we may do

         <!ELEMENT ol   (li | ili)+>

      instead.

      Also, not clear how to do numbering yet.

   -->
    <!ELEMENT ili (#PCDATA | inlinelist)*>

    <!ELEMENT blockquote (p)*>
    <!-- blockquote to be fleshed out -->

    <!ELEMENT table EMPTY>
    <!-- placeholder -->
    <!ELEMENT img EMPTY>
    <!-- placeholder: quite possibly this will become <object>  -->

    <!-- Inline Module -->
    <!ELEMENT abbr (#PCDATA | %Inline;)*>
    <!ELEMENT cite (#PCDATA | %Inline;)*>
    <!ELEMENT dfn (#PCDATA | %Inline;)*>
    <!ELEMENT em (#PCDATA | %Inline;)*>
    <!ELEMENT l (#PCDATA | %Inline;)*>
    <!ELEMENT quote (#PCDATA | %Inline;)*>
    <!ELEMENT span (#PCDATA | %Inline;)*>
    <!ELEMENT strong (#PCDATA | %Inline;)*>
    <!ELEMENT sub (#PCDATA | %Inline;)*>
    <!ELEMENT sup (#PCDATA | %Inline;)*>

<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                SIGNATURES

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->

    <!ELEMENT signatures EMPTY>
    <!-- placeholder -->



]>
<instrument>
    <h>Skeleton Contract Template</h>
    <datearea>
```

```
        <p>THIS AGREEMENT dated 4 April 2004</p>
    </datearea>
    <parties>
        <!-- UK style markup -->
        <h> BETWEEN </h>
        <party>Fiona First LLC ("First Party")</party>
        <p> AND </p>
        <party>Simon Second LLC ("Second Party")</party>
    </parties>
    <background>
        <h>BACKGROUND</h>
        <section>
            <p>Simon would like to ...</p>
        </section>
        <section>
            <p>Fiona has agreed to ... according to the terms and conditions of this agreement.</p>
        </section>
    </background>
    <operative>
        <h>IT IS HEREBY AGREED:</h>
        <section>
            <nr>1.</nr>
            <h>First clause</h>
            <p>Clause Body. Clause Body. Clause Body. Clause Body. Clause Body. Clause Body.
Clause Body. Clause Body. Clause Body. Clause Body. Clause Body. </p>
        </section>
        <section>
            <nr>2.</nr>
            <h>Second clause</h>
            <p>Clause Body. Clause Body. Clause Body. Clause Body. Clause Body. Clause Body.
Clause Body. Clause Body. Clause Body. Clause Body. Clause Body. </p>
        </section>
    </operative>
    <signatures/>
        <!-- Signature model to be completed -->
    <extras type="schedules">
        <h>Schedules</h>
        <extra>
            <nr>Schedule 1 - </nr>
            <h>Key Personnel</h>
            <section>
                <p>The key personnel are:<ol>
                        <li>
                            <p>Fred</p>
                        </li>
                        <li>
                            <p>Mary</p>
                        </li>
                        <li>
                            <p>Jane</p>
                        </li>
                    </ol>
                </p>
            </section>
        </extra>
        <extra>
            <nr>Schedule 2 - </nr>
            <h>Proforma NDA</h>
            <instrument>
```

```
                <h>Non disclosure Agreement</h>
                <!-- etc -->
                <operative></operative>
            </instrument>
        </extra>
    </extras>
</instrument>
```

Appendix 2 – UK/ANZ date area and parties

Example 1

THIS CONTRACT is made on the day of 2004

BETWEEN

COMMONWEALTH OF AUSTRALIA as represented by the Department of Family and Community Services ("**FaCS**")

AND _____ ("**Contractor**") ACN ….. .

RECITALS

:

1 INTERPRETATION

Comments:

- *"THIS CONTRACT" is not in bold, and nor is "AND".*
- *No heading before operative clauses*

Example 2

DEED OF CHARGE made at Melbourne on 2004

BETWEEN **XXX** of [address] (the "**Company**") and YYY of [address] (the "**Trustee**") (each a "Chargor")

AND **BLAGH BLAGH FOR AND ON BEHALF OF THE CROWN IN THE RIGHT OF THE STATE OF VICTORIA ("Chargee")**

THIS DEED PROVIDES

1. DEFINITIONS AND INTERPRETATION

Comments:

- *Opening line contains semantic information*
  - *Type of agreement: "Deed of Charge"*
  - *Place*
  - *Date*
- *Multiple parties in the first party line*
- *Operative clauses has an opening/heading*

Example 3

AGREEMENT dated #

BETWEEN # ACN of # of [address] ("#")

AND # ACN of # of [address] ("#")

AND # ACN of # of [address] ("#")

RECITALS

:

AGREEMENT

1. DEFINITIONS

Comments:

- *Pretty simple*

Example 4

THIS AGREEMENT is made this day of 2004.

BETWEEN

ACN of # of [address] (the "#")

AND

ACN of # of [address] ("#")

RECITALS

:

The parties now wish to record the terms of their strategic alliance in this Agreement.

1. DEFINITIONS AND GENERAL INTERPRETATION

Comments:

- *BETWEEN/AND on separate lines*
- *Opening for operative clauses does not look like a heading*

Example 5

Deed of Charge

Date

Parties

1. # ACN of # of [address] ("#").
2. # ACN of # of [address] ("#").

Recitals

A

B

It is agreed as follows.

1. Definitions and interpretation

Comments:

- *Date/Parties/Recitals headings are different from "**It is agreed** as follows"*
- *Here, date and parties are separate sections*

Example 6

DATE OF THIS CONTRACT

This Contract is dated

PARTIES TO THIS CONTRACT

This Contract is between

THE UNIVERSITY OF SYDNEY ABN 15 211 513 464

(**University**)

and

Name:

ABN:

(**Contractor**)

Example 7

This **Deed** is made on

between	(1)	The Companies listed in...(**Companies**)
and	(2)	s1 and s2 (together, **Secured Party**)

Introduction

The Secured Party…

It is declared

1. Interpretation

1.1 Definitions

Comments:

- *The party lines have three distinct columns (whether done as a table or tabs)*
- *"introduction" and "it is declared" have the same visual appearance; Compare "This Deed is made on", which is in normal paragraph style.*

Example 8

THIS AGREEMENT is made on the day of 2001

BETWEEN:

(1) # ACN of # of [address] ("#").

(2) # ACN of # of [address] ("#")

AND:

(3) # ACN of # of [address] ("#")

RECITALS

:

OPERATIVE PROVISIONS

1. DEFINITIONS AND INTERPRETATION

Comments:

- *Note layout of BETWEEN and single AND*

Appendix 3 – Marked up UK/ANZ <party> element examples

In the examples in appendix 2, a party is typically identified with a paragraph of text which includes the party's:

- legal name
- address
- statutory registration number
- abbreviated name for use throughout the contract

More than one party may be indentified in a single paragraph. See examples 2 and 7.

The markup proposed in part D above represents such a paragraph with a <party> tag. For example:

> **<party>**the Governor and Company of the Bank of England, having its principal place of business at Threadneedle Street, London, EC2R 8AH (hereinafter "the Bank")**</party>**

Peter Meyer and Jason Harrop believe it would be useful to mark up the information which appears in a <party> element, for several reasons:

- so that during authoring, the authoring application would know about the parties already identified, making it easy to enter the name of a party while typing a clause. This will reduce errors, and make it easier to search/replace party names.
- for easy extraction of party information from an eContract.

There are two simple ways of doing this which would meet the identified need.

First Approach – Party centric

The party-centric approach explicitly identifies all information relating to a particular party.

<party>
<partydetail nmtoken="boe" type="legalname">the Governor and Company of the Bank of England**</partydetail>**, having its principal place of business at **<partydetail nmtoken ="boe" type="address">**Threadneedle Street, London, EC2R 8AH**</partydetail>**
(hereinafter "**<partydetail nmtoken ="boe" type="reference">**the Bank**</partydetail>**")
</party>

All information relating to a particular party shares the same @ **nmtoken** value.

Second Approach – Name/Address oriented

The name/address oriented approach gives weight to the fact that sometimes you may wish to mark up something as a name or an address, even though it does not relate to a party to the contract in a strict legal sense.

This markup uses <name> and <address> tags, not <partydetail> tags:

<party>
<name nmtoken ="boe" role="party" type="legalname">the Governor and Company of the Bank of England**</name >**, having its principal place of business at **<address nmtoken ="boe">**Threadneedle Street, London, EC2R 8AH**</address>** (hereinafter "**<name nmtoken ="boe" type="reference">**the Bank**</ name>**")
</party>

Again, all information relating to a particular party shares the same @ **nmtoken** value.