# Pruning XHTML2 - Roadmap

## eContracts Structure sub-committee

Editor: Jason Harrop, jharrop@speedlegal.com

Sub-committee members:
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: ~~27 March~~ 12 April 2004

---

**Summary of changes:** The structural teleconf of Wed 7 April focused on highest level content, ie how to represent the top level structure of a contract, which is often something like:

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

So most of the changes are in Part D "Highest Level Content Additions" and Part E "Module Packaging"

---

**Agenda for next meetings** (this will only make sense in the context of the rest of this document):

- finish of 7 April discussion:
  - element name for high level structure element ( or one we make up?)
  - starter list of values for @class
- signature blocks
- numbering, tables, styles ....

---

Note: Our discussions to date have been based on the XHTML 2.0 working draft of May 2003. We understand there will 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. The next WD is intended to be the "penultimate one". As of 9 April, this document is not yet available.

~~Based on our discussions with the W3C rep, our work is likely to be classified as Integration Set conformant, but not Host Language conformant (not because of any additions we may make, but because we have reduced the grammar in some areas)~~

## A. Block level considerations

### 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>
```

- #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 p* ? Can a table cell be numbered? If so, just once, or once per p?
- 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.

- do 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>
- do 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.

## B. Paragraph level considerations

### 1. <l> in <p>

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

Note:
- There is no <br/> 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>
```

### 2. Block lists

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

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

### 3. 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

**4. 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.

**5. Definitions and their usage**

**6. Cross references (in conjunction with numbering above)**

- do we need to be able to do cross references to images and/or tables?

**7. Party references (out of scope for now?)**

## 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.
- (Should it be #REQUIRED on a <section> and anything else to which one can cross reference? )

"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. **What else (if anything) do we need to represent top level contract structures (especially from a numbering/cross reference point of view).**

~~Depending on what we decide here and in "E. Module Packaging" below,~~

~~<!ELEMENT body (h?, section*)>~~

~~may or may not be appropriate.~~

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.

It was noted that these artifacts are "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

We agreed (subject to notes below) that in order to provide appropriate flexibility/extensibility, we would introduce a single recursive generic container, which has an attribute (eg @class) which indicates which of the above or other structures the container represents.

The container is recursive, so that:

- example 1: the container can be used for a schedule, and also to group all of the schedules in a single container
- example 2: potentially, the container could be used to provide a highest-level structure of front-body-back

TODO: decide what to call this element. Is it appropriate to use  for this, or something else. Until we have decided on a name, I am using <struct>.

The agreed content model is:

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

Notes:

- the container is optionally numbered (for example, in “Schedule 1 – Pro Forma NDA”, the <nr> is “Schedule 1”)
- the container has an optional heading (for example, “Pro Forma NDA”)
- there are optional opening paragraphs
- then you choose between sections, recursive struct, or instruments (as to which see below)

  - example 1: the container can be used for a schedule, and also to group all of the schedules in a single container
  - example 2: potentially, the container could be used to provide a highest-level structure of front-body-back
- TODO: decide what element name to use instead of struct. Is use of XHTML’s  appropriate?
- TODO: produce a list of recommended values for the @class (and confirm @class is appropriate for this purpose)

Issues:

- John McClure is keen to have a way of forcing page breaks in the structural markup. TO DO: John to document the processing model that this is helpful for. If we decide to include page break handling in the structural markup (potentially with a page-set element), John would like to revisit our decision to use a single generic container, since in his view several may be necessary.
- John’s suggestion for handling signature blocks involves using <l> for the signature lines, and  (or struct) for the signature block. So he’d like to see <l> added to the content model for struct, if his model for a signature block is chosen.

## 2. Signature/Execution blocks

[Jason to:

- send photocopies of examples we need to be able to represent
- ~~-~~write up and circulate ~~model~~his proposal.]

[John McClure has circulated a proposal by email title “Signature blocks” of 7 April]

~~Does anyone else have a model they'd like to propose?]~~

## 3. Appendices/Annexures

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

The container can be nested, so that annexures are grouped together, for example:

```
<struct class=”Annexures”>
    <struct class=”Annexure”>
        <nr>Annexure 1</nr>
        <h>Pro forma NDA</h>
        <instrument>
```

```
                    <h>Nondisclosure Agreement</h>
                    :
                </instrument>
            </struct>
            <struct class="Annexure">
                <nr>Annexure 2</nr>
                :
            </struct>
            :
        </struct>
```

4. **Annotations:**

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

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

5. **Allow extensions via foreign namespace elements?**

6. **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.

7. **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

8. **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. Root element

~~Are we defining a grammar with root element <html> or something else (eg <legal:instrument>)?~~
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”.

The question has been raised as to whether/how other LegalXML TCs will be able to use this document type.

Feedback is to be solicited from the wider LegalXML community as to whether this choice is appropriate.

The agreed content model:

instrument (h*, struct*)

Notes:

- h*, so that it is possible to have a heading and a sub-heading
- the element name for struct is yet to be decided (see Part D above)
- we considered: instrument (h*, p*, section*, struct*), but decided it was better not to allow p* or section* directly within <instrument>

Issues:

- this content model is subject to resolution of the issues John has raised about page break handling (see Part D above), and also cover-page

### 2. Module use cases

If something else, 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”.

### 3. Choice of grammar

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

### 4. 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. Later

1. **Party references**

   Defered, since party handling is possibly part of semantic layer?

2. **Symbols**

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

3. **Content re-use mechanism**

   Consider after an initial report back to TC?

4. **Metainformation**

   Out of scope of sub-committee?

5. **Formulae / Equations**

6. **Notary Certificate / Attorney Certificate**

   Liaise with Notary TC about these.

## Appendix – indicative DTD

```
<!DOCTYPE html instrument [
<!ENTITY % primaryattributes "class  NMTOKENS  #IMPLIED
   id   ID   #IMPLIED
   style  CDATA  #IMPLIED
   dir (ltr|rtl|lro|rlo) 'ltr'
">
<!-- @property probably to be added;
    @style may go.

    TODO: do we want to attach these to all inline elements as well?
   -->
<!ENTITY % hypertext "href   CDATA  #IMPLIED
   cite  CDATA  #IMPLIED
   target CDATA; #IMPLIED
   rel  CDATA #IMPLIED
   rev  CDATA #IMPLIED
   accesskey CDATA #IMPLIED
   navindex CDATA #IMPLIED
   xml:base CDATA  #IMPLIED">
<!-- nb, i haven't attached %hypertext; anywhere, and i've temporarily changed its attribute definitions -->
<!ENTITY % Inline " abbr| cite | dfn | em | quote | span | strong | sub | sup ">
<!-- TODO: revisit suitability of cite and dfn -->
<!ENTITY % List " ol | ilist ">


<!ELEMENT instrument (h*, struct*)>

<!ELEMENT struct (nr?, h?, p*, (section* | struct* | instrument*))> <!-- name of this element to be discussed -->

<!ELEMENT ol (li)+>

<!ELEMENT li (nr?, h?, p*)>
<!ATTLIST li %primaryattributes; >

<!ELEMENT ilist (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 | ilist)*>

<!-- Structure Module -->
<!ELEMENT html (head, body)>
<!ELEMENT head (title)>
<!ELEMENT title (#PCDATA | %Inline;)*>
<!ELEMENT body (h?, section*)>
<!-- to be discussed -->
<!-- Block stuff -->
<!ELEMENT section (nr?, h?, p*, section*)>
<!ATTLIST section %primaryattributes; >


<!ELEMENT nr (#PCDATA)>
<!-- nr content model to be completed -->
<!ELEMENT h (#PCDATA | %Inline;)*>
<!ELEMENT p (#PCDATA | table | img | blockquote  | %Inline; | %List;)*>
<!ELEMENT blockquote (p)*>
<!-- blockquote to be fleshed out -->
```

```
<!ELEMENT div EMPTY>
<!-- div to be fleshed out and included in other content models as appropriate -->
<!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 quote (#PCDATA | %Inline;)*>
<!ELEMENT span (#PCDATA | %Inline;)*>
<!ELEMENT strong (#PCDATA | %Inline;)*>
<!ELEMENT sub (#PCDATA | %Inline;)*>
<!ELEMENT sup (#PCDATA | %Inline;)*>

]>
```