

Sectional Units in GELLMU

May 24, 2006 (still at draft stage)

Table of Contents

1	Introduction.....	1
2	Labeling and Referencing.....	2
3	Anchoring and Long Section Titles	3
Sec. 4	Fourth.....	4
Sec. 5	Another section.....	4
6	Sequencing.....	5
Division A	Special Sectional Units.....	6
A.1	First Subdivision.....	6
A.2	Second Subdivision.....	6
7	A Non-Special Section.....	6
8	General Usage for Sectional Units	7
8.1	The content model.....	7
8.2	The L ^A T _E X-Like Form of General Usage	7

1 Introduction

In classical L^AT_EX one writes simply

```
\section{Some title for a section}
```

to begin a new section. While this markup specifically delineates the section title, L^AT_EX understands that a section has begun. The *section* command has an option whose content is an alternate title for the table of contents, and the starred form of the command suppresses an otherwise automatic section number.

With SGML the classical approach is something along the following lines:

```
<section><sectiontitle>Some title for a section</sectiontitle>  
<para> A paragraph.  
<para> Another paragraph.  
...
```

</section>

Basic GELLMU markup corresponding to this would be:¹

```
\begin{section}\sectiontitle{Some title for a section}
\para A paragraph.
\para Another paragraph.
. . .
\end{Section}
```

If, moreover, the markup is to be conforming XML markup, then each *para* tag would need explicit closure with </para>.

The main point here is that the open and close tags for a section in a classical SGML document type encompass the whole contents of a section, and separate markup is required for the section title.²

The didactic SGML document type under *regular* GELLMU seeks to model classical L^AT_EX as closely as possible in order to provide a bridge for authors from L^AT_EX to SGML (and, indeed, XML). For this reason a command *section* similar to the classical L^AT_EX command that delineates a section title is provided in the didactic SGML document type. At the same time this document type has a whole section container *Section* (upper case *S*) that in the simplest case consists of a *shead* container for its title (or header) followed by any number of paragraphs. The XML form of the didactic document type supports only this latter tag, which means that the translation script that converts SGML to XML has an unambiguous way of performing the conversion from *section* to *Section* provided that the author's source leads to an SGML instance which is valid³ under the didactic document type.

2 Labeling and Referencing

The next sections demonstrate various ways of making sections. They also demonstrate how sections interact with labels, references, and the GELLMU table of contents.⁴

The markup for the first section of this document was simply the classical L^AT_EX-like

```
\section{Introduction} .
```

There are various combinations of sectional markup that may be used.

¹Or one could use:

```
\Section{\sectiontitle{Some...}
\para A para... . . .}
```

instead of using the environment-like *begin/end* construction.

²In fact, classical SGML document types are often even more elaborate than this.

³Caveats:

No document type definition under SGML is actually a complete language definition. A document type definition describes a document markup structurally; in particular, it does not provide definition for legal “field” values.

While the GELLMU syntactic translator is now considered as “alpha” software, the document type and the accompanying translators, which constitute the didactic production system are developmental. These materials have some support for obsolete practice and also contain sketch sections that are not fully robust.

⁴The handling of sectional unit numbers, labels, references, and tables of contents is built into GELLMU at the point of XML generation in order to provide consistency across all ultimate output formats where such consistency is possible.

The markup for the present section is:

```
\section{\shhead{Labeling and Referencing}} .
```

In the didactic GELLMU production system this is equivalent at the point of XML generation to the more L^AT_EX-like markup

```
\section{ ... } .
```

This sentence contains a label, marked up with the usage `\label{first}`, that is invisible and not part of the section header. While it is the first author-provided label in this document and the first label in the source file, it is not the first label in the XML version of the document since the translator from SGML to XML in the didactic production system writes a label automatically in every *shhead*. At any “inline” location in the document the label may be referenced by using the markup `\ref{first}`, and with this simple usage its value is 2, i.e., the identifier of the sectional unit in which the label is located.

The didactic production system also provides “visible key” label and reference commands *klabel* and *kref*. For example, a *kref* reference to the last section is `[last]`, which is provided with the markup `[\kref{last}]`. In a print-only format there might be little difference between writing “`\kref{last}`” and simply writing the key “last”. But in a hypertext format *kref* may be used to generate an anchor.

3 Anchor References and the Handling of Section Titles when the Section Title is Long Enough to Require a Linebreak Under Most Formattings

This section is opened with the L^AT_EX-like usage:

```
\section[Anchoring and Long Section Titles]{\label{longtitle}Anchor  
References and the Handling of Section Titles when the Section Title  
is Long Enough to Require a Linebreak Under Most Formattings}
```

The purpose of the option string, as with L^AT_EX is to provide a shorter section title for listing in the table of contents.

Here is an ordinary *ref* reference back to section 2. This sentence contains an anchor (with *anch*) back to section 2. If, as is the case here, the anchor uses an *iref* rather than an *href*, with the markup

```
\anch[iref="first"]{section \ref{first}} ,
```

then the anchor might be unnoticeable in a print-only formatting while “active” in an online formatting. This anchor⁵, using the markup

```
\anch[href="\#first"]{anchor} ,
```

⁵Reference to internal target with key: first

which is for general web references, might produce a footnote in a print-only formatting.

The option of an anchor, which is not an attribute option but rather becomes the element *anchref* in XML (while the anchor's argument — its “presented content” — becomes *anchv*), is expected to contain whitespace separated strings of the form `name="value"`⁶ where name is one of the following:

fref A footnote reference. Value is a string that becomes the content of an automatically created footnote to the text in *anchv*. This usage is deprecated; use `\footnote` instead.

href A web reference as with *href* in HTML. That is, value is a URI. It *could be* of the form `"#labelkey"` where `"labelkey"` is the name of a label key that could just as well be used with *iref*. For this the `#` needs to be escaped, i.e., marked up as `"\#"`. In a non-hypertext formatting the URI may be presented as a note or footnote associated with the text in *anchv*.

Href Same as *href* except that the author wishes to suppress any note or footnote presentation of the URI. For example, the URI might be obviously deducible from the *anchv* content.

iref An internal reference; value must be a label key arising from *label* or *klabel*.

Note also that there is a command *urlanch* (probably should have been *urianch*), taking a single argument, used for URIs, which is intended to have the same effect as a *newcommand* with one argument for creating a web anchor with the URI as presented content.

Sec. 4 A Fourth Section

This section is introduced with the markup

```
\begin{Section}[Fourth][Sec.\ ]{\label{begin1}A Fourth Section} .
```

(this time an upper case “S”) followed at its terminus by

```
\end{Section} .
```

It makes use of the first two of the three options for *Section* (and for *Subsection*, ..., *section*, *subsection*, ...). The options, in order, are *sopt*, *srefix*, and *sunit*. They are explained in section 8 below.

Sec. 5 Another Section

For this section the markup style of the previous section is used except that the options are replaced by commands bearing their explicit names. The section opening construction is

```
\begin{Section}\sopt{Another section}  
\srefix{Sec.\ }\thead{\label{begin2}Another Section}
```

followed at its terminus by `\end{Section}`.

⁶The value strings may contain simple markup such as, for example, `"\tld;"` to provide robust multiple output processing of `'` whereas an attribute option may not contain markup.

6 Sequencing

Although one *could* provide SGML modeling for L^AT_EX's counters, it would not be very much along the lines of main track XML document types.

Sequencing may be handled under the GELLMU didactic *artice* document type using labels and references. Toward this end one makes use of three SGML attributes that are provided with the *label* tag:

series The value is the name of a sequenced family of labels. A label may belong to at most one family, but there may be multiple labels at the same location. There is no default value of series.

serseq The value is the sequence number of the label in its series if a series is defined. It is meaningless if no series is specified for the label.

refkey The name of a label key from which to spawn an automatically generated value for the attribute *serseq* of the current label.

Every label has a reference value that is normally accessed with the *ref* command. This results in the creation, when the XML version is generated, of an XML entity reference with name based on the reference's *key* argument, that matches a CDATA entity definition at the top of the XML document. The use of indirection provided by this entity technique means that it is immaterial whether the reference is forward or backward.

The *evalref* command gives access to the literal value of a reference without indirection, and places that value as a literal in the XML version of an article. Thus, *evalref* is the name of a tag only in the SGML version. An *evalref* invocation will be successful only with a backward reference. This is extremely useful for managing non-default numbering of sectional units. For ordinary label references its use is undesirable even though it is possible for backward references.

The reference value of a label is determined as follows:

1. Basic reference values are positive integers. Upper and lower case alphabetic values and upper and lower case roman numeral values may be obtained by applying the *series* command (not to be confused with the *series* attribute for the *label* command) with the *type* attribute set to one of 'A', 'a', 'I', or 'i' to a basic reference value.
2. If the label is assigned to a label series and is given a *refkey* attribute, then the reference value of the label is the reference value of the label referenced by the key that is the value of the *refkey*. (This mechanism is used to re-start the sequencing of the sectional unit id's at the end of this document.)
3. Else if the label is assigned to a label series and the author supplies an explicit **literal** numeric value for the *serseq* attribute, then the value of *serseq* is its referenced value. (Markup — in particular, *evalref* — cannot be used in defining an attribute value.)
4. Else if the label is assigned to a label series, the reference value of the label is the next (positive integer) value for a label in that series. (This mechanism is used to control the sectional id of the last section of this document. It may also be used to run parallel interleaved sequences of sectional units, such as, for example, questions and answers, within a document.)

5. Else the reference value of the label is the sectional unit id of the smallest sectional unit containing the label.

Here is where a label, which is invisible, is planted with the markup “`\label{conmark}`” so as to record the sectional unit id for the current section. It evaluates to the sectional unit value, in this case 6. If that label were placed in a series, it would pick up as value the next value in the sequence associated with the series. So a second label is planted with the markup

```
\label[:series="nsec" refkey="conmark"]{}7
```

to prime a new label series with the value (6) of the current sectional unit id for later use in continuing the current sequence of sectional unit id's.

Division A Special Sectional Units

This sectional unit, which is parallel to top-level sections, is specially sequenced by “hand”. The markup for opening this unit is:

```
\section[] [Division ] [A]{Special ... }.
```

In *regular* GELLMU a L^AT_EX-like first option in which the first character is ‘:’ is a sequence of SGML attribute specifications. It is important to realize that attributes cannot contain markup. In *regular* GELLMU, where one is allowing characters such as ‘-’, ‘/’, ‘#’, and ‘=’ to serve as markup (because of their classical *overloading*⁸), only alpha-numeric characters are safe in attribute values when entered this way. For the most common situations, such as, for example, what might be the *href* attribute of *anch*⁹, a L^AT_EX-like option can be provided. L^AT_EX-like options handle what might be regarded as plain inline markup.

One could also use for this a very similar syntax with *Section*, i.e., “`\begin{Section} ... \end{Section}`”.

Make note of the secret phrase: valid markup¹⁰.

A.1 First Subdivision

A.2 Second Subdivision

7 A Non-Special Section

To avoid having to feed section numbers manually in the following or else spawning them from a label series, the opening of this section manually sets the *section* unit id.

⁷A key will be automatically assigned.

⁸This simply means that the actual meaning of the character depends on its context.

⁹If one really needs an arbitrary CDATA attributes for, say, *anch*, this can be done using the SGML internal declaration subset.

¹⁰Reference target for key: ”indiv”

8 General Usage for Sectional Units

This describes the content model in the didactic GELLMU document type for the “whole” sectional units *Section*, *Subsection*, ... as fully tagged.

8.1 The content model

The content model for sectional units is:

$$((\text{sopt})?, (\text{sprefix})?, (\text{sunit})?, (\text{shead}), (\%UnitContent)*)$$

where:

%UnitContent refers to the subsections, loose paragraphs, and other general content that is allowed inside a section.

shead is the required¹¹ section header or title.

sopt is an optional title for use in the table of contents¹².

sprefix is optional markup for text that is to precede the sectional unit sequence notation. For example, if the sectional unit sequence is “B” and *sprefix* is the markup string “**Appendix** ” (ending with a blank space), then the visible indicator for the sectional unit, when consistent with the setting of *secnumdepth*, is “Appendix B” both at the beginning of the section and in the table of contents. Or if the sequence is “3” and the *sprefix* is “A”, then the visible indicator is “A3”.

sunit is an optional setting for the sectional unit sequence. The GELLMU didactic document type has an attribute “sid” for the sectional units *Section*, *Subsection*, ... that is optional (and rare for author usage) in the SGML version but required in the XML version. The translator from SGML to XML computes it in the standard way. For example subsection 1 of subsection 3 in section 2 acquires the *sid* “2.3.1”. It is expected that formatters will use this value for the visible sectional unit indicator, preceded, as previously described, by any *sprefix*, unless the user provides *sunit*. While *sunit* is intended to override the visible indicator, it is not provided to override the *sid* attribute itself which a formatter should see as describing logical structure.

8.2 The L^AT_EX-Like Form of General Usage

Corresponding L^AT_EX-like *argument/option* syntax can be used, as previously indicated, in GELLMU source with *section*, *subsection*, If, however, *argument/option* syntax is used, one must be mindful of the ordering of the options, and use empty option brackets, as necessary, to indicate the position in the sequence of an option with content. For example, a sole option is understood as *sopt*, the version of the sectional unit title to be used in the table of contents.

¹¹It may be left empty, but it must be present.

¹²The presence of *sopt* does not cause a table of contents to be produced automatically. For that one uses “\tableofcontents”. Moreover, the presence of *sopt* should have no effect upon a manually constructed “\TableOfContents”.

To provide only *srefix* with *argument/option* syntax one precedes the bracket sequence for *srefix* with an empty pair "`[]`" of option brackets¹³ for *sopt*.

¹³The didactic GELLMU production system does not offer a way to furnish a formally empty string for actual use with the table of contents. That is, the markup "`\sopt{}`" furnishes an empty string which, in turn, signals "no *sopt*". For most purposes a visibly empty contents line may be achieved by inserting a blank space as the alternate title with the markup "`\sopt{\ }`". Beyond that the document type should be extended with the provision of an empty element *emptystring* to indicate the empty string.