IN: Introduction to CDL Text Data Conventions

Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

18 June 2004

IN.1 Background

As part of the work on creating a large-scale knowledge-base--the Cuneiform Digital Library (CDL)--for Assyriologists, the CDLI, PSD and DCCLT projects have agreed to utilize a single unified set of document types and data structures for the acquisition and archival storage of all kinds of documentation from the entire geographical and chronological extent of cuneiform culture.

This document gives a brief introduction to the most important data types by way of general orientation within the system. Other documents provide more detailed information in the form of tutorials, user manuals and quick reference sheets covering the conventions utilized in the Cuneiform Digital Library.

IN.2 Transliterations

Transliterations are the primary data type for the CDL. By definition, a transliteration is a rendering into roman characters of a specific object or group of related objects (such as a tablet and its envelope).

Transliterations are acquired in the ASCII Text Format (occasionally known as the ASCII Transliteration Format) or ATF; this format is fully defined in this series of documents. Transliterations are the default data type for ATF files and are organized according to the physical structure of the objects and their subdivisions such as columns. (Links to Text and Structure Conventions)

Transliterations are converted by a program to a rigorously defined XML format known as XTF, or XML Transliteration Format. This is the main archival format as well as the form of the text on which further processing, such as parsing and semantic tagging, is
Composite texts are a common data type in Assyriology and are a supported data type for the CDL. They may be entered using a variant of the ATF format which is specially flagged as a composite text in the input.

By definition, composite texts consist of a sequence of lines which represent a reconstruction of the textual contents of one or more exemplars, or actual objects. Composite texts may also give the texts of the exemplars on which the reconstructed line is based. (Links to Miscellaneous Document Type and Linking Conventions)

The synopticon is a variant of the composite text which lacks a composite line: it simply gives the text of a collection of exemplars which either give the same line or share some less reconstructionist relationship.

Synoptica are entered as ATF files, like composites, but have some special rules which are defined elsewhere in this documentation. Besides their occasional use as a data-entry device (where it may be useful to type up several similar tablets in the same file to be separated programmatically later), synoptica are the natural representation of collections of parallel lines which have been defined in the data set as links. (Links to Miscellaneous Document Type and Linking Conventions)

Matrices are a very efficient way of entering and manipulating multiple exemplars where the degree of textual variation is relatively small, such as the Nippur literary corpus. Matrices are not yet handled by the CDL system, though most of the software to manipulate them does exist. They are likely to be fully supported by mid-2004.

Tabulations are an efficient way to enter large numbers of links between different data
objects (transliterations, composites, etc.) by simply giving a table of ranges of lines in different documents which should be considered equivalent, for example, the lines of editions of the same composite text prepared by different scholars. Tabulations are not yet implemented. (Links to Miscellaneous Document Type and Linking Conventions)
QR: Quick Reference for ATF Format

Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

30 June 2004

QR.1 Structure Conventions

<table>
<thead>
<tr>
<th>@&lt;type&gt;</th>
<th>Type = @object (default); @composite; @matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Text/catalogue number designator</td>
</tr>
<tr>
<td>@&lt;object&gt;</td>
<td>Object = @tablet, @envelope, @prism, @object &lt;object-type&gt;</td>
</tr>
<tr>
<td>@&lt;surface&gt;</td>
<td>Surface = @obverse, @reverse, @surface, @edge, @left,</td>
</tr>
<tr>
<td></td>
<td>@right, @top, @bottom, @face [a..z]</td>
</tr>
<tr>
<td>@seal &lt;n&gt;</td>
<td>for transliteration of seal impressions on objects; n=number</td>
</tr>
<tr>
<td>@column</td>
<td>column</td>
</tr>
<tr>
<td>&lt;number&gt;.</td>
<td>line of text</td>
</tr>
<tr>
<td>&lt;number&gt;.&lt;subcases&gt;.</td>
<td>line of text with subdivision into cases</td>
</tr>
<tr>
<td>$</td>
<td>non-text, such as breakage, blank lines, etc. Specific keywords must be included in the non-text comment. These keywords are given in ST.8.</td>
</tr>
<tr>
<td>#</td>
<td>comment line</td>
</tr>
<tr>
<td>=:</td>
<td>multiplexing comment line giving original order of interpreted/reordered signs in preceding line of transliteration.</td>
</tr>
<tr>
<td>&lt;whitespace&gt;</td>
<td>continues previous line (modern convenience, not ancient line break)</td>
</tr>
</tbody>
</table>

QR.2 Inline ASCII Conventions

<table>
<thead>
<tr>
<th>[A-Z][a-z][0-9]</th>
<th>grapheme name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;hyphen&gt;</td>
<td>joiner for graphemes of single word</td>
</tr>
<tr>
<td>&lt;space&gt;</td>
<td>word separator</td>
</tr>
<tr>
<td>!</td>
<td>flags editor's correction of sign</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>!!</td>
<td>flags unusual sign-form</td>
</tr>
<tr>
<td>!!!</td>
<td>flags sign which is really on text despite expectation</td>
</tr>
<tr>
<td>?</td>
<td>flags uncertainty of identification or reading</td>
</tr>
<tr>
<td>*</td>
<td>flags collation</td>
</tr>
<tr>
<td>#</td>
<td>flags damage to sign</td>
</tr>
<tr>
<td>[...]</td>
<td>encloses material broken away from object</td>
</tr>
<tr>
<td>[(...)]</td>
<td>encloses material perhaps broken away from object</td>
</tr>
<tr>
<td>value(SIGN)</td>
<td>explanatory name or variant form after value</td>
</tr>
<tr>
<td>value!(SIGN)</td>
<td>actual signs on object given after corrected version</td>
</tr>
<tr>
<td>&lt;...&gt;</td>
<td>accidental omission supplied by editor</td>
</tr>
<tr>
<td>&lt;(...) &gt;</td>
<td>intentional omission supplied by editor</td>
</tr>
<tr>
<td>&lt;&lt;...&gt;&gt;</td>
<td>material removed by editor</td>
</tr>
<tr>
<td>{...}</td>
<td>single-sign gloss/determinative delimiters (written in normal script)</td>
</tr>
<tr>
<td>{...}{}</td>
<td>multi-sign or multi-word gloss delimiters (written in smaller script)</td>
</tr>
<tr>
<td>~</td>
<td>following sign is a logogram</td>
</tr>
<tr>
<td>x</td>
<td>unclear sign</td>
</tr>
<tr>
<td>X</td>
<td>clear sign not yet identified</td>
</tr>
<tr>
<td>{...}</td>
<td>compound grapheme delimiters</td>
</tr>
<tr>
<td>[.x%&amp;+()]</td>
<td>compound grapheme operators (see QR.3 below)</td>
</tr>
<tr>
<td>=[aeiu],^[aeiu]</td>
<td>long vowels in normalized Akkadian</td>
</tr>
<tr>
<td>%[sahrux]</td>
<td>language shift</td>
</tr>
<tr>
<td>%eg,%es,%ugn</td>
<td>register/writing system shift</td>
</tr>
<tr>
<td><em>...</em></td>
<td>encloses material is in alternate language</td>
</tr>
</tbody>
</table>

**QR.3 Compound Grapheme Conventions**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>juxtaposed signs, e.g., DU.DU</td>
</tr>
<tr>
<td>x</td>
<td>following sign(s) written over/within preceding sign, e.g., GA2xAN</td>
</tr>
<tr>
<td>%</td>
<td>signs are written crossed over each other (gilimmu), e.g., GI%GI</td>
</tr>
<tr>
<td>&amp;</td>
<td>signs are written one above the other, e.g., DU&amp;DU as opposed to standard DU</td>
</tr>
<tr>
<td>@</td>
<td>signs are written opposing, e.g., LU@LU</td>
</tr>
<tr>
<td>+</td>
<td>signs are ligatured, e.g., LAGAB+LAGAB (nigin2)</td>
</tr>
<tr>
<td>x4</td>
<td>signs are written in a square, e.g., LUx4</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>(...)</td>
<td>grouping of signs, e.g., GA2x(ME.EN)</td>
</tr>
<tr>
<td>@90</td>
<td>preceding sign is rotated 90 degrees clockwise</td>
</tr>
<tr>
<td>@180</td>
<td>preceding sign is rotated 180 degrees clockwise (inversum)</td>
</tr>
<tr>
<td>@270</td>
<td>preceding sign is rotated 270 degrees clockwise</td>
</tr>
<tr>
<td>@g</td>
<td>gunu, e.g., DU@g as opposed to standard DU</td>
</tr>
<tr>
<td>@t</td>
<td>tenu, e.g., GAN2@t</td>
</tr>
<tr>
<td>@s</td>
<td>sheshig, e.g., DU@s</td>
</tr>
<tr>
<td>@n</td>
<td>nutillu, e.g., SAG@n as opposed to the standard SAG</td>
</tr>
<tr>
<td>@k</td>
<td>kabatenu, e.g., ASZ@k as opposed to the standard ASZ</td>
</tr>
<tr>
<td>@z</td>
<td>zidatenu, e.g., ASZ@z</td>
</tr>
<tr>
<td>@c</td>
<td>curved, e.g., ASZ@c</td>
</tr>
<tr>
<td>@r</td>
<td>vertically reflected, e.g., U@r</td>
</tr>
<tr>
<td>@h</td>
<td>horizontally reflected, e.g., N07A@h (aka N07B) as opposed to the standard N07A</td>
</tr>
<tr>
<td>@v</td>
<td>variant, e.g., 4(ban2)@v as opposed to the standard 4(ban2)</td>
</tr>
</tbody>
</table>
ST: Structural Conventions for Cuneiform Text Editions

Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

30 June 2004

ST.1 Background

This document covers text structure conventions and is based on those developed for the CDLI project. We frequently refer to this format in the text below as the ATF format, i.e., the ASCII Text Format.

The basic structure of an ATF document consists of some or all of the following components in this order:

- atf comments (if any),
- the document type (default is @transliteration),
- the document name <&-line> (required),
- the object type (default is @tablet), and finally
- surface designations, line numbered transliteration, and non-transliterated material as described below (see example).

ST.2 Document types

The ATF format supports several document types; the document type label is given on a line of its own, beginning with an @-sign and must occur before the Text naming line (the one that starts with '&').

The following Document types are currently supported:

- @transliteration: This is the default type, for use with tablets and other objects.
- @composite: Composite text.
- @matrix: Matrix.
- @synopticon: Synoptic text.
- @tabulation: Tabulation.
To set a default language other than Sumerian for the entire text, the `#atf lang` command should be given after the document type and before the text name. For more information see the language shift documentation.

**ST.3 Text naming**

The text name should be given after an `&`; the preferred method is to give the record ID number after the ampersand, followed by a human-readable name given after an `=` sign, e.g.:  

- `&P100001 = AAS 1`

Record ID numbers are assigned by CDL. For individual tablets and other objects, the record ID number consists of a P, followed by 6 digits. For composite texts the ID consists of a Q, followed by 6 digits. A matrix ID number consists of an R, followed by 6 digits. A synoptic text will have a record ID number beginning with S, followed by 6 digits.

The following is a list of transliteration types and their record ID letters:

- transliteration (P)
- composite (Q)
- matrix (R)
- synopticon (S)
- tabulation (T)

**ST.4 Special protocols and definitions**

In cases where a linked source is to be defined or another protocol invoked, the line or lines preceding the text name line should contain this information, e.g.:  

- `#atf use lexical`

Information on the special protocols can be found in the documentation for ATF comments.

**ST.5 Block Structure**

ST.5.i Object types
After any special protocols or definitions and the text name, the next level of structural elements is that of the object type. This pertains only to transliterations (P). Object types are given as follows:

- @tablet
- @envelope
- @prism
- @object <object-type>

Suggested object types include 'seal', 'bulla', 'tag', 'brick', but there are in fact no constraints on what the name of an object-type should be, so it is possible to give transliterations of texts on any kind of object using this system. The CDLI text catalogue database contains a field for object type as well. There should be no whitespace between the '@' and the object type.

Transliteration of an envelope that has the same identifier as its tablet may be included with the tablet transliteration (see example).

**ST.5.ii Surfaces**

Within an object, the next level of structure is that of the surface:

- @surface
- @obverse
- @reverse
- @edge
- @left
- @right
- @top
- @bottom
- @face a..z
- seal <n>

The surfaces @left, @right, @top, and @bottom imply 'edge'. The surface @face is for use only with prisms. Each face should be identified with a single letter between a and z.

The generic surface @surface allows non-standard surfaces to be used, e.g.,
@surface shoulder.

The generic surface @edge is useful when the object is too badly broken to know which edge is preserved. For obverse and reverse uncertainty use @obverse? and @reverse?.

Surface designations are not required for objects with only one surface, such as seals, or odd ones, such as nails/nail heads.

Again, there should be no whitespace between the '@' and the surface type.

ST.5.iii Notes on Seals and Sealings

There are ambiguities in existing practices concerning the representation of texts which occur on seals. Seals as physical objects, seal inscriptions as composites pieced together from several instances, and transliterations of individual instances of a seal impression are all supported by the ATF format.

To transliterate a seal, i.e., a physical object, give the transliteration an '&-line' like any other object, and use the @object seal directive instead of @tablet.

To transliterate a sealing's inscription as a composite text (which is what most people do most of the time), use the @seal directive after the transliteration of the object on which the sealing occurs. Give the seal a simple integer number (e.g., @seal 1).

To transliterate a sealing's inscription as an instance use the @seal directive after the transliteration of the object on which the sealing occurs (just as with a composite transliteration of a sealing). Give the seal impression a number identifier for the composite sealing (as above), followed by a letter to identify the instance; use a different letter for each instance you wish to transliterate (e.g., @seal 1a, @seal 1b, etc.).

Internally @seal is handled as a surface, so it can naturally be divided up into @columns if required.

The numbers (1, 1a, etc.) given in transliteration containing the @seal directive can be
used in non-transliteration (see ST.8) to indicate which seal is rolled at a particular location.

Thus, for a tablet transliteration with a transliterated seal impression (@seal 1)

one can write simply

$ seal 1

to indicate that a rolling of seal 1 occurs at this location on the object.

Examples:
- **Format of transliteration of a cylinder seal**

&Pxxxxxx = cylinder seal name

@object seal
@column 1
1.
2.
3.
@column 2
1.
2.

- **Format of transliteration of a tablet with a seal impression**

&Pxxxxxx = tablet name

@tablet
@obverse
1.
2.
3.
$seal 1
@reverse
1.
2.
@seal 1
@column 1
1.
2.
3.
@column 2
1.
2.

ST.5.iv Surface Subdivision

Each surface contains one or more columns; it is not necessary to give a @column line if the surface has only one column, i.e. is not divided into multiple columns, as the conversion software takes care of this detail automatically.

At this level four kinds of line may be given:
• @column 1...n
• '$' <non-transliteration material>
• <line-number>. = <transliteration material>
• <whitespace> = <continuation of previous line>

The @column line indicates the start of a new column; note that the column number is given in arabic numbers after a space.

The '$'-line is used to note columns and lines which are broken, blank and the like; see the section on non-transliteration material below for more information.

The <line-number> is given as a decimal number followed by a period (dot) followed by at least one space. A line with subdivision into cases is given as a decimal number followed by a period (dot) followed by a letter followed by another period (e.g., 1.a.). In instances of line numbering beginning after a break, primes and double primes may follow the decimal number (e.g., 1').

In a transliteration, the atf2xtf processor automatically generates a new set of line numbers to ensure consistency within the CDL system. To override this default, the ATF comment

#atf use mylines

may be used, but this should be a rare occurrence. For all other text types, such as a
composite, the default behavior is to keep the line numbers used in the ATF file.

A line beginning with whitespace (spaces and/or tabs) is taken as a continuation line. This makes it comfortable to split up long lines if necessary for editing convenience. In general, white space at the beginning of a line should not be used to indicate the format of the original text such as an indented line on a tablet. See below for how to indicate alignment.

**ST.5.v Colophons and Other Text Divisions**

Logical divisions are indicated in ATF by the use of the @div tag. The @div tag requires a keyword, e.g., colophon, and a closing @end tag, which must take as its single argument the keyword of its corresponding opening @div. @div's of different kinds may not be interwoven.

```atf
@div part
...
@end part
@div colophon
...
@end colophon
```

In the liturgical corpus (including ETCSL editions of texts which could reasonably be considered liturgical), kirugu and other rubrics are used as logical structures, and they contain subdivisions giving the actual rubric; this is supported with the following syntax:

```atf
@div kirugu 1
  1. tur3-ra-na ...

@div rubric kirugu
  10. ki-ru-gu2 1-a-kam
  @end rubric

@end kirugu

@div giszgigal 1
  11. u2-a a-u3-a u2-a-u2-a
```
ST.6 Transliteration Material

The normal case is that transliterated material is simply a line of text given according to the ASCII transliteration conventions described here.

ST.7 Alignment in Transliterations

Three ways of aligning transliterations are supported. The preferred method is to indicate structural subdivisions (columns and fields) as follows.

ST.7.i Fields

The comma character (,) has two meanings in ATF texts. It is used to compose the ASCII sequences for emphatics (as in ʂ, for sadhe) and also to indicate a field division. Spaces are required around a field division unless the comma is followed directly by a field type notation.

Thus, to input a line of text as more than one field, one can say:
1. 1(N03) , EN~a2

Field type notations are discussed in greater detail in the documentation for lexical texts. In short, they consist of an exclamation mark followed by lower case letters, e.g., '!pr'. They should follow the comma except for a notation which applies to the initial field; since the first field of a line is marked implicitly, its type notation appears independently at the start of the line:
1. !pr du-ru ,!sg A

ST.7.ii Columns
Lines may be divided into columns using the ampersand ('&') character. Spaces are required around the ampersand. Thus, to input a ledger in tabular form one could write:
1. 1(u) & 1(u) & 2(u)
2. 1(u) & 1(u) & 2(u)

The difference between intra-line columns and intra-line fields is that fields are portions of a line with distinct content, such as translation as found in a bilingual lexical text, whereas columns are portions of a line which are aligned on distinct spatial boundaries that need not but may indicate distinct content. Because columnar organization often implies differentiation of content (as in lexical texts), field type notations may be given immediately after the ampersand of a column division.

In short, when a single line is divided according to content, the field marker (',') should be used, but when multiple lines are divided internally to form columns of text, with or without the columns indicating distinct content, the column marker ('&') can be used.

**ST.7.iii  Whitespace**

The third method of indicating text alignment is to use whitespace (spaces and tabs). This form of alignment is not structural, in that the ATF processor does not do any special translation of the whitespace when generating the XML and therefore should be avoided in most cases. It is, however, reversible, because the XML format encodes the literal whitespace as an attribute (unless the literal whitespace is a single space character [ASCII 32]). This means that processors which extract an ATF format text from an XML format text are able to generate the whitespace that was found in the original input.

Note that it does not matter if extra whitespace is intentional or unintentional in this implementation. It makes no difference to the archival XML data format.

**ST.8  Non-transliteration Material**

Non-transliteration material is specified using lines beginning with a "$"-sign.

A "$"-line should be formed on the basis of specific patterns; the parser recognizes key-terms and converts them to a more tightly-structured set of attributes. Here, we give some patterns to follow in entering "$"-lines; the quantities applicable to specific occurrences
may change, but the patterns should stay the same.

In each case, any text not in parentheses is the text you enter to encode a specific kind of feature on the object. To encode 'unknown number of' lines or columns blank/broken, use, e.g., "n lines broken."

**ST.8.i Single line features**
- broken (i.e., 1 line broken)
- traces (i.e., traces of 1 line)
- blank (i.e., 1 line blank)

**ST.8.ii Multi-line features**
- n lines broken
- 3 lines blank

**ST.8.iii Column-level features**
- n columns blank
- 1 column broken
- 2 columns with traces
- rest of column blank

**ST.8.iv Interlinear Rulings**
- ruling (i.e., single ruling)
- single rule
- double rule

**ST.8.v Intercolumnar Rulings**
- single column rule
- double column rule

**ST.8.vi Seal impressions**
- seal n (see notes on seals above)

**ST.9 Comments**

In general, comments are discouraged. However, support is provided for comments, and
the comment mechanism may be used to support extensions to the ATF format.

Comments may be interspersed among the text entered in ATF format, with each comment line beginning with '#' followed by a whitespace. Comments belong to the line preceding them; a comment occurring just after the '&' line belongs to the text as a whole. Multiple comments can be attached to a text or line by separating them with a blank line, e.g.,

```
& Text 1
# This is a comment about text 1.

# This is another such comment.
1. a ba bu
# This is a multi-line
# comment about line 1.
```

**ST.9.i Multiplexing**

Similar to a comment line is a multiplexing line which follows a line of transliteration containing an interpreted reordering of signs. Multiplexing is used primarily for Early Dynastic texts in which sign order is arbitrary. A line of transliteration with interpreted reordering and its multiplexing comment line take the following form:

```
1. X Y Z
   =: Z X Y
```

Discontiguous glosses can also be clarified in this way, e.g.,

```
{{%a id}}mu-un-cum2{{%a din}}
```

on the tablet should be rendered:

```
{{%a id-din}}mu-un-cum2
```

with the additional multiplexing comment line if required as follows:

```
1.    {{%a id-din}}mu-un-cum2
    =:    {{%a id}}mu-un-cum2{{%a din}}
```

**ST.9.ii ATF Comments**
Some special comments are defined to enable communication with the ATF processor program, these have the form:

#atf <keyword> <value>

Recognized keywords are documented in the various sections of the ATF documentation and in the index of ATF comments. All ATF comments precede the &-line of a transliteration.

**ST.10 Example**

Here is an artificial example text which illustrates many of the things described above.

@transliteration
&DC 265
@tablet
@obverse?
@column 1
1. 1(N03) , EN~a2
$  rest of column broken
@column 2
1. 1(u) & 1(u) & 2(u)
2. 1(u) & 1(u) & 2(u)
$  single rule
$  seal 1
$  traces
1'. 1(asz) udu
# unusual form of the udu sign
@reverse?
@column 1
1. 2(asz) udu
$  2 lines broken
$  1 column lost
@left
$ seal 1
1. bu bu
@envelope
@obverse
1. 2(asz) udu
@seal 1
1. da-da
2. arad-zu
AT: ASCII Conventions for Cuneiform Text Transliterations

Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

30 June 2004

AT.1 Background

This document describes a simple ASCII set of transliteration conventions which were originally developed on the basis of various existing practices for use by the CDLI project in data entry of Sumerian administrative texts.

The conventions described here are compatible with the original CDLI conventions, but add facilities for representing language shifts and also extend the definition of compound grapheme syntax to support the full needs of Sumerian sign forms.

The primary purpose of these conventions is to enable content preparers to input data with minimal overhead, but they also serve a useful secondary purpose for translation between the many different formats used by individuals in preparing work for publication using, e.g., Microsoft Word. This is because character-oriented (as opposed to markup-oriented) transliterations tend to use a similar underlying model even when they use different brackets, characters for shin, etc. Because the model implemented by the XML data format of CDLI (and, hopefully, other projects) provides generic support for cuneiform transliteration, the ASCII format given here can be used as an interim target for translating between many different ad hoc notations and the rigorously specified CDLI format.

We frequently refer to this input format as the ATF format, i.e., the ASCII Text Format.

AT.2 Conventions

The standard range of features is supported in the ASCII input syntax.

- Separate graphemes that form part of the same word should be joined by hyphens.
- Signs that are ligatured (sharing wedges) may be joined with a plus, e.g., um+me.
• Words are separated by spaces
• Determinatives should be bracketed as in {d}nanna or nibru{ki}
• Signs accidentally omitted by the scribe and then supplied by the editor should be put in angle brackets, e.g., <ba>.
• Signs intentionally omitted by the scribe should be put in angle brackets and parentheses, e.g., <(ba)>. Examples of this include refrains in litanies and lexical lists which omit repeated words.
• Signs removed by the editor should be put in double angle brackets, e.g., <<ba>>.
• Signs corrected should be followed by '!'. For numbers, the extant text on the tablet should always be given as well, in parentheses. E.g., 2(disz)!(1(disz)).
• Signs which have an unusual form should be followed by '!!'.
• Signs which are not expected but which really are there and cannot be corrected should be followed by '!!!!'
• Uncertain signs should be followed by '?'.
• Collated signs should be followed by '*' but only when the fact of the collation could be of interest (e.g., a correction to the published copy).
• Damaged signs should be followed by '#'; the convention of putting square brackets in the middle of a transliterated sign is not be used (i.e., write [ba−ab]−du# never [ba−ab−d]u).
• (The sequence of the '!', '*', '?' and '#' flags is unimportant)
• Signs which are no longer on the tablet should be given in square brackets.
• Square brackets can also be used with 1 or more 'x'-signs or with '...' (3 periods without spaces) to indicate a known or unknown number of missing signs (e.g., [x x], [...]).
• 'x' (lowercase) can be used to indicate unclear signs. The assumption is that a sign transliterated by "x" is broken.
• 'X' (uppercase) can be used to indicate clear signs whose identity has not yet been established but only if a sign-list reference cannot be provided.
• ';' can be used to indicate a place on the tablet where the scribe has begun a new line in between rulings (often referred to as an 'indented' line).
• '~' precedes signs to be read as logograms, (e.g., ~A = logogram A, ~A.BA = logogram A.BA). A space, end-of-line, or any delimiter other than a period ends the logogram, so writing ~AN-e indicates that only the AN is to be read as a logogram.
• In cases where it is appropriate to render text in normalized Akkadian, the following conventions are used to indicate long vowels:
  _ a macron is represented by = before the vowel (e.g., =a = _)
 a circumflex is represented by ^ before the vowel (e.g., ^a = â)
• Language shifts may be specified using a notation described below.

AT.3 Grapheme Representation

Body
• The body of a grapheme contains only the characters A to Z and '. No accented characters of any kind should be used.
• gna is represented with 'j', 'J'; CDLI transliterations do not use 'j'; other corpus projects may.
• heth is represented with 'h', 'H'
• shin is represented with 'c', 'C' or 'sz', 'SZ'
• sin is represented with 's'' (s followed by apostrophe)
• sadhe is represented with 's,', 'S,' (s followed by comma)
• teth is represented with 't,', 'T,' (t followed by comma)
• aleph is represented with ''' (the apostrophe character)
• for the semivowel 'j', we use 'y' to avoid conflict with 'gna'

Indices
The indices of graphemes are given in regular ASCII digits 0 to 9. Graphemes which are not yet assigned an index in the standard sign-lists should be specified using 'x' followed by the sign name in parentheses, e.g., kirix(KA). If qualifying a sign-name with 'x' you must use uppercase, 'X' (i.e., LUMX, not LUMx; this is to avoid an ambiguity with 'x' meaning times in combination with grouping).

Extension
We provide a general mechanism for extending the grapheme to support notation of paleographic variants (as in the Archaic texts, for example). The structure is a tilde followed by any sequence of lowercase letters and digits, or followed by a single '-' or '+ character, e.g., EN~a EN~2b EN2~a1 EN2~23.

Normalization
Circumflex and macron accents in transcribed Akkadian can be indicated by a carat (') or equals (=) respectively followed by the vowel; e.g., ^a for u+circumflex (û), =a for a+macron.

Sign-values
This is the normal case; the grapheme is lowercase.

Sign-names
The sign is known, but the value is uncertain; the grapheme is a sign-name in uppercase.
Signlist References
Sometimes even the name of a sign is unknown; in this case, identify the sign by signlist name and number. Put no hyphen between the signlist name and the number, e.g., KWU1, ZATU666. Valid sign lists are

- **ABZ**: R. Borger, *Assyrisch-babylonische Zeichenliste* (AOAT 33; Neukirchen-Vluyn 1978)
- **BAU**: E. Burrows, *Archaic Texts* (UET 2; London 1935)
- **CDSL**: Cuneiform Digital Library Sign List
- **HZL**: C. Ruster and E. Neu, *Hethitisches Zeichenlexikon* (Harrassowitz Verlag 1989)
- **KWU**: N. Schneider, *Die Keilschriftzeichen der Wirtschaftsurkunden von Ur III* (Rome 1935)
- **LAK**: A. Deimel, *Liste der archaischen Keilschriftzeichen* (WVDOG 40; Berlin 1922)
- **PSL**: Pennsylvania (Sumerian Dictionary) Sign List
- **REC**: F. Thureau-Dangin, *Recherches sur l'origine de l'écriture cunéiforme* (Paris 1898)
- **ZATU**: M. Green and H. J. Nissen, *Zeichenliste der Archaischen Texte aus Uruk* (ATU 2; Berlin 1987)

Numbers
The syntax of numbers is a repetition count and a unit identifier (given in parentheses), e.g., 1(asz); 5(u). Consult the *Numbers and Metrology documentation* for full details on numeric transliteration conventions.

Compound Graphemes
Graphemes whose reading are not yet known, but which consist of multiple elements or a modified element or elements, are given between vertical bars, e.g., |GA2xAN| or |GAN2@tl|.

x, X
lowercase 'x' is used to indicate that the sign on the tablet is unclear and cannot be identified. Uppercase 'X' is used to indicate that the sign on the tablet can be identified but has no identity (i.e., there is no way of referring to it in a published signlist, as it is not given in any such lists, or the lists are insufficiently subtle in their distinctions between sign-forms).
AT.4 Compound Grapheme Separators and operators

The compound grapheme operators are only recognized with a compound grapheme group delimited by vertical bars.

Juxtaposed
   
   
   

Written over/within
   'x', e.g., |GA2xAN|

Written over/crossed
   '%', e.g., |GI%GI|

Written above
   '&', e.g., |DU&DU| as opposed to standard DU

Ligatured
   '+', e.g., |LAGAB+LAGAB| (nigin2)

Written opposing
   '@', e.g., |LU2@LU2|

Written in square
   'x4', e.g., |LU2x4|

Grouping
   '(...)', e.g., |GA2x(ME.EN)|

Rotated
   '@<number>', e.g., |LU2.LU2@180|; rotations are counted in degrees clockwise.

Modifier labels

- gunu: '@g', e.g., |DU@g| as opposed to standard DU
- tenu: '@t', e.g., |GAN2@t|
- sheshig: '@s', e.g., |DU@s|
- nutillu: '@n', e.g., |SAG@n| as opposed to the standard SAG
- zidatenu: '@z', e.g., |ASZ@z|
- kabetenu: '@k', e.g., |ASZ@k| as opposed to the standard ASZ
- curved: '@c', e.g., |ASZ@c|
- reflected about vertical axis: '@r', e.g., |U@r|
- reflected about horizontal axis: '@h', e.g., |N07A@h| (aka N07B) as opposed to the
• variant: '@\v', e.g., 4 (ban2)@\v as opposed to the standard 4 (ban2)

Note that modifier labels go outside parentheses, e.g., 4(ban2)@\v rather than 4(ban2@\v).

**AT.5 Language Shifts**

Shifts between multiple languages within a single text can be indicated in one of two ways as described in AT.5.ii and AT.5.iii below. In addition, the default primary and alternate languages of a text can be set for the entire transliteration.

**AT.5.i Setting Default and Alternate Languages**

The default language (the one assumed when no language shift has been given for the current context) is Sumerian. It is possible to set a different default language for a text by using an **ATF comment** with the keyword 'lang' and a language shift code as the value, e.g.:

- `#atf lang a` (sets default language to Akkadian)

To set the default language for an entire text in this manner, give the `#atf lang` comment before the &-line. To set the alternate language use :

- `#atf altlang <language-code>`

If no altlang is given, the following alt-langs are set based on the default language:

<table>
<thead>
<tr>
<th>default</th>
<th>alternate</th>
</tr>
</thead>
<tbody>
<tr>
<td>%s</td>
<td>%a</td>
</tr>
<tr>
<td>%a</td>
<td>%s</td>
</tr>
<tr>
<td>%e</td>
<td>%s</td>
</tr>
<tr>
<td>%h</td>
<td>%a</td>
</tr>
</tbody>
</table>

Glosses which are given in single curly brackets are always Sumerian unless there is a language shift inside the opening curly bracket.

**AT.5.ii Shifting Languages (Standard)**
The standard form of a language shift is %<language-code> e.g., du = %a a-la-ku. The space after the language code is optional, and is discarded if given.

Note that the language shifts are only defined for switching into a language. Instead of conceptually enclosing a piece of text in a given language, they change the current language; this means that to change back and forth one simply specifies each time the new current language: e.g., %a a-la-ku = %s du.

These changes can be made within words, in which case the ATF processor encodes the language shifting at the graphemic level. The default language for a word is set by the first shift of any sequence of language shifts immediately preceding the word. This permits designating the language of a word as different from the language of the initial grapheme, e.g., (very contrivedly) %a %s lugal-%a be2-li2.

Language shifts remain in effect until the end of the current line, or until the end of the enclosing gloss group, field or intra-line column. So, to specify an Akkadian gloss one simply needs to say: {%a a-la-ku}du.

The following language shifts are defined:
- %s = Sumerian
- %a = Akkadian
- %h = Hittite
- %r = Hurrian
- %u = Ugaritic
- %e = Eblaite
- %x = unknown (for lexical texts or incantations where words or passages are in an unidentifiable language)

For Sumerian, three register/writing system shifts are provided analogous to the language shifts:
- %eg = Emegir
- %es = Emesal
- %ugn = UD.GAL.NUN

**AT.5.iii Alternate Languages Shift**
In addition to the standard method of switching languages, a shorthand method can be used to switch in and out of the alternate language of the text. As described above, the alternate languages can be either the standard default alternate language or the alternate language established for the text by using the `#atf altlang` comment.

An underscore character before and after the text

```
_<text>_  
```

indicates a shift to the alternate language between '_' characters. There should be no space after the opening '_' or before the closing '_'.

After the closing '_' the current language, which could be different from the default language, is resumed:

```
#atf lang s  
#atf altlang a  

... %e ba _bu_ bi ...  
```

In the example given above, ba is in Eblaite (%e), bu is in Akkadian (%a), and bi is back in Eblaite (%e) because the standard switch to Eblaite occurred before the alternate switch and remains in effect until the end of the line or until another standard switch is made.

### AT.6 Summary of Character Meanings

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A-Z][a-z][0-9]</td>
<td>grapheme name</td>
</tr>
<tr>
<td>&lt;hyphen&gt;</td>
<td>joiner for graphemes of single word</td>
</tr>
<tr>
<td>&lt;space&gt;</td>
<td>word separator</td>
</tr>
<tr>
<td>!</td>
<td>flags editor's correction of sign</td>
</tr>
<tr>
<td>!!</td>
<td>flags unusual sign-form</td>
</tr>
<tr>
<td>!!!</td>
<td>flags sign which is really on text despite expectation</td>
</tr>
<tr>
<td>?</td>
<td>flags uncertainty of identification or reading</td>
</tr>
<tr>
<td>*</td>
<td>flags collation</td>
</tr>
<tr>
<td>#</td>
<td>flags damage to sign</td>
</tr>
</tbody>
</table>
AT.7 Notes on Transliterating Determinatives and Glosses

Determinatives and glosses are simple to deal with in the ATF format, but a few simple principles need to be observed.

Determinatives and glosses which are written in the normal-sized script should be enclosed in single curly brackets, as in \{d\}.

Glosses which are written in smaller script should be enclosed in double curly brackets, as in szum2\{suz\}.

Glosses in smaller script which are written discontiguously on the tablet must be normalized as contiguous in ATF transliteration, i.e., the sequence of signs {id}mu-un-szum2{di-in} should be entered as mu-un-szum2{%a id-di-in}.

This restriction is necessary to minimize processing complexity when manipulating the text as linguistic data. A multiplexing comment line can be given to show the original
sign sequence (see ST.9.i).

**AT.8 Unicode Support**

Conversion of ATF character sequences to Unicode is supported through use of the `--unicode` argument to the ATF processor `atf2xml.plx`.

**AT.9 Example**

Here is a consolidated collection of examples to illustrate some of what is presented above.

- `ba-[ab]-du#`
- `ba-<ab>-du!(IC)`
- `ba-<<ab*>>-du?`
- `{d}nanna urim5{ki}-ma`
- `carx(NE)`
LK: Conventions for Intertext Links

Madeleine Fitzgerald and Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

18 June 2004

LK.1 Background

The ATF format supports inline notations of three relationships between lines in
different texts:
• parallels, in which a line or lines in one text are similar to lines in another text
• sources, or lines which are being included in a composite text but originate in other
texts
• contributors, or lines which are being entered in a transliteration and at the same time
are relevant to the reconstruction of a composite text

LK.2 Operators

Linking composite text lines to lines in individual texts, lines in exemplars to lines in
composite texts, and matching lines in two or more individual exemplars or composite
texts are indicated with the following notation at the beginning of the next line after the
line of transliterated text to be linked.
<<
   line comes from tablet instance (source)
>>
   line goes to composite text (contributor)
||
   before reference to line in parallel composite or tablet (parallel)

The source ('comes from') and contributor ('goes to') facilities are intended to allow
creation of composite texts in the absence of a complete set of transliterated sources and,
conversely, to allow the transliteration of individual sources by reference to a composite
text which may not yet include the original sources.
The parallel link facility is especially useful for editing tablets such as those containing liturgical texts and incantations where creating a composite text may be practically futile as well as methodologically dubious.

**LK.3 Targets and Link Definitions**

A **TARGET** is a reference to another text related to the one at hand. Every linked text must be defined and given a **TARGET** identifier in the file in which it is to be used before the first use; we recommend grouping all the definitions together just before the &-line (text ID/name line).

A linked text definition takes the form:

```
#atf def linktext <TARGET> = <ID> = <NAME>
```

For example:

```
#atf def linktext A = P227635 = CBS 10792 (OB Syllabary B)
```

The variable elements of this definition are:

**TARGET**
- the name that is used in the link specifiers; this is normally an uppercase letter, e.g., A, but the only actual restriction on the spelling of a target is that it may not contain any whitespace.

**ID**
- the ID as assigned by the CDLI project; for transliterated objects this begins with a 'P', e.g., P123456. Other text types have similar identifiers with different initial letters. ([see documentation on text naming](#))

**NAME**
- the human-readable name assigned by the CDLI project; in the XML format this is the value of the 'N' attribute. ([see documentation on text naming](#))

The ID and NAME values of texts can be obtained from the CDLI website; for texts which have not yet been assigned ID's, use an initial 'X' followed by digits as an interim measure and e-mail the CDLI staff to request IDs ([cdli@ucla.edu](mailto:cdli@ucla.edu)).

**LK.4 Labels**
The target is followed by a label that gives the location of the related text in the target document. Labels are constructed according to the following abbreviations (which are designed to be easy to type while still allowing programmatic reconstruction of the location of the reference in the XML dataset):

- o = obverse
- r = reverse
- t = top edge
- b = bottom edge
- l = left edge
- r = right edge
- e = edge
- <surface> = other surface, e.g., shoulder
- face a..z = prism face a to z
- seal n = seal n (n = the number of the seal in the transliteration)
- <roman> = column number in lowercase roman numerals
- <line number>

Spaces are required between elements of a label, for example, o i 2 = obverse, column 1, line 2.

Two labels, a 'from' label and a 'to' label, are used when there is a need to indicate a range of text beyond a single line in the target document. A range requires a hyphen character between the two labels.

**LK.5 Syntax**

The syntax of these constructs is either:

<OPERATOR> <TARGET> <LABEL>

or:

<OPERATOR> <TARGET> <FROM_LABEL> - <TO_LABEL>

**LK.6 Implementation Notes**

A separate program manages the links in such a way that it is unnecessary to group together all of the links to a specific parallel. In other words, given three texts which
contain the same parallel, let's say Liturgy 1, 2 and 3, one can encode the relationship as follows:
@transliteration
#atf def linktext A = P222222 = Liturgy 2
&P111111 = Liturgy 1
1. a-u-a
|| A 1

&P222222 = Liturgy 2
1. a-u2-a

#atf def linktext A = P222222 = Liturgy 2
&P333333 = Liturgy 3
1. a-u3-a
|| A 1

The link manager will resolve the links in Liturgy 1 and Liturgy 3 and construct a link-ring in which all three parallels refer mutually to each other. The search engine will automatically display all parallels whenever a match is found in any of the lines.

LK.7 Examples

(a) Source (line in composite 'comes from' exemplar):

File 1:
&P121323 = OB Lu excerpt N 4304
1. lu2

File 2:
@composite
#atf def linktext A = P123123 = OB Lu excerpt N 4304
&Q123238 = A = OB Lu A
1. lu2
<< A 1

(b) Contributor (line in exemplar 'goes to' composite)
The above example shows two exemplars of a text. The second transliteration contains a definition of the first text as "D" and indicates that its first line is paralleled (||) in the first line of the second column of the reverse of the first text. Parallels may be drawn between composite texts, transliterations or a combination of the two.
LX: Additional Conventions for Lexical Text Editions

Madeleine Fitzgerald and Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

18 June 2004

LX.1 Background

This document defines and demonstrates the use of conventions which are used in addition to the standard ATF text conventions when editing lexical texts.

LX.2 The #atf Lexical Protocol

Lexical texts are indicated by a special protocol which should be given at the start of each file. This protocol takes the form:
#atf use lexical

LX.3 Field Types Defined for Lexical Texts

Columns of lexical texts are treated as fields in the ATF sense, i.e., they are segments of a line which have distinct content. The ATF field-separator ',' is used to separate columns of a lexical text; if the physical alignment of a lexical text is to be mimicked, the ATF column-separator code ' & ' can be used instead.

Fields are marked for their content, whether the column contains a sign, a pronunciation, a translation or other data type. If the column is unmarked, the column contains a word or phrase.

The following are the markers for column types. Note that there are both shorthand and explicit markers. Shorthand markers must be preceded and followed by at least one space or tab character.
In addition, a bullet-character may be transliterated at the start of the line using '*-', optionally followed by the grapheme in parenthesis, e.g., * or *(disz).

Example:
1. !pr e-a ,!sg A ,!eq %a na-a-qu

which may also be entered as:
1. " e-a ~ A = %a na-a-qu

This is a three-column text with the first column being the pronunciation, the second being the sign, and the third being the translation, in this case into Akkadian as indicated by the standard language shift marker "%a." The first example is the full form with the standard notation for field breaks, ',' followed by the notation for the type of column. The second example above is the same transliteration with shorthand rather than explicit notation. Remember that it is very important to have whitespace on either side of the shorthand markers.

LX.4 Further Examples

LX.4.i Sign lists

LX.4.i.a Paleographic Ea
1. !sg A

This example shows how to mark up a single column list of sign names.
LX.4.i.b Proto-Ea
1. !pr e-a ,!sg A

LX.4.i.c Proto-Aa
1. !pr e-a ,!sg A ,!eq%a mu-u

Here we have a three-column list with pronunciation, sign name, and akkadian translation. Shorthand for the same line of translation would be:
1. " e-a : A = %a mu-u

LX.4.ii Thematic Lists

LX.4.ii.a Unilingual
1. !wp a

or:
1. ^ a

Note that because !wp is the default field type, this can also be written as:
1. a

LX.4.ii.b Bilingual
1. !wp a ,!eq%a mu-u

In this case we have a two column list with the Sumerian word in the first column and the Akkadian translation in the second. The shorthand version would be
1. ^ a = %a mu-u

or (because an unmarked column is assumed to contain a word or phrase):
1. a = %a mu-u

Note again the whitespace on each side of the shorthand markers ^ and = in the last two examples above.

LX.4.ii.c Trilingual etc.
1. !wp a ,!eq%a mu-u ,!eq%h ba-ba
A three-column text with Sumerian, Akkadian, and Hittite, which can also be rendered in shorthand as:
1. 1. ^ a = %a mu-u = %h ba-ba

or:
1. a = %a mu-u = %h ba-ba

LX.5 Examples with Structure

LX.5.i Prism with unilingual Sumerian vocabulary excerpt from Hh
#atf use lexical
&Pxxxxxx = Hh IX excerpt 44
@prism
@face 1
@column 1
1. ,!pr a-ab ,!wp ab
2. ,!pr i-ig ,!wp ig

LX.5.ii Syllabary
#atf use lexical
&Pxxxxxx = XX
@tablet
@obverse
1. * ,!pr du-u ,!sg KAK

or:
1. * " du-u : KAK

LX.5.iii Trilingual
#atf use lexical
&Pxxxxxx = XX
@tablet
@obverse
1. " tak-tak ~ TAK4.TAK4 | tak min-a-bi = %a e-ze-bu = %h ar-ha da-lu-mar
LX.5.iv Unilingual Proto-Ea
#atf use lexical
&Pxxxxxx = XX
@tablet
@obverse
1. " su-un : BUR2
2. " bu-ur : BUR2
3. " du-un : BUR2
4. " u3-szu-um : BUR2

LX.5.v Bilingual Proto-Ea
#atf use lexical
&Pxxxxxx = XX
@tablet
@obverse
1. ,!pr mu-ul ,!sg MUL ,!eq%a ka-ka-bu
2. " ~ =%a szi-t,ir-tu
3. " ~ =%a na-pa-hu
4. " ~ =%a na-ba-t,u
5. " szu2-hub2 ~ MUL = %a zu-hu-pu

Remember that !pr is equivalent to ", not "ditto," and !sg is equivalent to ~ ; .

If you want to indicate that empty space is meant to indicate a repetition of data from a preceding line, you can include the data between <(...)>(intentional omission supplied by editor). The example above would then be rendered as follows:
#atf use lexical
&Pxxxxxx = XX
@tablet
@obverse
1. ,!pr mu-ul ,!sg MUL ,!eq %a ka-ka-bu
2. <(mu-ul)> ~ <(MUL)> = %a szi-t,ir-tu
3. <(mu-ul)> ~ <(MUL)> = %a na-pa-hu
4. <(mu-ul)> ~ <(MUL)> = %a na-ba-t,u
5. " szu2-hub2 ~ MUL = %a zu-hu-pu
LX.5.vi  Emesal

#atf use lexical

&Pxxxxxx = XX

@tablet

@obverse

1. %es ga-sza-an = %eg nin = %a bel-tu
2. %es u5-mu = %eg i3-gisz = %a el-lu
3. %es ze2-eg3 = %eg szum2 = %a na-da-nu
MX: Miscellaneous CDL Document Types

Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

18 June 2004

MX.1 Background

This document describes the minor data types which are supported in the ATF/XTF software as well as giving some basic idea of the types for which support is planned in the near future.

MX.2 Composites

Composite texts are entered in a manner similar to transliterations. They must be prefaced by the @composite directive to inform the ATF processor that this is a composite text.

The &-line is similar to that given for a transliteration, but the identifier of a composite text begins with 'Q' rather than 'P'.

MX.2.i Links and Inline Exemplars

In a composite text, the other sources may be referred to as links or by transliterating them in-line in the form of exemplars entered immediately after the reconstructed line. For the linking conventions see the documentation on text linking. Conventions for giving exemplars in-line are described below.

MX.2.ii Siglum Definitions

When transliterating exemplars, siglum definitions must, as with all other ATF comment lines, precede the &-line (text ID/name line), one for each siglum that is referred to in the body of the composite. Siglum definitions have the following form: #atf def siglum <TARGET> = <ID> = <NAME>
The variable elements of this definition are:

**TARGET**
the name that is used in the link specifiers; this is normally an uppercase letter, e.g., A, but the only actual restriction on the spelling of a target is that it may not contain any whitespace.

**ID**
the ID as assigned by the CDLI project; for transliterated objects this begins with a 'P', e.g., P123456. Other text types have similar identifiers with different initial letters.

**NAME**
the human-readable name assigned by the CDLI project; in the XML format this is the value of the 'N' attribute.

The ID and NAME values of texts can be obtained from the CDLI website; for texts which have not yet been assigned ID's, use an initial 'X' followed by digits as an interim measure as well and e-mail the CDLI staff to request IDs (cdli@ucla.edu).

**MX.2.iii Exemplar lines**

Besides lines which have the normal **ATF line number syntax** (i.e., a non-space string at the start of the line followed by a period followed by one or more space), composites may contain exemplar lines that have the following syntax:

```plaintext
<SIGLUM> ':' <SPACE>  
```
or:

```plaintext
<SIGLUM> '_' <LABEL> ':' <SPACE>  
```

Here, `<SIGLUM>` must be a siglum name as defined in a siglum definition earlier in the composite text.

The optional `<LABEL>` is joined by an underscore character to the siglum name; the first sequence of `:'` followed by a space is taken as the end of the label. The rules for forming a label are given in the documentation on links.

**MX.2.iv Example**

```plaintext
@composite
#atf def siglum A=P123456=Textname1
```
Matrices will be fully defined and documented at a later date.

**MX.3.i Example**

@matrix
#atf def siglum N1 = N2343
#atf def siglum N2 = N4321
&R123234 = Matrix of Shulgi A
1. a-b c
N1 + + -
N2 . + +

**MX.3.ii Implementation notes**

Several complete matrices already exist in a form suitable as extended examples (see Tinney's work on the Tetrads). A program (‘matrix’ written by Tinney in the C programming language) which reads matrices and exports them either as reformatted ASCII or in a TeX format suitable for typesetting already exists. The remaining work is a formal definition of the ASCII syntax, the definition of a suitable XML data structure, and the production of an output routine for the C program to generate appropriate XML. The C program will simply be called from the Perl atf2xml program to return XML fragments for each block of lines.

**MX.4 Synoptica**

Synoptic texts are subject to the same rules as composite texts with the certain exceptions that will be documented soon.

**MX.4.i Example**

@synopticon
#atf def siglum A=P123456=Textname1
MX.5 Tabulations

Tabulations will be fully defined and documented at a later date.

MX.5.i Example

@tabulation
#atf def tabcol 1 = MSL OB Lu
#atf def tabcol 2 = NV OB Lu
#atf def tabcol 3 = JT OB Lu
&T123123 = MSL OB Lu vs. NV OB Lu
1-100 = 1-100 = 1-100
101 = - = -
102-200 = 101-199 = 101-199

MX.5.ii Implementation notes

Ranges will be required to contain same the number of lines; ',' will be supported in ranges, e.g.:
1-9,11-99,103-113 = 1-100
10 = -

Absent lines will be required to be notated to avoid tabulation compiler warnings about omitted entries. Ranges require a hyphen between the two numbers.
AC: Overview of ATF Comments

Madeleine Fitzgerald and Steve Tinney

For the CDLI, DCCLT and PSD Projects; placed in the Public Domain

30 June 2004

AC.1 Background

Special comments are defined to enable communication with the ATF processor program. These protocols all precede the &-line (text ID/name line) and have the form:

#atf <keyword> <value>

Recognized keywords are documented below and in the various sections of the ATF documentation.

AC.2 Set Languages

AC.2.i #atf lang

To set the default language for an entire text, give the #atf lang comment before the &-line (text ID/name line), e.g.,

#atf lang a

sets the default language to Akkadian. (See AT.5 for language shift documentation.)

AC.2.ii #atf altlang

To set the default alternate language for an entire text, give the #atf altlang comment before the &-line (text ID/name line), e.g.,

#atf altlang a

sets the default alternate language to Akkadian.
If no altlang is given, the following alt-langs are set based on the default language:

<table>
<thead>
<tr>
<th>default</th>
<th>alternate</th>
</tr>
</thead>
<tbody>
<tr>
<td>%s</td>
<td>%a</td>
</tr>
<tr>
<td>%a</td>
<td>%s</td>
</tr>
<tr>
<td>%e</td>
<td>%s</td>
</tr>
<tr>
<td>%h</td>
<td>%a</td>
</tr>
</tbody>
</table>

(See AT.5 for language shift documentation.)

AC.3 #atf rws

The 'rws' keyword functions like 'lang' to set the register/writing-system at a global level, e.g.,

#atf rws es

sets the register as Emesal for the entire transliteration. (See AT.5 for list of registers/writing-systems.)

AC.4 #atf use mylines

In a transliteration, the atf2xtf processor automatically generates a new set of line numbers to ensure consistency within the CDLI system. To override this default, the ATF comment

#atf use mylines

may be used, but this should be a rare occurrence. For all other text types, the default behavior is to keep the line numbers used in the ATF file.

AC.5 #atf use lexical

Lexical texts are indicated by a special protocol which should be given at the start of each file, before the &-line (text ID/name line). This protocol takes the form:

#atf use lexical

For more on lexical list conventions, see LX.2.
AC.6 #atf def

The #atf def comment is used for relating two or more texts. Multiple relationship definitions can be given, but each #atf def comment should be on a separate line. The following three values can be used with the #atf def comment:

AC.6.i linktext

As described in the text linking documentation (LK), the linktext value is used with the #atf def comment to define a text related to the one at hand. Link(text) definitions take the form:
#atf def linktext <TARGET> = <ID> = <NAME>

For example:
#atf def linktext A = P227635 = CBS 10792 (OB Syllabary B)

AC.6.ii siglum

In a composite text, the other sources may be referred to as links (as described in AC.4.i above) or by transliterating them inline in the form of exemplars entered immediately after the reconstructed line. When transliterating exemplars inline, siglum definitions are used instead of link definitions as follows:
#atf def siglum <TARGET> = <ID> = <NAME>

See the siglum documentation (MX.2.ii) for more information

AC.6.iii tabulation

Tabulation is not yet fully defined and documented. The general idea is that the #atf def comment will take the tabcol value in order to define the texts to be aligned for comparison in table format. Table column definitions will take the form:
#atf def tabcol <TARGET> = <ID> = <NAME>

See the tabulation documentation (MX.5) for more information
NM: Numeric and Metrological Notations Basics

Robert Englund & Steve Tinney

For the CDLI and PSD Projects; placed in the Public Domain

2 July 2004

NM.1 Introduction

This document gives basic guidelines for entering numbers and metrological transliterations for CDLI. We are preparing an exhaustive account of numbers and metrology, a preliminary version of part of which is available for download (pdf 120KB).

We are aware of the fact that this is a preliminary version and is restricted to more or less uncontroversial numerical/metrological systems of the Ur III period; further, that while restricted, the proposal does include a paradigmatic completeness, particularly of the smaller units of the systems, that is not covered by our text corpus. At the same time, complex fractions of the NINDA2xSIGNS type, and the igi-n-gal2 complex, are not complete.

We are thankful for any advice and corrections of mistakes or omissions you might offer us in this, and subsequent proposals that will deal with other, and earlier 4th and 3rd millennium systems.

A previous paper analyzing the requirements and laying out a design for the XML infrastructure for number and metrology handling is available (see Numbers Whitepaper).

NM.2 How to Transliterate Numbers

All numbers are given with digits or a fraction followed by a qualifier in parenthesis, e.g., 1(disz). A full list of the allowable forms is given in the reference document.

NM.3 Large Numbers Transliterated According to Their Graphemes

Numerical notations are to be transliterated in the form n(sign1) n(sign2) etc., for
instance "3(gesz2) 5(u) 7(disz) gurusz" for "180 + 50 + 7 workmen" or "6(bur3) 2(esze3) 4(iku) 1/4(iku)" GAN2" for "6 bur 2 esze 4 1/4 iku field."

NM.4 A Note on UxKASKAL

Note that for the time being we will transliterate Ugunu/UxKASKAL ("10 bur") with "bur'u" in avoidance of either buru (assigned to BUR by the sign lists) or burux (requiring description in parentheses that would unnecessarily complicate perl scripting); in like manner, szar'u for SZAR2xU, although this sign is listed with the value szaru by Borger.

NM.5 Systems Included in the Reference Tables [download (pdf 120KB)]

- Sexagesimal
- Length
- Surface
- Volume
- Dry & Liquid Capacity
- Weight
- Brick

NM.6 Systems on the TODO list

- pre-Ur III generally; eventually post-Ur III also
- Bisexagesimal
- Decimal
- Baskets
- others
WNM: Whitepaper on Numeric and Metrological Notations for Cuneiform Text
Transliterations

Steve Tinney

For the CDLI and PSD Projects; placed in the Public Domain

24 June 2004

WNM.1 Introduction

WNM.1.i Note on Revised Version

This document was originally written in April 2002 in order to establish the general principles for dealing with numbers and number systems for CDLI. As of June 2004, the infrastructure described below has largely been implemented in the ATF to XTF processor's number parsing routines. The body of this document has not yet been revised as the XML data formats have not yet been finalized; the major change in the text below is the elimination of the 'minimalist' notation. For further information on number notations see the Numbers and Metrology Basics document.

WNM.1.ii Disclaimer

The development of a set of precise notations for cuneiform numeric and metrological data poses a substantial challenge; this document aims to work from the known to the unknown, in the explicit expectation that it will be revised repeatedly, perhaps interminably. It is also expected that any revisions will be backward-compatible with previous versions of this document.

WNM.1.iii Use-cases

We require a notation and implementation which:
• is easy for data enterers to input
• is easy for data collators to correct
• permits unambiguous representation of numeric graphemes
• permits representation on the level of raw formal graphemes (3(u))
• permits representation on the level of interpreted graphemes (3(bur3))
• facilitates reliable machine-interpretation
• supports an override mechanism to force specific machine-interpretation
• converts without loss of information from ASCII format to XML format and back again
• is expandable
• allows conversion to ancient base values (for mathematical operations)
• allows conversion to modern base values (for visualizing quantities)
• is portable (i.e., can be used with DTD's and Schemas other than the CDLI one)?

WNM.2 Definitions

It is necessary to define specific terms for each of the aspects of number-production in cuneiform writing.

WNM.2.i Formal Constituents

Unit
an element of measure, e.g., sila3, gur, etc.
Count
a means of expressing 1 or more of a Unit.
Value
a single combination of COUNT and UNIT.
Ancient Equivalency Value (AEV)
an ancient equivalency VALUE, e.g., 5000 sila3; a property of either UNIT, VALUE or AMOUNT.
Modern Equivalency Value (MEV)
a modern equivalency VALUE, e.g., 1 liter; a property of either UNIT, VALUE or AMOUNT.
System
a definitional collection of UNITs, COUNTs, VALUEs, AEVs and MEVs which may optionally be localized in time and/or place and/or by keyword.
Quantity
a complete sequence of VALUEs expressed in a given SYSTEM.
Commodity
the counted referent of an AMOUNT.
WNM.2.ii  Written Instantiation

Count-grapheme
   a grapheme which represents a simple numeric value such as 1(disz); Count-
   graphemes are ambiguous with respect to unit in context-free situations.

Unit-grapheme
   a grapheme which establishes the unit of the preceding count-grapheme.

Integral-value-grapheme
   a grapheme which represents a VALUE in a SYSTEM by virtue of its integral
   constitution, e.g., BANMIN, i.e., "PA" in the sense of 2-BAN. Integral-value-
   graphemes are unambiguous with respect to unit in context-free situations.

Repeater-value-grapheme
   a grapheme which represents a VALUE in a SYSTEM by virtue of being repeated
   to express multiples of itself, e.g., esze2. Repeater-value-graphemes are
   unambiguous with respect to unit in context-free situations.

System-grapheme
   a grapheme which serves to specify the SYSTEM of an AMOUNT.

System Diagnostic Significance (SDS)
   a property of graphemes or grapheme combinations which serves to identify
   diagnostically the SYSTEM of a given AMOUNT. ISSUE: HOW TO SPECIFY
   THIS FORMALLY?

WNM.3  Implementation

'Implementing numbers' for CDLI requires definition of several formats and the creation
of software to bridge the formats:

WNM.3.i  ATF Notations

The fully-qualified notation explicitly notates every grapheme which makes up an
AMOUNT, and qualifies the grapheme in terms of count and unit.

In fully-qualified notation, the graphemes are disambiguated according to their form and
unit-reference. E.g., 1(disz), 3(sila3).

Additionally, the fully-qualified version will explicitly give the system at the start of the
AMOUNT. The notation used for languages, i.e., %s (=sumerian) and so on can be extended for use with SYSTEMS to avoid having too many escape characters. Thus, the fully-qualified version will express a sexagesimal number as, e.g., %sx 1(disz).

Given that reducing all localized systems to a brief short-hand will be difficult, we should anticipate that less commonly used systems will be named with a notation such as %sys=<system-name>, e.g., %sys=sexagesimal.

WNM.3.ii XML Notations

In the present implementation of cdli-text.dtd a generic tag <n> is used to identify numbers. This tag will remain as a non-system-specific way of marking a sequence of graphemes known to represent a number. In the normal case, however, the N tag will be superseded by a Q tag indicating a QUANTITY. This step will normally be made programmatically, and the extant N will remain embedded in the Q for ease of auditing.

The new Q tag is defined by the following DTD fragment which is simplified to the extent that it is not namespaced.

<!--
  The Q element contains a QUANTITY consisting of an optional N (which is the original grapheme sequence classified as N in initial conversion process) followed by a sequence of COUNT-UNIT pairs, an optional system-classifier (SYS-CLASS).

  The @SYSTEM attribute on Q defines the SYSTEM and is an IDREF which should refer to the XML database of systems.

  The @MEV attribute enables caching of modern-equivalency-values.

  The @AEV attribute enables caching of ancient-equivalency-values.

  This means that display of texts can toggle between displaying the original 'N', the digested sequence of count-unit pairs, the modern equivalent or the ancient equivalent (expressed in the base unit of the respective system).
-->


We assume that N contains graphemic data as defined in cdli-text.dtd, and that this data represents the actual grapheme sequence rather than interpreted values (i.e., numbers like 263 are deconstructed into their constituent elements).

Note that with this design we do not link directly from count-unit pair to written instantiation; should we? Also, we do not cache equivalency values for individual count-unit pairs; should we?

WNM.3.iii SYSTEM

Every QUANTITY given in an actual text must have a specified SYSTEM (even if that SYSTEM is the special value 'unknown'). This allows software to perform more advanced validity-checking as well as equivalency calculation. It also opens up the potential for corpus-queries such as 'show me all uses of system x where it is measuring y'.

Systems are defined in a separate XML document, and identified by identifiers which consist of a base system (sexagesimal, weight, capacity, etc.) followed by optional qualifiers in the sequence: BASE–DATE–PLACE–KEY.
This permits such identifiers as `weight-ur3-ur-royal`.

TODO: define the XML format of a system.

**WNM.4 Known Systems**

TODO: translate Bob's document to a format which can be imported here

TODO: ensure that every system has a %-escape

TODO: ensure that the acceptable graphemic elements of every system are clearly defined

TODO: every known system must have sample documents standard ATF notations.