WD 13250-6:
Topic Maps – Compact Syntax (CTM)

Editors:	Heuer, Hopmans, Oh, Pepper
Status:	Editors' Working Draft
Version:	0.3
Date:	2006-07-28
Issues:	• What functionality for templates? • What syntax for templates? • Restrict templates to header or allow anywhere? • What form should name and occurrence flags have (related to template syntax)? • How to represent type-instance and subtype-of shortcuts? • Should WS+ be required before ".", even in cases where it is not strictly speaking necessary? • Do we need Terms and definitions clause?

1. Introduction

CTM (Compact Topic Maps syntax) is a text-based notation for representing topic maps. It provides a simple, lightweight notation that complements the existing XML-based interchange syntax described in [XTM] and can be used for

- manually authoring topic maps;
- providing human-readable examples in documents;
- serving as a common syntactic basis for TMCL and TMQL.

The principal design criteria of CTM are compactness, ease of human authoring, maximum readability, and *comprehensiveness* rather than *completeness*. Thus, although CTM supports almost all the constructs of the [TMDM], care should be taken when using CTM as a basis for interchanging topic maps.

> **NOTE 1:** Clause 6 *Limitations* provides an overview of those aspects of [TMDM] that are not supported.

This document should be read in conjunction with [TMDM] since the interpretation of the CTM syntax is defined through a mapping from the syntax to the data model there defined.

2. Scope

This document defines a text-based notation for representing instances of the data model defined in [TMDM]. It also defines a mapping from this notation to the data model. The syntax is defined through an EBNF grammar, and more precision is provided through the mapping to the data model, which effectively also defines the interpretation of the syntax. Informative guidance on how to serialize instances of the data model to the CTM syntax is also provided.

3. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISSUE: Check that the EBNF in Annex A is conformant with ISO 14977.

[EBNF]
ISO/IEC 14977:1966 Information technology – Syntactic metalanguage – Extended BNF
http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

[IRI]
Internationalized Resource Identifiers (IRIs)
http://www.ietf.org/rfc/rfc3987.txt

[TMDM]
ISO/IEC 13250-2:2006 Information technology – Topic Maps – Data Model
http://www.jtc1sc34.org/repository/0696.pdf

[XML]
Extensible Markup Language (XML) 1.0 (Third Edition)
http://www.w3.org/TR/2004/REC-xml-20040204

[XSD-2]
XML Schema Part 2: Datatypes
http://www.w3.org/TR/xmlschema-2/

[XTM]
ISO/IEC 13250-3:2006 Information technology – Topic Maps – XML Syntax (XTM)
http://www.jtc1sc34.org/repository/FIXME.pdf

4. Terms and definitions

@@@TBD: Terms and definitions.

ISSUE: Do we need this clause or are definitions best given in the text itself? We are introducing some new terminology (e.g., "assertion", "assertion block", "topic identifier", etc.)

5. Syntax description

5.1 Basic structure

A **CTM document** consists of a *header* and a *body*. The header contains required and optional declarations. The body normally consists of *assertion blocks* and may also contain additional associations (called *verbose associations*). Comments and topic identifiers can occur in both the head and the body.

5.1.1 Comments

A **one-line-comment** starts with the "#" character and continues until the end of the line:

```
# This is a comment. It continues until the end of the line
```

A **multiline-comment** starts with "/*" and ends with "*/"; it can contain line breaks:

```
/*
This is also a comment.
It continues across
multiple lines.
*/
```

5.1.2 Topic identifiers

A **topic-identifier** is one of the following:

Name	Form	Example
local ID	`ID`	`puccini`
QName	`ID ":" ID`	`wikipedia:Puccini`
subject identifier	`HTTP-IRI`	`http://en.wikipedia.org/wiki/Puccini`
	`"<" IRI ">"`	`<urn:x-myns:music:puccini>`
subject locator	`"=" HTTP-IRI`	`=http://www.puccini.it/`
	`"=" "<" IRI ">"`	`=<ftp://example.com/opera/tosca/synopsis.txt>`

NOTE 2: An ID has the same form as a **Name** in [XML], except that it cannot contain a colon (":").

NOTE 3: CTM accords a special status to HTTP-IRIs. Unlike other IRIs they do not have to be delimited with "<" and ">".

NOTE 4: A local ID, like an XML ID in the XTM syntax, becomes an item identifier, whereas a QName expands to a subject identifier. (See the sections on prefixes and deserialization, below.)

5.1.3 Datatypes

CTM supports the datatypes of [XSD-2]. The following are "built-in", which means they are automatically recognized by the CTM parser:

String

The same as `xsd:string`. Strings are delimited either by double quotes (") or by triple double quotes ("""). Double quotes are escaped when necessary by a backslash (\"), e.g.:

```
"A string containing \"quote\" marks"
```

```
"""Another string containing "quote" marks"""
```

```
"""Quote marks at the end of a string must always be "escaped\""""
```

IRI

The same as `xsd:anyURI`. IRIs are normally delimited by "<" and ">"; however, IRIs belonging to the HTTP scheme may be written without delimiters, e.g.:

```
<http://en.wikipedia.org/wiki/>
```

```
http://en.wikipedia.org/wiki/
```

```
<urn:x-myns:music:puccini>
```

```
<ftp://example.com/opera/tosca/synopsis.txt>
```

Date

The same as `xsd:date`, e.g.:

```
1858-12-22
```

Integer

The same as `xsd:integer`, e.g.:

```
42
```

Decimal

The same as `xsd:decimal`, e.g.:

```
+42.0
```

Any datatype can be expressed by representing the value as a string and appending the datatype-qualifier (`^^`) and the IRI of the datatype, e.g.:

```
"2A" ^^xsd:hexBinary
"P1Y2M3DT10H30M" ^^xsd:duration
"12-22" ^^xsd:gMonthDay
```

5.2 Document header

The header of a CTM document includes the following information: version-directive, encoding-directive, prefix directives, and templates.

5.2.4 The version directive

The **version-directive** identifies the CTM document as being in CTM syntax and states the version number, which is currently "1.0":

```
%version ctm 1.0
```

5.2.5 The encoding declaration

The **encoding-directive** specifies the character encoding used by the document. If the encoding declaration is omitted, UTF-8 is assumed. The name of the encoding should be given as a string in the form recommended by [XML], e.g.

```
%encoding "Shift-JIS"
```

5.2.6 Prefix declarations

A **prefix-declaration** associates an identifier with an XML namespace and has the form "%prefix *id namespace*", e.g.

```
%prefix w <http://en.wikipedia.org/wiki/>
```

5.2.7 Templates

CTM allows templating of names, occurrences, and associations. The use of templates permits CTM documents to achieve greater compactness, readability and consistency.

ISSUE: Restrict to header or allow anywhere?

ISSUE: IMPORTANT NOTE: The syntax for templates used here is a **placeholder** for a better syntax that is yet to be devised. Alternative, less cryptic, and more powerful proposals are welcome.

5.2.7.1 Name templates

A **name-template** is used to assert that a certain assertion type is to be interpreted by the CTM parser as a name type and (optionally) to specify the scope of all such names.

```
%name  foaf:name                # foaf:name is a name type
%name  country-code  @iso639    # country-code is a name type; all names
                                # of this type are scoped by iso639
```

ISSUE: Not entirely happy with the example of a name type that is always in a certain scope. Can we find a better example? If not, is the whole notion of binding a scope to a name type (or occurrence type) misconceived?

5.2.7.2 Occurrence templates

An **occurrence-template** is used to assert that a certain assertion type is to be interpreted by the CTM parser as an occurrence type and (optionally) the datatype and/or the scope of all such occurrences.

```
%occur  bio:birthYear  ^^xsd:gYear
%occur  geo:lat_long   @deprecated
```

5.2.7.3 Association templates

An **association-template** is used to specify the role types to be inferred from an abbreviated association in an assertion block. It consists of a role type (specifying the role played by the subject of the assertion block), followed by the association type, zero or more additional role types, and (optionally) the scope to be assumed for all such associations, e.g.

```
%assoc  person  bio:born-in  place  @bio             # binary (with scope)
%assoc  victim  bio:killed-by  perpetrator  cause    # ternary
%assoc  work    opera:unfinished                     # unary
```

ISSUE: Would topic templates be useful?

5.3 Document body

5.3.8 Assertion blocks

An **assertion-block** consists of a topic-identifier for the subject of the assertions that follow, followed by one or more assertions, which are separated by semicolons. Assertion blocks are terminated by a period or an empty line:

```
subject          # the subject of all assertions in this block
  assertion-1 ;  # a semicolon separates this assertion from the next
  assertion-2 ;
  [...]
  assertion-n .  # a period terminates the assertion block (empty line could be used instead)
```

NOTE 5: When a period that terminates an assertion block comes after a local ID, a QName, or an undelimited datatyped value, it must be preceded by at least one space character.

White space is not significant, except as a separator between topic identifiers and when used to terminate assertion-blocks with an empty line:

```
subject
  assertion-1 ;          # space before semicolon
  assertion-2;           # no space before semicolon
  assertion-3            # period omitted; replaced by empty line

# multiple assertions on one line:
subject
  assertion-1; assertion-2; assertion-3; assertion-4 .

# whole block on one line:
subject assertion-1 .

# multiple blocks on one line:
subject assertion-1 . subject2 assertion-2 . subject3 assertion-3 .
```

NOTE 6: In the remaining examples in this document, the end of a block will be marked by an *empty line* when the example consists of multiple assertion blocks, and by a *period* otherwise.

5.3.9 Assertions

An **assertion** can be one of the following:

- a subject-identifier
- a subject-locator
- a name (or list of names of the same type)
- an occurrence (or list of occurrences of the same type)
- an association (or list of associations of the same type)

The general form of all assertions is

```
type  value  scope?  reifier?
```

where *type* is understood as

- either the "kind" of identifier (i.e., subject identifier or subject locator),
- or the [TMDM] [type] of the characteristic (i.e., name type, occurrence type, or association type);

and *value* can be either a single value or a comma-separated list of values, each of which gives rise to an assertion of type *type*. Thus

```
my-subject  my-type  value1,  value2,  value3 .
```

is equivalent to

```
my-subject  my-type  value1 ;
            my-type  value2 ;
            my-type  value3 .
```

which is also equivalent to

```
my-subject  my-type  value1 .
my-subject  my-type  value2 .
my-subject  my-type  value3 .
```

NOTE 7: For an explanation of the optional *scope* and *reifier* components of an assertion, see below.

In order to achieve greater brevity and/or improved readability, the general form of an assertion takes slightly different forms depending on the specific *kind* of assertion in question. The following sections described the specific form of each kind of assertion.

5.3.9.1 Subject identifiers

- *type* is always omitted and is inferred by the CTM parser;
- *value* is a single IRI;
- *scope* and *reifier* are not applicable.

```
puccini  <http://en.wikipedia.org/Giacomo_Puccini> .
```

5.3.9.2 Subject locators

- *type* is indicated by "=";
- *value* is a single IRI;
- *scope* and *reifier* are not applicable.

```
csgp-homepage =<http://www.puccini.it/> .
```

5.3.9.3 Names

- *type* can be omitted, in which case the default name type is inferred by the parser;
- *value* is a comma-separated list of strings (each one optionally followed by one or more variants);

```
puccini
              "Puccini, Giacomo" ;  # default name type assumed
   foaf:name  "Giacomo Puccini" .   # name type specified
```

Names with explicitly specified types are distinguished from occurrences by the presence of a name directive for the type in question. Thus, in the example above, `foaf:name` will be interpreted as an occurrence unless the document contains a directive like the following:

```
%name  foaf:name
```

Name templates may be overridden or dispensed with by prefacing the assertion with a name flag or an occurrence flag. The occurrence flag ("%occur") causes any existing name template for that type to be ignored and the assertion to be interpreted as an occurrence; the name flag ("%name") causes the assertion to be interpreted as a topic name, regardless of whether any template exists for that type.

```
#TM 1
%name foaf:name
[...]
puccini
  # force interpretation of assertion as occurrence
  %occur  foaf:name  "Giacomo Puccini"
```

```
#TM 2
puccini
# force interpretation as name, even in the absence of a name template
  %name  foaf:name  "Giacomo Puccini"
```

NOTE 8: Name and occurrence flags are mostly used in the context of topic map fragments where use of templates would be overly verbose. Otherwise the use of templates is encouraged, especially when authoring larger topic maps, in order to ensure both consistency and conciseness.

ISSUE: The exact form that the name and occurrence flags should take has yet to be finalized and will depend on the syntax chosen for templates; "%name" and %occur" are just **placeholders**.

Variant names are given in parentheses immediately following the base name (before any scope or reifier), and consist of a string or IRI followed by a space-delimited list of topic identifiers:

```
boito
    "Boïto, Arrigo" ( "boito, arrigo"  @sort ) .
```

5.3.9.4 Occurrences

- *value* a comma-separated list of strings (each one optionally followed by a datatype), IRIs, dates, integers, or XML fragments;

```
puccini
  article        <http://en.wikipedia.org/wiki/Giacomo_Puccini> ;       # IRI
  description    "The greatest of the verismo composers" ;              # string
  date-of-birth  1858-12-22 ;                                           # date
  bibref         """Budden, Julian: "Puccini: His Life and Works",
                 Oxford University Press (Oxford, 2002)""" ;            # string
  xml-bibref                                                            # XML
    """<bibitem id="budden"><bib>Budden</bib>
       <pub>Budden, Julian:
       <highlight style="ital">Puccini: His Life and Works</highlight>,
       Oxford University Press (Oxford, 2006)</pub>
       </bibitem>""" ^^xsd:anyType .
```

Occurrence templates may be overridden by prefacing the assertion with the name flag ("%name"), which causes the assertion to be interpreted as a name, regardless of any template for that type.

```
%occur iso639:code
[...]
norway
  %name  iso639:code  "no" .
```

5.3.9.5 Associations

- *value* is a comma separated list of role-player lists, each of which consists of zero or more topic-identifiers of role players;

```
# Example 1
puccini
  bio:born-in   lucca ;                         # binary association
  bio:died-in   brussels ;                      # binary association
  pupil-of      ponchielli, bazzini, angeloni   # 3 binary associations

# Example 2
scarpia
  bio:killed-by  tosca  stabbing                # ternary association

# Example 3
turandot opera:unfinished                       # unary association
```

Role types are assigned to role players using the corresponding association template, as follows:

- The first topic identifier in the template specifies the role type of the assertion block's subject (i.e., 'puccini', 'scarpia', and 'turandot' in the three examples above).
- The second topic identifier in the template specifies the *association type* and is used to match up with the *type* parameter of the assertion (i.e., 'bio:born-in', 'bio:died-in', and 'pupil-of' in Example 1 above).
- Further topic identifiers in the template specify the role type of the corresponding role players in the assertion; correspondence is based on positionality: the *first topic identifier* following the association type in the template specifies the *role type* of the *first role player* following the association type in the assertion, etc.

Assuming the templates given in Clause 5.3.2.3, the following assignments are therefore implied for (some of) the examples given above:

Association type	Role #1	Role #2	Role #3
bio:born-in	puccini: person	lucca: place	N/A
pupil-of	puccini: pupil	ponchielli: teacher	N/A
bio:killed-by	scarpia: victim	tosca: perpetrator	cause: stabbing
opera:unfinished	turandot: work	N/A	N/A

NOTE 9: An association (in an assertion block) *must* have a corresponding association template, and there can only be one association template per association type. Associations that do not have a template, or whose "signature" (i.e., configuration of role types) does not match that of the template, can be expressed using verbose associations, as described below.

ISSUE: If it is decided that templates should be allowed anywhere, and that multiple templates should be used for the same type, the note above must be changed.

Roles that are specified by the template but omitted in the assertion, can be indicated through the omitted role marker, "%role". In the following example, the role of 'perpetrator' is omitted and the role player 'shooting' is assigned the role type 'cause':

```
cavaradossi
  bio:killed-by  %role  shooting .
```

ISSUE: The keyword "%role" is a **placeholder** which will be aligned with other keywords when the template syntax has been finalized.

CTM has two built-in association templates, **ISA** ("is a") and **AKO** ("a kind of"), which correspond to the type-instance and supertype-subtype association types defined by [TMDM], respectively:

```
%prefix tmdm <http://psi.topicmaps.org/iso13250/model/>

# ISA association type
%assoc  tmdm:instance  tmdm:type-instance  tmdm:type

# AKO association type
%assoc  tmdm:subtype  tmdm:supertype-subtype  tmdm:supertype
```

The keywords ISA and AKO can be used within an assertion block in place of the topic identifier of the association type, as follows:

```
puccini ISA composer

# equivalent to
puccini tmdm:type-instance composer
```

```
composer AKO person

# equivalent to
composer tmdm:supertype-subtype person
```

ISSUE: The keywords "ISA" and "AKO" are **placeholders** whose form should be reviewed in the light of the templating syntax. The following issues should be considered: Are ISA and AKO the best choices as short-cuts for type-instance and supertype-subtype? Should they be more distinguishable from local IDs? Is it inconsistent to adopt the one (ISA) without adopting the other (AKO)? Is it too inconvenient to have to write uppercase? Is it a good idea to use ISA (or isa) at all, considering the confusion this leads to in distinguishing type-instance from supertype-subtype?

5.3.10 Scope

The scope of a characteristic is expressed using the "@" character followed by a space-delimited list of topic identifiers immediately after the assertion:

```
opera
  "Opera" ;  "Oopera" @fi ;
  dc:description """A drama set to music; consists of singing with orchestral
                    accompaniment and an orchestral overture and interludes."""
                    @wordnet ,
                 """Opera refers to a dramatic art form, originating in Italy, in
                    which the emotional content or primary entertainment is
                    conveyed to the audience as much through music, both vocal and
                    instrumental, as it is through the lyrics.""" @wikipedia en ,
                 """L'opera lirica è un genere teatrale e musicale in cui l'azione
                    scenica è sottolineata ed espressa attraverso, appunto, la
                    musica ed il canto.""" @wikipedia it .
```

5.3.11 Reifier

Reification of a characteristic is expressed using the "~" followed by a topic identifier for the reifying topic immediately after the assertion:

```
tosca "Tosca"
  takes-place-in  rome  ~ tosca-in-rome    # reified association

tosca-in-rome "The Setting of Tosca in Rome" ;
      bibref  """Nicassio, Susan Vandiver:
                 "Tosca's Rome: The Play and the Opera in Historical Perspective",
                 University of Chicago Press (Chicago, 2002)"""
```

The topic map itself can be reified by placing the "~" and topic identifier anywhere *except* within an assertion block or immediately following a verbose association. The topic map reifier is typically placed at the start of the document body:

```
%version ctm 1.0

[ ... directives ... ]

~ myTopicMap           # reifies the topic map

myTopicMap              # assertion block whose subject is the reified topic map
  "My Topic Map"
```

ISSUE: LH would like to define a CTM subject ("ctm:self" or "ctm:this") that is dynamically bound to the CTM instance the parser reads, rather than having what amounts to a blank (or omitted) subject, i.e.

```
ctm:self "This Topic Map";
  author nn ;
  descr "This is a topic map about...".
```

5.3.12 Verbose associations

ISSUE: Some issues remain relating to the terminology of associations. The text of this draft and the EBNF are currently inconsistent with respect to each other.

Associations as expressed within an assertion block rely on templates for information about role types. This permits a compact syntax very similar to that of names and occurrences (especially in the case of binary associations); however, it does not support the full generality of [TMDM]. In particular, it restricts associations of a given type to a single “signature”, or configuration of role types.

In order permit the expression of associations that do not conform to a template, CTM offers a verbose notation whose general form is

```
assoc-type( r-type-1 r-player-1 reifier? ;
            r-type-2 r-player-2 reifier? ;
            ... ;
            r-type-n r-player-n reifier? ) scope? reifier?
```

i.e., the topic identifier of an association type is followed by semicolon delimited pairs of parameters, which are enclosed within a single pair of parentheses; the parameter pairs consist of role type and role player. The following examples show the verbose form of the example associations given in section 5.3.2.5:

```
bio:born-in( person puccini ; place lucca ) .
bio:died-in( person puccini ; place brussels ) .

pupil-of( pupil puccini ; teacher ponchielli ) .
pupil-of( pupil puccini ; teacher bazzini ) .
pupil-of( pupil puccini ; teacher angeloni ) .

killed-by( victim scarpia ; perpetrator tosca ; cause stabbing ) .

unfinished( work turandot ) .
```

Scope on a verbose association is expressed using the "@" character followed by a space-delimited list of topic identifiers:

```
pupil-of( pupil puccini; teacher angeloni ) @budden .
```

Reification of associations and association roles is expressed using the "~" followed by a topic identifier for the reifying topic immediately after the construct in question:

```
# reify the relationship between Puccini and Angeloni:
pupil-of( pupil puccini; teacher angeloni ) @budden ~ puccini-angeloni .

# reify the role played by Puccini in that relationship:
pupil-of( pupil puccini ~ puccini-pupil-role ; teacher angeloni ) @budden .
```

6. Limitations

The following constructs, which are allowed by the [TMDM] are not supported by CTM:

- Multiple item identifiers on topics
- Item identifiers for topic maps, names, variants, occurrences, associations, and association roles.

ISSUE: Should the reasons for not supporting these constructs be given?

7. Conformance

A CTM document conforms to this International Standard provided it:

- Conforms to the grammar in Annex A.
- Is deserializable according to the procedure defined in Annex B without causing any errors or violating any data model constraints.

An CTM processor conforms to this International Standard provided it meets all the requirements given below.

- The CTM processor shall reject any input which is not a conforming CTM document.

- The CTM processor shall produce a representation that is isomorphic to the data model instance created by the procedure given in Annex B for all CTM documents.

Annex A (normative)

Formal language specification

This annex contains the formal language specification for CTM, expressed using Extended Backus-Naur Format (EBNF) as defined in [EBNF].

```
topicmap            ::= version-directive? encoding-directive?
                        (prefix-directive
                         | template
                         | assertion-block
                         | association
                         | comment)*

version-directive   ::= '%version ctm 1.0'
prefix-directive    ::= '%prefix' ID namespace
namespace           ::= iri

topic-identifier    ::= local-id
                        | subject-identifier
                        | subject-locator
                        | qname
local-id            ::= ID
subject-identifier  ::= iri-reference
subject-locator     ::= '=' iri
iri-reference       ::= iri | qname
qname               ::= ID ':' ID

assertion-block     ::= subject (assertion (";" assertion)*)? block-end
subject             ::= topic-identifier
block-end           ::= "." | EOL{2}
assertion           ::= (subject-identifier
                         | subject-locator
                         | name
                         | occurrence
                         | tpl-expansion)

name                ::= ((name-flag topic-identifier) | name-type)? name-value (',' name-value)*
name-flag           ::= '%name'
name-type           ::= topic-identifier   # Constraint:  The topic-identifier must be
                                           # defined as a name type via a name-template
name-value          ::= string variant* scope? reifier?
variant             ::= '(' datatyped-value scope reifier? ')'

occurrence          ::= occ-flag? occ-type occ-value (',' occ-value)*
occ-flag            ::= '%occ'
occ-type            ::= topic-identifier
occ-value           ::= datatyped-value scope? reifier?

tpl-expansion       ::= template-id tpl-expansion-body
template-id         ::= ID | iri-reference
tpl-expansion-body  ::= topic-identifier ( (',' | WS+) topic-identifier)*

association         ::= assoc-type '(' role (';' role)* ')' scope? reifier?
assoc-type          ::= topic-identifier
role                ::= role-type role-player reifier? (WS+ role-player reifier?)*
role-type           ::= topic-identifier
role-player         ::= topic-identifier

template            ::= (assoc-template
                         | occ-template
                         | name-template) block-end
assoc-template      ::= '%assoc' topic-identifier (WS topic-identifier)* scope?
occ-template        ::= '%occ'  topic-identifier datatype? scope?
name-template       ::= '%name' topic-identifier scope?

reifier             ::= '~' topic-identifier
scope               ::= '@' topic-identifier (WS+ topic-identifier)*

# NOTE: IRIs must conform to the specification given in [IRI] and its successors.
iri                 ::= ('<' [^ \r\n>]+ '>') | http-iri
http-iri            ::= 'http://' [^ \r\n]+
# NOTE: These datatyped values are identical to those defined in [XSD-2].
datatyped-value     ::= iri
                        | integer
                        | decimal
                        | date
                        | string datatype?
datatype            ::= '^^' iri-reference
integer             ::= ('-' | '+') ? [0-9]+
decimal             ::= ('-' | '+')? ( [0-9]+ '.' [0-9]* | '.' ([0-9])+)
date                ::= '-'? [0-9]{4} [0-9]* '-' (0|1)[0-9] '-' [0-3][0-9]
string              ::= quoted-string | triple-quoted-string
quoted-string       ::= '"' ([^\r\n"]* | '\"'*) '"'
triple-quoted-string::= '"""' .* '"""'

comment             ::= multiline-comment | one-line-comment
multiline-comment   ::= '/*' .* '*/'
one-line-comment    ::= '#' [^\r\n] EOL?

WS                  ::= [ \t\f]
EOL                 ::= \r|\n|\r\n

ID                  ::= (Letter | '_') (NameChar)*

# NOTE: The production for NameChar is identical to that in [XML]
#       except that colons are not permitted.
NameChar            ::= Letter
                        | Digit
                        | '.'
                        | '-'
                        | '_'
                        | CombiningChar
                        | Extender
Letter              ::= BaseChar | Ideographic
BaseChar            ::=   [#x0041-#x005A] | [#x0061-#x007A] | [#x00C0-#x00D6] | [#x00D8-#x00F6]
                        | [#x00F8-#x00FF] | [#x0100-#x0131] | [#x0134-#x013E] | [#x0141-#x0148]
                        | [#x014A-#x017E] | [#x0180-#x01C3] | [#x01CD-#x01F0] | [#x01F4-#x01F5]
                        | [#x01FA-#x0217] | [#x0250-#x02A8] | [#x02BB-#x02C1] | #x0386
                        | [#x0388-#x038A] | #x038C | [#x038E-#x03A1] | [#x03A3-#x03CE]
                        | [#x03D0-#x03D6] | #x03DA | #x03DC | #x03DE | #x03E0 | [#x03E2-#x03F3]
                        | [#x0401-#x040C] | [#x040E-#x044F] | [#x0451-#x045C] | [#x045E-#x0481]
                        | [#x0490-#x04C4] | [#x04C7-#x04C8] | [#x04CB-#x04CC] | [#x04D0-#x04EB]
                        | [#x04EE-#x04F5] | [#x04F8-#x04F9] | [#x0531-#x0556] | #x0559
                        | [#x0561-#x0586] | [#x05D0-#x05EA] | [#x05F0-#x05F2] | [#x0621-#x063A]
                        | [#x0641-#x064A] | [#x0671-#x06B7] | [#x06BA-#x06BE] | [#x06C0-#x06CE]
                        | [#x06D0-#x06D3] | #x06D5 | [#x06E5-#x06E6] | [#x0905-#x0939] | #x093D
                        | [#x0958-#x0961] | [#x0985-#x098C] | [#x098F-#x0990] | [#x0993-#x09A8]
                        | [#x09AA-#x09B0] | #x09B2 | [#x09B6-#x09B9] | [#x09DC-#x09DD]
                        | [#x09DF-#x09E1] | [#x09F0-#x09F1] | [#x0A05-#x0A0A] | [#x0A0F-#x0A10]
                        | [#x0A13-#x0A28] | [#x0A2A-#x0A30] | [#x0A32-#x0A33] | [#x0A35-#x0A36]
                        | [#x0A38-#x0A39] | [#x0A59-#x0A5C] | #x0A5E | [#x0A72-#x0A74]
                        | [#x0A85-#x0A8B] | #x0A8D | [#x0A8F-#x0A91] | [#x0A93-#x0AA8]
                        | [#x0AAA-#x0AB0] | [#x0AB2-#x0AB3] | [#x0AB5-#x0AB9] | #x0ABD | #x0AE0
                        | [#x0B05-#x0B0C] | [#x0B0F-#x0B10] | [#x0B13-#x0B28] | [#x0B2A-#x0B30]
                        | [#x0B32-#x0B33] | [#x0B36-#x0B39] | #x0B3D | [#x0B5C-#x0B5D]
                        | [#x0B5F-#x0B61] | [#x0B85-#x0B8A] | [#x0B8E-#x0B90] | [#x0B92-#x0B95]
                        | [#x0B99-#x0B9A] | #x0B9C | [#x0B9E-#x0B9F] | [#x0BA3-#x0BA4]
                        | [#x0BA8-#x0BAA] | [#x0BAE-#x0BB5] | [#x0BB7-#x0BB9] | [#x0C05-#x0C0C]
                        | [#x0C0E-#x0C10] | [#x0C12-#x0C28] | [#x0C2A-#x0C33] | [#x0C35-#x0C39]
                        | [#x0C60-#x0C61] | [#x0C85-#x0C8C] | [#x0C8E-#x0C90] | [#x0C92-#x0CA8]
                        | [#x0CAA-#x0CB3] | [#x0CB5-#x0CB9] | #x0CDE | [#x0CE0-#x0CE1]
                        | [#x0D05-#x0D0C] | [#x0D0E-#x0D10] | [#x0D12-#x0D28] | [#x0D2A-#x0D39]
                        | [#x0D60-#x0D61] | [#x0E01-#x0E2E] | #x0E30 | [#x0E32-#x0E33]
                        | [#x0E40-#x0E45] | [#x0E81-#x0E82] | #x0E84 | [#x0E87-#x0E88] | #x0E8A
                        | #x0E8D | [#x0E94-#x0E97] | [#x0E99-#x0E9F] | [#x0EA1-#x0EA3] | #x0EA5
                        | #x0EA7 | [#x0EAA-#x0EAB] | [#x0EAD-#x0EAE] | #x0EB0 | [#x0EB2-#x0EB3]
                        | #x0EBD | [#x0EC0-#x0EC4] | [#x0F40-#x0F47] | [#x0F49-#x0F69]
                        | [#x10A0-#x10C5] | [#x10D0-#x10F6] | #x1100 | [#x1102-#x1103]
                        | [#x1105-#x1107] | #x1109 | [#x110B-#x110C] | [#x110E-#x1112] | #x113C
                        | #x113E | #x1140 | #x114C | #x114E | #x1150 | [#x1154-#x1155] | #x1159
                        | [#x115F-#x1161] | #x1163 | #x1165 | #x1167 | #x1169 | [#x116D-#x116E]
                        | [#x1172-#x1173] | #x1175 | #x119E | #x11A8 | #x11AB | [#x11AE-#x11AF]
                        | [#x11B7-#x11B8] | #x11BA | [#x11BC-#x11C2] | #x11EB | #x11F0 | #x11F9
                        | [#x1E00-#x1E9B] | [#x1EA0-#x1EF9] | [#x1F00-#x1F15] | [#x1F18-#x1F1D]
                        | [#x1F20-#x1F45] | [#x1F48-#x1F4D] | [#x1F50-#x1F57] | #x1F59 | #x1F5B
                        | #x1F5D | [#x1F5F-#x1F7D] | [#x1F80-#x1FB4] | [#x1FB6-#x1FBC] | #x1FBE
                        | [#x1FC2-#x1FC4] | [#x1FC6-#x1FCC] | [#x1FD0-#x1FD3] | [#x1FD6-#x1FDB]
                        | [#x1FE0-#x1FEC] | [#x1FF2-#x1FF4] | [#x1FF6-#x1FFC] | #x2126
                        | [#x212A-#x212B] | #x212E | [#x2180-#x2182] | [#x3041-#x3094]
                        | [#x30A1-#x30FA] | [#x3105-#x312C] | [#xAC00-#xD7A3]
Ideographic         ::=   [#x4E00-#x9FA5] | #x3007 | [#x3021-#x3029]
CombiningChar       ::=   [#x0300-#x0345] | [#x0360-#x0361] | [#x0483-#x0486] | [#x0591-#x05A1]
                        | [#x05A3-#x05B9] | [#x05BB-#x05BD] | #x05BF | [#x05C1-#x05C2] | #x05C4
                        | [#x064B-#x0652] | #x0670 | [#x06D6-#x06DC] | [#x06DD-#x06DF]
                        | [#x06E0-#x06E4] | [#x06E7-#x06E8] | [#x06EA-#x06ED] | [#x0901-#x0903]
                        | #x093C | [#x093E-#x094C] | #x094D | [#x0951-#x0954] | [#x0962-#x0963]
                        | [#x0981-#x0983] | #x09BC | #x09BE | #x09BF | [#x09C0-#x09C4]
                        | [#x09C7-#x09C8] | [#x09CB-#x09CD] | #x09D7 | [#x09E2-#x09E3] | #x0A02
                        | #x0A3C | #x0A3E | #x0A3F | [#x0A40-#x0A42] | [#x0A47-#x0A48]
                        | [#x0A4B-#x0A4D] | [#x0A70-#x0A71] | [#x0A81-#x0A83] | #x0ABC
                        | [#x0ABE-#x0AC5] | [#x0AC7-#x0AC9] | [#x0ACB-#x0ACD] | [#x0B01-#x0B03]
                        | #x0B3C | [#x0B3E-#x0B43] | [#x0B47-#x0B48] | [#x0B4B-#x0B4D]
                        | [#x0B56-#x0B57] | [#x0B82-#x0B83] | [#x0BBE-#x0BC2] | [#x0BC6-#x0BC8]
                        | [#x0BCA-#x0BCD] | #x0BD7 | [#x0C01-#x0C03] | [#x0C3E-#x0C44]
                        | [#x0C46-#x0C48] | [#x0C4A-#x0C4D] | [#x0C55-#x0C56] | [#x0C82-#x0C83]
                        | [#x0CBE-#x0CC4] | [#x0CC6-#x0CC8] | [#x0CCA-#x0CCD] | [#x0CD5-#x0CD6]
                        | [#x0D02-#x0D03] | [#x0D3E-#x0D43] | [#x0D46-#x0D48] | [#x0D4A-#x0D4D]
                        | #x0D57 | #x0E31 | [#x0E34-#x0E3A] | [#x0E47-#x0E4E] | #x0EB1
                        | [#x0EB4-#x0EB9] | [#x0EBB-#x0EBC] | [#x0EC8-#x0ECD] | [#x0F18-#x0F19]
                        | #x0F35 | #x0F37 | #x0F39 | #x0F3E | #x0F3F | [#x0F71-#x0F84]
                        | [#x0F86-#x0F8B] | [#x0F90-#x0F95] | #x0F97 | [#x0F99-#x0FAD]
                        | [#x0FB1-#x0FB7] | #x0FB9 | [#x20D0-#x20DC] | #x20E1 | [#x302A-#x302F]
                        | #x3099 | #x309A
Digit               ::=   [#x0030-#x0039] | [#x0660-#x0669] | [#x06F0-#x06F9] | [#x0966-#x096F]
                        | [#x09E6-#x09EF] | [#x0A66-#x0A6F] | [#x0AE6-#x0AEF] | [#x0B66-#x0B6F]
                        | [#x0BE7-#x0BEF] | [#x0C66-#x0C6F] | [#x0CE6-#x0CEF] | [#x0D66-#x0D6F]
                        | [#x0E50-#x0E59] | [#x0ED0-#x0ED9] | [#x0F20-#x0F29]
Extender            ::=    #x00B7 | #x02D0 | #x02D1 | #x0387 | #x0640 | #x0E46 | #x0EC6 | #x3005
                        | [#x3031-#x3035] | [#x309D-#x309E] | [#x30FC-#x30FE]
```

ISSUE: Terminology of 'tpl-expansion', etc. inconsistent with text

ISSUE: Omitted role marker (%role) not in EBNF

ISSUE: Predefined templates (ISA and AKO) not in EBNF

ISSUE: Line continuation (for strings and everywhere)

ISSUE: Unicode escaping

ISSUE: Escaping of '"' (and '\'?)

Annex B (normative)

Deserialization

@@@TBD: Deserialization is probably best specified on the basis of (some of) the productions in the EBNF, many of which equate more-or-less to elements in the XTM syntax.

Annex C (informative)

Serialization

@@@TBD: Serialization advice.

ISSUE: Is this annex necessary and/or useful?