

Editors:	Heuer, Hopmans, Oh, Pepper
Status:	Editors' Working Draft
Version:	0.5
Date:	2006-09-15
Issues:	<ul style="list-style-type: none"> • Templates: What functionality, what syntax? Flags and markers: what syntax? • Clarify IRI terminology (especially RFC 3987 vs. xsd:anyURI regarding relative IRIs) • Topic map reification syntax? • Should local identifiers become part of the model (as item identifiers)?

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

ISO/IEC 13250-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information Technology, Subcommittee SC 34, Document Description and Processing Languages.

ISO/IEC 13250 consists of the following parts, under the general title Information Technology — Topic Maps:

- Part 1: Overview and Basic Concepts
- Part 2: Data Model
- Part 3: XML Syntax
- Part 4: Canonicalization
- Part 5: Reference Model
- Part 6: Compact Syntax
- Part 7: Graphical Notation

Introduction

CTM (Compact Topic Maps) 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 [\[13250-3\]](#) 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 [\[13250-2\]](#), care should be taken when using CTM as a basis for interchanging topic maps.

NOTE 1: The only feature of [\[13250-2\]](#) that is not supported is item identifiers on topic maps, names, variants, occurrences, associations, and association roles. (Item identifiers on topics *are* supported.)

This part of ISO/IEC 13250 should be read in conjunction with [\[13250-2\]](#) since the interpretation of the CTM syntax is defined through a mapping from the syntax to the data model there defined.

WD 13250-6: Topic Maps – Compact Syntax

1. Scope

This part of ISO/IEC 13250 defines a text-based notation for representing instances of the data model defined in [13250-2]. It also defines a mapping from this notation to the data model. The syntax is defined through an EBNF grammar using the notation defined in [XML]. More precision is provided through the mapping to the data model, which effectively also defines the interpretation of the syntax.

2. 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.

[13250-2]

ISO/IEC 13250-2:2006 Information technology – Topic Maps – Data Model

<http://www.jtc1sc34.org/repository/0696.pdf>

[13250-3]

ISO/IEC 13250-3:2006 Information technology – Topic Maps – XML Syntax (XTM)

<http://www.jtc1sc34.org/repository/FIXME.pdf>

[13250-4]

FCD 13250-4 Information technology – Topic Maps – Canonicalization

<http://www.jtc1sc34.org/repository/FIXME.pdf>

[IRI]

Internationalized Resource Identifiers (IRIs)

<http://www.ietf.org/rfc/rfc3987.txt>

[XML]

Extensible Markup Language (XML) 1.0 (Fourth Edition)

<http://www.w3.org/TR/2004/REC-xml-20060816>

[XMLN]

Namespaces in XML 1.0 (Second Edition)

<http://www.w3.org/TR/2006/REC-xml-names-20060816>

[XSD-2]

XML Schema Part 2: Datatypes

<http://www.w3.org/TR/xmlschema-2/>

3. Terms and definitions

For the purposes of this part of ISO/IEC 13250 the following terms and definitions apply, in addition to those defined in [13250-2]:

3.1

assertion

a statement or identifier assignment

3.2

assertion block

a set of assertions about one or more subjects

3.3

association template

a construct that allows role types to be specified once for all associations of a given type, thereby permitting the use of compact associations

3.4

compact association

a compact representation of an association in which role types are inferred from an association template

3.5

compact IRI

an IRI expressed as a tuple consisting of a prefix and a local part which are concatenated on deserialization

3.6

identifier assignment

the assignment of a subject identifier, subject locator, or item identifier to a subject

3.7

subject (of an assertion block)

the topic(s) to which identifiers are assigned and about which statements are made in an assertion block

4. Syntax description

4.1 About the syntax

The abbreviation CTM is often used to refer to the syntax defined in this part of ISO/IEC 13250. Its full name is Topic Maps Compact Syntax. The namespace for the CTM syntax is <http://www.topicmaps.org/ctm/>.

A CTM document is a text document that conforms to the CTM syntax. This clause defines the syntax of CTM documents using an EBNF grammar based on the notation described in [\[XML\]](#), and their semantics using prose describing the mapping from CTM documents to [\[13250-2\]](#). The full EBNF can be found in Annex A.

4.2 Deserialization

The process of exporting a topic map from an implementation's internal representation of the data model to an instance of a Topic Maps syntax is known as serialization. The opposite process, deserialization, is the process of building an instance of an implementation's internal representation of the data model from an instance of a Topic Maps syntax.

This clause defines how instances of the CTM syntax are deserialized into instances of the data model defined in [\[13250-2\]](#). Serialization is only implicitly defined, but implementations should guarantee that for any data model instance the CTM serialization produced by the implementation should when deserialized to a new data model instance produce one that has the same canonicalization as the original data model instance, according to [\[13250-4\]](#).

The input to the deserialization process is:

- A CTM document.

Deserialization is performed by processing each component of the document in document order. Components are defined in terms of text that matches a syntactic variable of the EBNF. For each component encountered the operations specified in the clause for the corresponding syntactic variable are performed.

NOTE 2: The interpretation of some components is described in connection with their parent or some other ancestor components.

Whenever a new information item is created, those of its properties which have set values are initialized to the empty set; all other properties are initialized to null.

4.3 Common syntactical constructs

4.3.1 White space

White space consists of one or more space (#x20) characters, carriage returns (#x0D), line feeds (#x0A), or tabs (#x09). It has no significance in a CTM document except inside strings.

4.3.2 COMMENT

A COMMENT component may occur anywhere in a CTM document where white-space is allowed and is declared as follows:

```
COMMENT          ::= ONE-LINE-COMMENT | MULTILINE-COMMENT
ONE-LINE-COMMENT ::= '#' [^\r\n] EOL?
MULTILINE-COMMENT ::= '/*' .* '*/'
```

There are two kinds of comment: one-line (marked by a '#' and terminated by the end of a line) and multiline (marked by '/*' and terminated by '*/').

Example 1:

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

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

Comments are ignored during deserialization.

4.3.3 TOPIC-ID

A TOPIC-ID component is used to identify a topic and is declared as follows:

```
TOPIC-ID          ::= (LOCAL-ID | ANY-IRI)
LOCAL-ID          ::= NCName
ANY-IRI           ::= (HTTP-IRI | DELIMITED-IRI | COMPACT-IRI)
# IRIs must conform to \[IRI\] and its successors.
HTTP-IRI          ::= 'http://' [^\r\n]+
DELIMITED-IRI    ::= '<' [^\r\n]+ '>'
COMPACT-IRI       ::= PREFIX '!' LOCAL-PART
PREFIX            ::= NCName
LOCAL-PART        ::= (NCNameChar)+
# NCName          ::= see \[XMLN\]
# NCNameChar      ::= see \[XMLN\]
```

NOTE 3: NCName and NCNameChar are the same as Name and NameChar, respectively, in [\[XML\]](#), except that they cannot contain a colon (":").

There are three kinds of topic ID that have significance in terms of the Topic Maps Data Model, all of which are interpreted as IRIs:

- An HTTP-IRI is an IRI that uses the HTTP scheme; it may be used in a CTM document without delimiters.
- A DELIMITED-IRI can be any valid IRI; it is delimited by "<" and ">".

- A COMPACT-IRI is a tuple that expands to an IRI through the simple concatenation of its two components, the PREFIX and the LOCAL-PART.

A local ID is an editing convenience for the author of a CTM document. Multiple occurrences of the same local ID give rise to a single topic item. Other than this, a local ID has no significance in terms of the Topic Maps Data Model and all information about it is lost once the CTM document has been parsed into an internal representation of the Topic Maps data model.

It is an error for the same local ID to be bound as a prefix to more than one IRI, used as an identifier for more than than one template, used as both prefix and template identifier, or used as both prefix and subject of an assertion block.

NOTE 4: The following table provides examples of the four kinds of topic ID:

Name	Form	Example
LOCAL-ID	NCName	puccini
COMPACT-IRI	PREFIX "!" LOCAL-PART	w!Puccini
HTTP-IRI	http://...	http://en.wikipedia.org/wiki/Puccini
DELIMITED-IRI	"<" IRI ">"	<http://en.wikipedia.org/wiki/Puccini>
		<urn:x-myns:music:puccini>
		<ftp://example.com/opera/tosca/synopsis.txt>

During deserialization a TOPIC-ID results in the creation of a topic item (unless one already exists that corresponds to that LOCAL-ID or IRI). An HTTP-IRI, DELIMITED-IRI or COMPACT-IRI gives rise to a locator which is added to the [subject identifiers] property of the topic item (in the case of COMPACT-IRI, after concatenating its two parts).

4.3.4 DATATYPED-VALUE

A DATATYPED-VALUE component is used to specify the value of an occurrence or variant name and is declared as follows:

```

DATATYPED-VALUE ::= ANY-IRI
                  | INTEGER
                  | DECIMAL
                  | DATE
                  | STRING DATATYPE?

DATATYPE ::= '^' ANY-IRI

INTEGER ::= ('-' | '+') ? [0-9]+

DECIMAL ::= ('-' | '+') ? ( [0-9]+ '.' [0-9]* | '.' ([0-9]+) )

DATE ::= '-' ? [0-9] [0-9] [0-9] [0-9] [0-9] * '-' (0|1) [0-9] '-' [0-3] [0-9]
        TIMEZONE?

TIMEZONE ::= (('+' | '-') [0-9] [0-9] ':' [0-9] [0-9] ) | 'Z'

STRING ::= QUOTED-STRING | TRIPLE-QUOTED-STRING

QUOTED-STRING ::= '"' ('\'' | '\\\'' | [^"\\])* '"'

TRIPLE-QUOTED-STRING ::= '"""' .* '"""'

```

CTM supports the datatypes of [\[XSD-2\]](#). The following datatypes are "built-in", i.e., 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.:

Example 3:

```

"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, or as compact IRIs, e.g.:

```
<urn:x-myns:music:puccini>  
<ftp://example.org/opera/tosca/synopsis.txt>  
<http://en.wikipedia.org/wiki/Puccini>  
http://en.wikipedia.org/wiki/Puccini  
wiki!Puccini
```

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.:

Example 4:

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

4.4 TOPICMAP

The TOPICMAP component is the equivalent of the CTM document. It acts as a container for the topic map but has no further significance. It is declared as follows:

```
TOPICMAP ::= HEADER BODY
```

During deserialization the TOPICMAP component causes a topic map item to be created.

4.5 HEADER

The HEADER component contains required and optional directives and templates and is declared as follows:

```
HEADER ::= ENCODING? VERSION? TOPICMAP-REIFIER?  
(PREFIX-DECL | TEMPLATE | MERGEMAP | INCLUDE )*
```

The HEADER component has no effect during deserialization.

4.5.1 ENCODING

The ENCODING component specifies the character encoding used by the document and is declared as follows:

```
ENCODING ::= '%encoding' STRING
```

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.

Example 5:

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

During deserialization the ENCODING component determines how the parser interprets the remainder of the CTM document.

@@@TBD: Make the preceding paragraph say something intelligent.

4.5.2 VERSION

The VERSION component identifies the CTM document as being in CTM syntax and states the version number, which is currently "1.0". It is declared as follows:

```
VERSION ::= '%version' 'ctm' '1.0'
```

The VERSION component tells the parser which version of the CTM syntax to use during deserialization. Currently the only legal version is 1.0, as defined by this part of ISO/IEC 13250.

4.5.3 TOPICMAP-REIFIER

The TOPICMAP-REIFIER component specifies an identifier for the topic which reifies the topic map represented by the CTM document and is declared as follows:

```
TOPICMAP-REIFIER ::= '%reifier' TOPIC-ID
```

During deserialization a topic item is created corresponding to TOPIC-ID and set as the value of the [reifier] property of the topic map item.

4.5.4 PREFIX-DECL

A PREFIX-DECL component associates an identifier with an XML namespace and is declared as follows:

```
PREFIX-DECL ::= '%prefix' LOCAL-ID (DELIMITED-IRI | HTTP-IRI)
```

Example 6:

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

During deserialization the PREFIX-DECL component binds the local ID to the IRI.

4.5.5 TEMPLATE

CTM allows templating of names, occurrences, and associations. The use of templates permits CTM documents to achieve greater compactness, readability and consistency. A TEMPLATE component is declared as follows:

```
TEMPLATE ::= NAME-TEMPLATE |
           OCCURRENCE-TEMPLATE |
           ASSOCIATION-TEMPLATE
```

The TEMPLATE component has no effect during deserialization.

4.5.5.1 NAME-TEMPLATE

A NAME-TEMPLATE component is used to assert that a certain assertion type is to be interpreted by the CTM parser as a name type. It is declared as follows

```
NAME-TEMPLATE ::= '%name' TOPIC-ID
```

Example 7:

```
%name foaf!name # foaf!name is a name type
```

The NAME-TEMPLATE component has no direct effect during deserialization.

4.5.5.2 OCCURRENCE-TEMPLATE

An OCCURRENCE-TEMPLATE component 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 of all such occurrences. It is declared as follows

```
OCCURRENCE-TEMPLATE ::= '%occur' TOPIC-ID DATATYPE?
```

Example 8:

```
%occur bio!birthYear ^^xsd:gYear
```

The OCCURRENCE-TEMPLATE component has no direct effect during deserialization.

4.5.5.3 ASSOCIATION-TEMPLATE

An ASSOCIATION-TEMPLATE component is used to specify the role types to be inferred for a compact association. It consists of a role type (specifying the role played by the subject of the assertion block), followed by the association type, and zero or more additional role types. It is declared as follows:

```
ASSOCIATION-TEMPLATE ::= '%assoc' TOPIC-ID TOPIC-ID ':' TOPIC-ID*
```

Example 9:

```
%assoc person born-in: place # binary
%assoc victim killed-by: killer cause # ternary
%assoc work unfinished: # unary
```

The first TOPIC-ID specifies the type of role played by the subject(s) of the assertion block in which an instance of the association type specified by the second TOPIC-ID occurs. Further TOPIC-IDs (if any) specify the role types of other role players in associations of the given type, and the order in which they will be assigned to role players in a compact association construct.

NOTE 5: Thus, in the example above, 'born-in' is a binary association type whose corresponding role types are 'person' (played by the subject(s) of the assertion block in which a compact association of type 'born-in' occurs) and 'place' (played by the other role player).

Similarly, 'killed-by' is a ternary association type whose corresponding role types are 'victim' (played by the subject(s) of the assertion block in which a compact association of type 'killed-by' occurs) and 'killer' and 'cause', respectively (played by the other role players *in that order*).

Finally, 'unfinished' is a unary association type whose only corresponding role type is 'work' (played by the subject(s) of the assertion block in which a compact association of type 'unfinished' occurs).

For an extended example explaining how association templates are applied in practice, see the section [COMPACT-ASSOCIATIONS](#), below.

During deserialization a topic item is created for each TOPIC-ID in the template.

4.5.6 MERGEMAP

The MERGEMAP component is used to merge an external topic map into the topic map produced by deserializing the CTM topic map and is declared as follows:

```
MERGEMAP ::= '%mergemap' (ANY-IRI | FILEREF) NOTATION?  
FILEREF ::= STRING  
NOTATION ::= ANY-IRI
```

The topic map to be merged can be referenced by an IRI or a file reference (which may be absolute or relative). The topic map can be in any syntax, which must be declared (using the NOTATION component) if it is not CTM.

NOTE 6: This International Standard defines the following identifiers for Topic Maps syntaxes:

CTM

<http://www.topicmaps.org/ctm/>

XTM

<http://www.topicmaps.org/xtm/>

A new namespace is established for local IDs in the merged topic map (including prefixes). Declarations in the header of the calling topic map do not apply to the merged topic map, and declarations in the header of the merged topic map have no effect on the remainder of the calling topic map.

During deserialization the MERGEMAP component causes the referenced topic map to be immediately deserialized into a data model instance. The new data model instance (B) is then merged into the current one (A) by:

- Adding all topic items in B's [topics] property to A's [topics] property.
- Adding all association items in B's [associations] property to A's [associations] property.

NOTE 7: Adding topics and associations to A may trigger further merges, as described in [\[13250-2\]](#).

4.5.7 INCLUDE

The INCLUDE component is used to include another CTM document into the current CTM document and is declared as follows:

```
INCLUDE ::= '%include' (ANY-IRI | FILEREF)
```

The document to be included can be referenced by an IRI or a file reference (which may be absolute or relative). No new namespace is established for local IDs. Declarations in the header of the calling document apply to the included document, and declarations in the header of the included document apply to the rest of the calling document.

During deserialization the INCLUDE component causes the referenced topic map to be immediately deserialized into the current data model instance as though its contents were part of the CTM document from which it is called.

4.6 BODY

The BODY component consists of assertion blocks and associations. It is declared as follows:

```
BODY ::= (ASSERTION-BLOCK | ASSOCIATION | ASSOCIATION-SET)*
```

The BODY component has no effect during deserialization.

4.6.1 ASSERTION-BLOCK

An ASSERTION-BLOCK component consists of a set of assertions about one or more subjects and is declared as follows:

```
ASSERTION-BLOCK ::= SUBJECTS? '{' ASSERTIONS? '}'  
ASSERTIONS     ::= ASSERTION ( ';' ASSERTION )* ';'?
```

Assertions are terminated by semicolons; however, the semicolon following the final assertion is optional:

Example 10:

```
subject(s) {           # the subject(s) of all assertions in this block  
  assertion-1 ;  
  assertion-2 ;  
  [...]   
  assertion-n ; # the final semicolon is optional  
}
```

White space is not significant; therefore the following examples are also valid:

Example 11:

```
# multiple assertions on one line:  
subject(s) {  
  assertion-1; assertion-2; assertion-3; assertion-4  
}  
  
# whole assertion block on one line:  
subject(s) {assertion-1}  
  
# multiple assertion blocks on one line:  
subject(s) {assertion-1} subject(s)2 {assertion-2} subject(s)3 {assertion-3}
```

The ASSERTION-BLOCK component has no effect during deserialization.

4.6.1.1 SUBJECTS

A SUBJECTS component is a comma-separated list of local identifiers for the topics to which the assertions in an assertion block apply. It is declared as follows:

```
SUBJECTS ::= SUBJECT ( ',' SUBJECT ) *  
SUBJECT  ::= LOCAL-ID
```

```
puccini {  
  [assertions]  
}  
tosca, butterfly, turandot {  
  [assertions]           # assertions are assigned to all three subjects  
}  
{  
  [assertions]           # assertion block with an anonymous subject  
}
```

During deserialization, one topic item is created (unless one already exists) for each LOCAL-ID. If the SUBJECT component is empty, a single topic item is created and, if the assertion block does not contain at least one identifier assignment, an item identifier is generated for the topic item.

The assertions result in statements and/or identifiers being created for each of those topic items.

4.6.2 ASSERTION

An ASSERTION component consists of either one or more statements (i.e., names, occurrences, or associations), or one or more identifier assignments (i.e., assignments of a subject-identifier, subject-locator, or item identifier). It is declared as follows:

```
ASSERTION ::= SUBJECT-IDENTIFIERS | SUBJECT-LOCATORS | ITEM-IDENTIFIERS
           | NAMES | OCCURRENCES | COMPACT-ASSOCIATIONS
```

The specific form of each kind of assertion is described below.

NOTE 8: All assertions share the same general form, which can be summarized as follows:

```
type value SCOPE? REIFIER?
```

where *type* is understood as

- *either* the “kind” of identifier (i.e., subject identifier, subject locator, or item identifier),
- *or* the [type] property of the item(s) created during deserialization (i.e., informally, the 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 }
```

The SCOPE and REIFIER components of an assertion are described below.

The ASSERTION component has no effect during deserialization.

4.6.2.1 SUBJECT-IDENTIFIERS

A SUBJECT-IDENTIFIERS component is used to assign one or more subject identifiers to the subject(s) of an assertion block and is declared as follows:

```
SUBJECT-IDENTIFIERS ::= 'sid' ':' SUBJECT-IDENTIFIER (',' SUBJECT-IDENTIFIER)*
SUBJECT-IDENTIFIER ::= ANY-IRI
```

Example 12:

```
%prefix w http://en.wikipedia.org/wiki/
composer {
  sid: w!Composer ;
  sid: http://www.kanzaki.com/ns/music#Composer ;
}
```

NOTE 9: The example above may be alternatively expressed, using a comma separated list of values, as

Example 13:

```
%prefix w http://en.wikipedia.org/wiki/
composer {
  sid: w!Composer , http://www.kanzaki.com/ns/music#Composer ;
}
```

During deserialization a locator is created for the IRI and added to the [subject identifiers] property of each topic item representing the subject of the assertion block. If this makes one topic item equal to another topic item, the two topic items are merged according to the procedure given in [\[13250-2\]](#).

4.6.2.2 SUBJECT-LOCATORS

A SUBJECT-LOCATORS component is used to assign one or more subject locators to the subject(s) of an assertion block and is declared as follows:

```
SUBJECT-LOCATORS ::= 'slo' ':' SUBJECT-LOCATOR (',' SUBJECT-LOCATOR)*
SUBJECT-LOCATOR ::= ANY-IRI
```

Example 14:

```
csgp-homepage {
  slo: http://www.puccini.it/
}
```

During deserialization a locator is created for the IRI and added to the [subject locators] property of each topic item representing the subject of the assertion block. If this makes one topic item equal to another topic item, the two topic items are merged according to the procedure given in [\[13250-2\]](#).

4.6.2.3 ITEM-IDENTIFIERS

A ITEM-IDENTIFIERS component is used to assign one or more item identifiers to the subject(s) of an assertion block and is declared as follows:

```
ITEM-IDENTIFIERS ::= 'iid' ':' ITEM-IDENTIFIER (',' ITEM-IDENTIFIER)*
ITEM-IDENTIFIER ::= ANY-IRI
```

Example 15:

```
puccini {
  iid: file:/usr/topicmaps/opera.ltm#composer
}
```

During deserialization a locator is created for the IRI and added to the [item identifiers] property of each topic item representing the subject of the assertion block. If this makes one topic item equal to another topic item, the two topic items are merged according to the procedure given in [\[13250-2\]](#).

4.6.2.4 NAMES

A NAMES component is used to assign one or more names to the subject(s) of an assertion block and is declared as follows:

```
NAMES ::= (NAME-FLAG? NAME-TYPE ':')? NAME-VALUE (',' NAME-VALUE)*
NAME-FLAG ::= '%name'
NAME-TYPE ::= TOPIC-ID
NAME-VALUE ::= STRING VARIANT* SCOPE? REIFIER?
VARIANT ::= '(' DATATYPED-VALUE SCOPE REIFIER? ')'
```

Example 16:

```

%prefix foaf http://xmlns.com/foaf/0.1/
%name foaf!name
puccini {
    "Puccini, Giacomo" ; # default name type assumed
    foaf!name: "Giacomo Puccini" ; # name type specified
    %name foaf!title: "Maestro" ; # force interpretation as name type
} # despite absence of a name template

boito {
    "Boito, Arrigo"
    ( "boito, arrigo" @sort ) # variant in the scope 'sort'
}

```

NOTE 10: Name flags are intended for use in the context of topic map fragments where templates would be overly verbose. Otherwise the use of templates is encouraged, especially when authoring larger topic maps, in order to ensure consistency and conciseness.

During deserialization an assertion is interpreted as a topic name if any of the following conditions holds true:

1. the name type is omitted (i.e., the first token in the assertion is a string);
2. a corresponding name template exists; or
3. the name flag is present.

Each NAME component causes a topic name item to be created and added to the [topic names] property of each topic item representing the subject of the assertion. The [value] property of each topic name item is set to the string contained in the NAME-VALUE component.

If the name type is present it produces a topic item which is set as the value of the [type] property of the topic name item. Otherwise the [type] property of the topic name item is set to the topic item whose [subject identifiers] property contains "http://psi.topicmaps.org/iso13250/model/topic-name"; if no such topic item exists, one is created.

SCOPE and REIFIER components, if present, are processed as described below.

Each VARIANT component causes a variant item to be created and added to the [variants] property of the corresponding topic name item. The [value] and [datatype] properties of the variant item are set by the DATATYPED-VALUE component, and the [scope] property is set to the union of the topic items created by the variant's SCOPE subcomponent and the parent name's SCOPE subcomponent.

4.6.2.5 OCCURRENCES

An OCCURRENCES component is used to assign one or more occurrences to the subject(s) of an assertion block and is declared as follows:

```

OCCURRENCES ::= OCC-FLAG? OCC-TYPE ':' OCC-VALUE (',' OCC-VALUE)*
OCC-FLAG    ::= '%occur'
OCC-TYPE    ::= TOPIC-ID
OCC-VALUE   ::= DATATYPED-VALUE SCOPE? REIFIER?

```

Example 17:

```

puccini {
    article      : http://en.wikipedia.org/wiki/Giacomo_Puccini , # IRI
                  http://it.wikipedia.org/wiki/Giacomo_Puccini ; #
    description  : "The greatest of the verismo composers" ; # string
    date-of-birth : 1858-12-22 ; # date
    bibref       : ""Budden, Julian: "Puccini: His Life and Works", # string
                  Oxford University Press (Oxford, 2002)"" ;
    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
}

```

NOTE 11: Occurrence flags are intended for use 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 consistency and conciseness.

During deserialization an assertion is interpreted as an occurrence if any of the following conditions holds true:

1. a corresponding occurrence template exists; or
2. the occurrence flag is present; or
3. no corresponding name template exists and no name flag is present.

Each OCCURRENCE component causes an occurrence item to be created and added to the [occurrences] property of each topic item representing the subject of the assertion. The [value] property of each topic occurrence item is set to the contents of the VALUE component and the [datatype] property is set to the IRI corresponding to the datatype.

The occurrence type produces a topic item which is set as the value of the [type] property of the occurrence item.

SCOPE and REIFIER components, if present, are processed as described below.

4.6.2.6 COMPACT-ASSOCIATIONS

A COMPACT-ASSOCIATIONS component is used to assert one or more associations in which the subject(s) of an assertion block play roles. It is declared as follows:

```
COMPACT-ASSOCIATIONS ::= ASSOC-TYPE ':' OTHER-ROLES (',' OTHER-ROLES)*
ASSOC-TYPE           ::= TOPIC-ID
OTHER-ROLES          ::= ROLE-PLAYER* SCOPE? REIFIER?
ROLE-PLAYER          ::= TOPIC-ID | NULL-ROLE
NULL-ROLE            ::= '%null'
```

The interpretation of a compact association is governed by an association template, as described below, and a compact association will *only* be interpreted as such if there exists a corresponding template.

NOTE 12: In the absence of an association template, constructs intended to represent binary compact associations will be interpreted as external occurrences, while compact associations of other arities (unary, ternary, etc.) will provoke parsing errors.

Example 18:

```
%prefix bio http://psi.ontopia.net/biography/#
%assoc work unfinished: # unary
%assoc person born-in: place # binary
%assoc pupil pupil-of: teacher # binary
%assoc victim killed-by: killer cause # ternary
%assoc killer kills: victim cause # ternary

killed-by, kills {
  sid: bio!killed-by ; # use of compact IRI (see 4.3.3)
}
turandot {
  unfinished: ; # unary association
}
puccini {
  born-in: lucca ; # binary association
  pupil-of: ponchielli, bazzini, angeloni ; # three binary associations
}
floria {
  kills: scarpia stabbing, floria jumping ; # two ternary associations
}
mario {
  killed-by: %null shooting ; # association with omitted
} # role player
```

NOTE 13: In the example above there are association templates for 'unfinished', 'born-in', 'pupil-of', 'killed-by', and 'kills', respectively. They all specify role types for the subject(s) of assertion blocks in which they occur (i.e., 'work', 'person', 'pupil', 'victim', and 'killer', respectively).

No further role types are specified for 'unfinished', so it is a unary association type. One additional role type is specified for each of 'born-in' ('place') and 'pupil-of' ('teacher'), so these are binary association types. Two additional role types are specified for each of 'killed-by' and 'kills', so they are ternary association types.

In fact, the local IDs 'killed-by' and 'kills' actually refer to the one and the same association type, which has the subject identifier 'bio!killed-by'. The 'killed-by' and 'kills' templates thus provide two alternative ways of expressing compact associations of the same type: one in which the role played by the subject(s) of the assertion block is 'victim' and one in which it is 'killer'. The assertion blocks whose subjects are 'floria' and 'mario' exemplify each of these.

Roles that are specified by the template but omitted in the assertion, can be indicated through the null role marker, "%null". In the example above, the role of 'killer' is omitted in the assertion block whose subject is 'mario' and the role player 'shooting' is assigned the role type 'cause' instead of 'killer'.

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

Association type	Role #1		Role #2		Role #3	
	role type	role player	role type	role player	role type	role player
unfinished	work	turandot				
born-in	person	puccini	place	lucca		
pupil-of	pupil	puccini	teacher	ponchielli		
pupil-of	pupil	puccini	teacher	bassani		
pupil-of	pupil	puccini	teacher	angeloni		
kills	killer	floria	victim	scarpia	cause	stabbing
kills	killer	floria	victim	floria	cause	jumping
killed-by	victim	mario			cause	shooting

For the full form of these associations, see the section [ASSOCIATION](#), below.

CTM has a built-in association template called 'isa' which is used to represent type-instance relationships using compact associations. The following template declaration and identifier assignment must be assumed by the CTM parser:

```
%prefix tm
%assoc tm!instance isa: tm!type
isa { sid: tm!type-instance }
```

The local identifier 'isa' can be used within an assertion block as follows:

Example 19:

```
puccini { isa: composer }
```

During deserialization each OTHER-ROLES component causes an association item to be created for each subject of the assertion block. For each of these association items, one association role item is created for each subject of the assertion block and added to the [roles] property. The [player] property of the association role item is set to the topic item that represents the subject in question.

Additional association role items are created for each TOPIC-ID in the OTHER-ROLES component in the same way as for the subjects of the assertion block.

The association type produces a topic item which is set as the value of the [type] property of each association item.

SCOPE and REIFIER components, if present, are processed as described below.

NOTE 14: It is not possible to reify association roles when using compact associations (only the association itself may be reified). In order to reify an association role, a freestanding association must be used as described in the section [ASSOCIATION](#), below.

4.6.3 SCOPE

The SCOPE component is used to express the scope of a statement and is declared as follows:

```
SCOPE ::= '@' TOPIC-ID+
```

Example 20:

```
opera {  
  
  # two names, one in the scope 'fi'  
  "Opera" ; "Oopera" @fi ;  
  
  # three description occurrences, in the scopes 'wordnet',  
  # 'wikipedia'+en', and 'wikipedia'+it', respectively  
  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  
  
}
```

During deserialization each TOPIC-ID in a SCOPE component gives rise to a topic item which is added to the [scope] property of the corresponding topic name, variant, occurrence, or association item. In the case of scoped variant items, the [scope] property consists of the union of the topics produced by the variant's SCOPE component and that of the NAME-VALUE component to which it belongs.

4.6.4 REIFIER

The REIFIER component signals the reification of the immediately preceding construct and is declared as follows:

```
REIFIER ::= '~' TOPIC-ID
```

The TOPIC-ID is the identifier of the reifying topic.

Example 21:

```
%assoc event takes-place-in place  
  
tosca {  
  "Tosca" ;  
  takes-place-in rome ~tosca-in-rome # reified association  
}  
tosca-in-rome { # assertion block for the reified association  
  "The Setting of Tosca in Rome" ; # its name  
  bibref # an occurrence of type 'bibref'  
  ""Nicassio, Susan Vandiver:  
  "Tosca's Rome: The Play and the Opera in Historical Perspective",  
  University of Chicago Press (Chicago, 2002)""  
}
```

During deserialization the TOPIC-ID of the REIFIER component gives rise to a topic item which is set as the value of the [reifier] property of the corresponding topic map, topic name, variant, occurrence, association, or association role item.

NOTE 15: Reification of the topic map represented by the CTM document as a whole is described in the section [TOPICMAP-REIFIER](#), above. Reification of association roles is described in the section [ASSOCIATION](#), below.

4.6.5 ASSOCIATION

An ASSOCIATION component is used to assert a freestanding association independently of assertion blocks and (optionally) association templates. It is declared as follows:

```
ASSOCIATION ::= ASSOC-TYPE '(' ROLES ')' SCOPE? REIFIER?
ROLES       ::= ROLE (' ' ROLE)* ','?
ROLE        ::= (ROLE-TYPE ':')? ROLE-PLAYER REIFIER?
ROLE-TYPE   ::= TOPIC-ID
```

When a role type is omitted, it is inferred from the corresponding association template. It is an error to omit a role type on an association for which no association template exists.

NOTE 16: Freestanding associations are intended for use in the following contexts: (1) when representing small topic map fragments (i.e., in situations where the templating mechanism would be unwieldy); (2) when the CTM document author wishes to keep associations of the same type together (rather than distributing them across multiple assertion blocks); (3) when there is a need to reify one or more association roles; and (4) in order to represent associations types that have multiple “signatures” (i.e., do not restrict themselves to a single configuration of role types).

Example 22:

```
unfinished( work: turandot )
born-in( person: puccini , place: lucca )
pupil-of( pupil: puccini , teacher: ponchielli )
pupil-of( pupil: puccini , teacher: bassani )
pupil-of( pupil: puccini , teacher: angeloni )
kills( killer: floria , victim: scarpia , cause: stabbing )
kills( killer: floria , victim: floria , cause: jumping )
killed-by( victim: mario , cause: shooting )
```

NOTE 17: The preceding example shows how the compact associations given in the example in the section [COMPACT-ASSOCIATIONS](#), above, would be represented in their full form. The following examples show the use of freestanding associations with templates, scope, and reification.

Example 23:

```
# freestanding associations that exploit a template in order to assign role types
%assoc pupil pupil-of teacher
pupil-of( puccini , ponchielli )
pupil-of( puccini , bassani )
pupil-of( puccini , angeloni )

# scoped association
pupil-of( pupil puccini; teacher angeloni ) @budden

# reified association (relationship between Puccini and Angeloni)
pupil-of( pupil: puccini, teacher: angeloni ) ~ puccini-angeloni

# reified association role (Puccini qua pupil of Angeloni)
pupil-of( pupil: puccini ~puccini-pupil-role , teacher angeloni )
```

4.6.6 ASSOCIATION-SET

An ASSOCIATION-SET component is used to assert a set of associations of the same type. It is declared as follows:

```

ASSOCIATION-SET ::= ASSOC-TYPE '[' BRANCH+ ']'
BRANCH          ::= ROOT-NODE CHILD-NODE+
ROOT-NODE       ::= TOPIC-ID
CHILD-NODE      ::= ('-'|'*') ( TOPIC-ID | CHILD-NODE )

```

Each child node starts with one or more hyphens (or asterisks); however, it must not start with more than one more hyphen (or asterisk) than the immediately preceding child (or root) node.

The interpretation of an association set is governed by a corresponding association template; it is an error if no such template exists or if the template does not specify exactly two role types. The first role type in the template is referred to as the "parent" role and the second role type is referred to as the "child" role.

Example 24:

```

%assoc super supertype-subtype sub
# "parent" role type is 'super'
# "child" role type is 'sub'

supertype-subtype [
    person
    - musician
    - - composer
    - - conductor
    - writer
    - - librettist
    - - playwright

    place
    - city
    - country
    - region
]

```

NOTE 18: Association sets are particularly useful for representing hierarchical structures, and this is reflected in the terminology used to describe them. However, their use is not restricted to hierarchies, as the following example shows:

Example 25:

```

%assoc composer composed work
# "parent" role type is 'composer'
# "child" role type is 'work'

composed [
    puccini * la-boheme * toscas * butterfly * turandot
    verdi * la-traviata * otello * aida * falstaff
    mascagni * cavalleria-rusticana * iris
]

```

During deserialization topic items are created for each ROOT-NODE and CHILD-NODE (unless corresponding items already exist). In addition, an association item and two association role items are created for each CHILD-NODE:

- The [type] property of the association item is set to the topic item corresponding to ASSOC-TYPE.
- The [type] properties of the association role items are set to the topic items representing the "parent" and "child" role types in the corresponding template, respectively.
- The [player] property of the "child" association role item is set to the topic item representing the CHILD-NODE.
- The [player] property of the "parent" association role item is set to the topic item representing the closest preceding CHILD-NODE or ROOT-NODE that starts with one fewer hyphen(s) than the CHILD-NODE that led to the creation of the association item.

5. 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 Clause 4 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 Clause 4 for all CTM documents.

Annex A (normative)

Formal language specification

This annex contains the formal language specification for CTM, expressed as an EBNF using the notation described in [\[XML\]](#).

```
COMMENT                ::= ONE-LINE-COMMENT | MULTILINE-COMMENT
ONE-LINE-COMMENT      ::= '#' [^\r\n] EOL?
MULTILINE-COMMENT    ::= '/*' .* '*/'

TOPIC-ID              ::= (LOCAL-ID | ANY-IRI)
LOCAL-ID              ::= NCName
ANY-IRI               ::= (HTTP-IRI | DELIMITED-IRI | COMPACT-IRI)
# IRIs must conform to [IRI] and its successors.
HTTP-IRI              ::= 'http://' [^\r\n]+
DELIMITED-IRI        ::= '<' [^\r\n]+ '>'
COMPACT-IRI           ::= PREFIX '!' LOCAL-PART
PREFIX                ::= NCName
LOCAL-PART            ::= (NCNameChar)+
# NCName               ::= see [XMLN]
# NCNameChar           ::= see [XMLN]

DATATYPED-VALUE      ::= ANY-IRI
                       | INTEGER
                       | DECIMAL
                       | DATE
                       | STRING DATATYPE?
DATATYPE              ::= '^' ANY-IRI
INTEGER               ::= ('-' | '+') ? [0-9]+
DECIMAL               ::= ('-' | '+')? ( [0-9]+ '.' [0-9]* | '.' ([0-9]+) )
DATE                  ::= '-'? [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] * '-' (0|1) [0-9] '-' [0-3] [0-9]
                       TIMEZONE?
TIMEZONE              ::= (('+' | '-') [0-9] [0-9] ':' [0-9] [0-9]) | 'Z'
STRING                ::= QUOTED-STRING | TRIPLE-QUOTED-STRING
QUOTED-STRING         ::= '"' ('\'' | '\\\'' | [^\"])* '"'
TRIPLE-QUOTED-STRING ::= '"""' .* '"""'

TOPICMAP              ::= HEADER BODY
HEADER                ::= ENCODING? VERSION? TOPICMAP-REIFIER?
                       (PREFIX-DECL | TEMPLATE | MERGEMAP | INCLUDE ) *

ENCODING              ::= '%encoding' STRING
VERSION               ::= '%version' 'ctm' '1.0'
TOPICMAP-REIFIER      ::= '%reifier' TOPIC-ID
PREFIX-DECL           ::= '%prefix' LOCAL-ID (DELIMITED-IRI | HTTP-IRI)
TEMPLATE              ::= NAME-TEMPLATE |
                       OCCURRENCE-TEMPLATE |
                       ASSOCIATION-TEMPLATE
NAME-TEMPLATE         ::= '%name' TOPIC-ID
OCCURRENCE-TEMPLATE  ::= '%occur' TOPIC-ID DATATYPE?
ASSOCIATION-TEMPLATE ::= '%assoc' TOPIC-ID TOPIC-ID ':' TOPIC-ID*
MERGEMAP              ::= '%mergemap' (ANY-IRI | FILEREF) NOTATION?
FILEREF               ::= STRING
NOTATION              ::= ANY-IRI
INCLUDE               ::= '%include' (ANY-IRI | FILEREF)

BODY                  ::= (ASSERTION-BLOCK | ASSOCIATION | ASSOCIATION-SET) *

ASSERTION-BLOCK      ::= SUBJECTS? '{' ASSERTIONS? '}'
ASSERTIONS           ::= ASSERTION ( ';' ASSERTION ) * ';' ?
SUBJECTS              ::= SUBJECT ( ',' SUBJECT ) *
SUBJECT              ::= LOCAL-ID
ASSERTION             ::= SUBJECT-IDENTIFIERS | SUBJECT-LOCATORS | ITEM-IDENTIFIERS
                       | NAMES | OCCURRENCES | COMPACT-ASSOCIATIONS

SUBJECT-IDENTIFIERS  ::= 'sid' ':' SUBJECT-IDENTIFIER (',' SUBJECT-IDENTIFIER) *
SUBJECT-IDENTIFIER   ::= ANY-IRI
SUBJECT-LOCATORS     ::= 'slo' ':' SUBJECT-LOCATOR (',' SUBJECT-LOCATOR) *
SUBJECT-LOCATOR      ::= ANY-IRI
ITEM-IDENTIFIERS     ::= 'iid' ':' ITEM-IDENTIFIER (',' ITEM-IDENTIFIER) *
ITEM-IDENTIFIER      ::= ANY-IRI
```

```

NAMES ::= (NAME-FLAG? NAME-TYPE ':' )? NAME-VALUE (',' NAME-VALUE)*
NAME-FLAG ::= '%name'
NAME-TYPE ::= TOPIC-ID
NAME-VALUE ::= STRING VARIANT* SCOPE? REIFIER?
VARIANT ::= '(' DATATYPED-VALUE SCOPE REIFIER? ')'

OCCURRENCES ::= OCC-FLAG? OCC-TYPE ':' OCC-VALUE (',' OCC-VALUE)*
OCC-FLAG ::= '%occur'
OCC-TYPE ::= TOPIC-ID
OCC-VALUE ::= DATATYPED-VALUE SCOPE? REIFIER?

COMPACT-ASSOCIATIONS ::= ASSOC-TYPE ':' OTHER-ROLES (',' OTHER-ROLES)*
ASSOC-TYPE ::= TOPIC-ID
OTHER-ROLES ::= ROLE-PLAYER* SCOPE? REIFIER?
ROLE-PLAYER ::= TOPIC-ID | NULL-ROLE
NULL-ROLE ::= '%null'

SCOPE ::= '@' TOPIC-ID+
REIFIER ::= '~' TOPIC-ID

ASSOCIATION ::= ASSOC-TYPE '(' ROLES ')' SCOPE? REIFIER?
ROLES ::= ROLE (',' ROLE)* ','?
ROLE ::= (ROLE-TYPE ':' )? ROLE-PLAYER REIFIER?
ROLE-TYPE ::= TOPIC-ID

ASSOCIATION-SET ::= ASSOC-TYPE '[' BRANCH+ ']'
BRANCH ::= ROOT-NODE CHILD-NODE+
ROOT-NODE ::= TOPIC-ID
CHILD-NODE ::= ('-'|'*') ( TOPIC-ID | CHILD-NODE )

```

Annex B (informative)

Extended example

```
%encoding "UTF-8"
%version ctm 1.0

%reifier operatm

# IRI prefix declarations (truncated for brevity)
%prefix dc http://purl.org/dc/elements/1.1/
%prefix mus http://psi.ontopia.net/music/
%prefix op http://psi.ontopia.net/opera/
%prefix bio http://psi.ontopia.net/biography/
%prefix wiki http://en.wikipedia.org/wiki/
%prefix snl http://www.ontopia.net/topicmaps/examples/opera/occurs/snl/
%prefix az http://www.azopera.com/learn/synopsis/
%prefix met http://www.metopera.org/synopses/

# association templates using local IDs (truncated for brevity)
%assoc person born-in: place
%assoc person died-in: place
%assoc work composed-by: composer
%assoc opera libretto-by: librettist
%assoc work premiere: place
%assoc result based-on: source
%assoc victim killed-by: perpetrator cause-of-death
%assoc subtype subtype-of: supertype
%assoc opera takes-place-in: place
%assoc work unfinished:

# association templates using subject IDs
%assoc resource dc!creator: value
%assoc resource dc!format: value
%assoc resource dc!language: value
%assoc resource dc!publisher: value
%assoc resource dc!subject: value
%assoc resource dc!type: value

# TOPIC MAP -----
operatm {
  "The Italian Opera Topic Map" ;
  "Opera TM" @short-name ;
  dc!date: "$Date: 2006/09/08 16:26:43 $" ;
  dc!description: "This topic map was originally written ..." ;
  dc!rights: "Copyright (c) 1999-2006, Steve Pepper" ;
  dc!creator: pepper ;
  dc!format: CTM ;
  dc!language: english ;
  dc!publisher: ontopia ;
  dc!subject: italy, opera ;
  dc!type: topicmap ;
  revision: "$Revision: 1.5 $" ;
}

# COMPOSERS -----
puccini {
  "Giacomo Puccini" ;
  "Puccini, Giacomo" @indexform ;
  sid: wiki!Puccini ;
  isa: composer ;
  born-in: lucca ;
  date-of-birth: 1858-12-22 ;
  died-in: brussels ;
  date-of-death: 1924-11-29 ;
  bibref: ""Budden, Julian: "Puccini: His Life and Works",
          Oxford University Press (Oxford, 2002)"" ;
  website: http://www.puccini.it @italian ;
}
```

```

# OPERAS -----
tosca {
  "Tosca" ;
  sid: wiki!Tosca ;
  isa: opera ;
  composed-by: puccini ;
  libretto-by: giacosa, illica ;
  premiere-date: 1900-01-14 ;
  premiere: teatro-costanzi ;
  based-on: la-tosca ;
  synopsis: az!tosca.shtml @az-opera,
            met!tosca.html @operanews ;
  article: wiki!Tosca @wikipedia,
            snl!tosca.htm @snl ;
  audio-recording: "7 47175 8" ;
  takes-place-in: roma ~ tosca-takes-place-in ;
}

madama-butterfly {
  "Madama Butterfly" ;
  sid: wiki!Madama_Butterfly ;
  isa: opera ;
  composed-by: puccini ;
  libretto-by: giacosa, illica ;
  premiere-date: 1904-02-17 ;
  premiere: la-scala ;
  based-on: madama-butterfly-src ~ madama-butterfly-based-on ;
  synopsis: az!butterfly.shtml @az-opera,
            met!madama.html @operanews ;
  article: wiki!Madama_Butterfly @wikipedia,
            snl!madama-butterfly.htm @snl ;
  audio-recording: "423 567-2" ;
  takes-place-in: nagasaki ;
}

# ONTOLOGY -----
composer {
  "Composer" ;
  "Skladatel" @czech ;
  "Komponist" @dutch german norwegian ;
  "Säveltäjä" @finnish ;
  "Compositeur" @french ;
  "Zeneszerző" @hungarian ;
  "Tónskáld" @icelandic ;
  "Compositore" @italian ;
  "???" @japanese ;
  "???" @korean ;
  "Kompozytor" @polish ;
  "Композитор" @russian ;
  "Cyfansoddwr" @welsh ;
  sid: mus!composer ;
  subtype-of: musician ;
}

opera {
  "Opera" ;
  "Ooppera" @finnish ;
  sid: mus!opera ;
  isa: artform ;
  subtype-of: musical-work, theatrical-work ;
  bibref: ""Boyden, Matthew: "Opera: The Rough Guide",
           Rough Guides (London, 1999)""",
           ""Keolker, James: "Last Acts : The Operas of Puccini and his Italian Contemporaries",
           Opera Companion Publications (Napa, 2000)""",
           ""Pogue, David and Scott Speck: "Opera for Dummies",
           IDG Books (Foster City, 1997)""";
  webpage: http://home.prcn.org/~pauld/opera/ @web ~ pauld-website ;
  website: http://opera.stanford.edu ;
}

/* assertion blocks with identifiers for many local IDs omitted for brevity */

/* TBD: extend example to illustrate further aspects of the syntax */

```