INTERNATIONAL STANDARD

ISO/IEC 13250-6

First edition
2010-11-15

Information technology — Topic Maps —

Part 6: Compact syntax

Technologies de l'information — Plans relatifs à des sujets —

Partie 6: Syntaxe compacte

Reference number
ISO/IEC 13250-6:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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*

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 defined in ISO/IEC 13250-3:2007 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, and maximum readability. CTM supports all constructs of ISO/IEC 13250-2, except item identifiers on constructs that are not topics.

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

Information Technology — Topic Maps —

Part 6:
Compact syntax

1 Scope

This part of ISO/IEC 13250 defines a text-based notation for representing instances of the data model defined in ISO/IEC 13250-2. It also defines a mapping from this notation to the data model. The syntax is defined through an Extended Backus-Naur Form (EBNF) grammar.

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.

NOTE Each of the following documents has a unique identifier that is used to cite the document in the text. The unique identifier consists of the part of the reference up to the first comma.

IANA-CHARSETS, *CHARACTER SETS*, Internet Assigned Numbers Authority, 14 May 2007, available at <http://www.iana.org/assignments/character-sets>

ISO/IEC 13250-2, *Information technology — Topic Maps — Part 2: Data model*

XSDT, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, 28 October 2004, available at <http://www.w3.org/TR/xmlschema-2/>

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, Internet Standards Track Specification, January 2005, available at <http://www.ietf.org/rfc/rfc3986.txt>

IETF RFC 3987, *Internationalized Resource Identifiers (IRIs)*, Internet Standards Track Specification, January 2005, available at <http://www.ietf.org/rfc/rfc3987.txt>

3 Syntax description

3.1 About the syntax

The acronym *CTM* is often used to refer to the syntax defined in this part of ISO/IEC 13250. Its full name is Compact Topic Maps Syntax.

This clause defines the syntax of CTM using an EBNF grammar based on the notation described in XML 1.0. It uses prose to define the mapping from CTM to ISO/IEC 13250-2. The full EBNF can be found in Annex A.

3.2 Deserialization

This clause defines how instances of the CTM syntax are deserialized into instances of the data model defined in ISO/IEC 13250-2. Serialization is only implicitly defined, but implementations should produce CTM serializations that, when deserialized to a new data model instance, produce a data model instance that has the same canonicalization as the original data model instance, according to ISO/IEC 13250-4: 2009.

Each CTM instance shall produce a valid data model instance according to ISO/IEC 13250-2.

The input to the deserialization process is:

- An optional character set name according to IANA-CHARSETS which specifies the encoding of the CTM instance.
- A byte stream which is converted into a character stream by the following steps:

    - If the optional input character set name is provided, the encoding is set to the provided value.
    - If the byte stream starts with EF BB BF (UTF-8 byte order mark), the encoding is set to "UTF-8".

      It is an error if the encoding is already set to another value due to the optional character set name.
    - If the next bytes in the stream contain the sequence 25 65 6E 63 6F 64 69 6E 67 (%encoding in ASCII), all following 09 or 20 bytes are skipped until the very first 22 byte is read. The following byte sequence until the next 22 (exclusive) byte is interpreted as character set name according to IANA-CHARSETS.

      It is an error if the encoding is already set to another value.
    - If the previous steps did not produce an encoding, the encoding shall be set to "UTF-8"
    - The byte stream is converted according to the encoding into Unicode (c.f.), where the optional byte sequence EF BB BF is removed previously from the start of the byte stream.

    This character stream is processed according to the grammar specified by this part of ISO/IEC 13250.
- An absolute IRI. This is the IRI from which the byte stream was retrieved, known as the document IRI. This IRI shall always be provided, as it is necessary in order to assign the item identifiers of the topic items created during deserialization. If the CTM instance was not read from any particular IRI the application is responsible for providing an IRI considered suitable.
- An absolute IRI which will be used to resolve wildcards against 3.3.8; this IRI is called "wildcard-iri". If the wildcard-iri is not provided, its value is set to the document IRI.
- A non-negative integer, called "wildcard-counter". If the wildcard-counter is not provided, the wildcard-counter is initialized with 0.

Deserialization is performed by processing each component of the CTM source 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.

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.

3.3 Common syntactical constructs

3.3.1 Whitespace

Whitespace consists of one or more space (#x20) characters, carriage returns, line feeds, or tabs.

Whitespace character are allowed everywhere to separate tokens (terminals and non-terminals).

3.3.2 Comments

Comments are fragments of the character stream which are ignored by a CTM processor. Comments are allowed where whitespace characters are allowed.

Multiline comments are delimited with #(and)# and may be nested.

Single line comments are introduced by a hash (#) and continue until the end of the current line, or until the end of the character stream, whichever comes first.

3.3.3 Creating IRIs from strings

The delimiters < and > are removed from iri-delimited; the resulting string may represent either an absolute or a relative IRI which shall met the the requirements of IETF RFC 3986 and IETF RFC 3987.

To create an IRI from a string, unescape the string by replacing %HH escape sequences with the characters they represent, and decode the resulting character sequence from UTF-8 to a sequence of abstract Unicode characters. The resulting string is turned into an absolute IRI by resolving it against the document IRI.

3.3.4 Creating IRIs from QNames

QNames are used to abbreviate IRIs. The syntax of QNames is as follows:

A QName causes a locator to be created. During deserialization, the IRI to which the prefix is bound is concatenated with the local part. The result of such a process is always an absolute IRI.

It is an error if the prefix has not been bound to an IRI as specified in 3.16.1.

```
%prefix isbn urn:isbn:

isbn:3-7026-4850-X isa book;
- "Das kleine Ich bin Ich".
```

3.3.5 IRI References

IRI references are either QNames or IRIs. They are interchangeable: Everywhere an IRI can be used a QName can also be used (provided the prefix is defined at that point).

3.3.6 Topic Identity

Topics are referenced by an item identifier, a subject identifier, or a subject locator.

During deserialization, one topic item is created for each topic-identity.

If the topic-identity is an identifier, a locator is created by concatenating the document IRI, a # character, and the value of the identifier. The locator is added to the [item identifiers] property of the topic item.

If the topic-identity is specified by a subject identifier, a locator is created and added to the [subject identifiers] property of the topic item.

If the topic-identity is specified by a subject locator, a locator is created (the leading = is not part of the locator) and added to the [subject locators] property of the topic item.

If the topic-identity is specified by an item identifier, a locator is created (the leading ^ is not part of the locator) and added to the [item identifiers] property of the topic item.

If the topic-identity is specified by a wildcard, the locator created in accordance with the procedure described in 3.3.8 is added to the [item identifiers] property of the topic item.

If the topic item created through deserialization of a topic-identity is equal to another topic item (c.f. ISO/IEC 13250-2 5.3); the two topic items are merged according to the procedure given in ISO/IEC 13250-2.

Variables shall occur only within templates.

```
# A topic referenced by the subject locator "http://www.isotopicmaps.org/"
= http://www.isotopicmaps.org/ .

# A topic referenced by a subject identifier
http://psi.example.org/John_Lennon .

# A topic with a unique item identifier. Within the CTM
```

```
# document it is not possible to reference this topic by item identifier.
? - "A topic".

# A topic with a unique item identifier which may be
# referenced again within the topic map.
?foo - "A new, unique topic".

# Another reference to the same topic later in the document
?foo isa subject .
```

3.3.7 Topic References

If the topic-ref is specified by a topic-identity, the topic item resolved by 3.3.6 is used.

If the topic-ref is an embedded-topic, a topic item is created. A locator is created according to the procedure described in 3.3.8, where the [is hereby handled like an anonymous-wildcard. The locator is added to the [item identifiers] property of the topic item. The topic item is used as input for the topic-tail (3.9) procedure.

3.3.8 Creating locators from wildcards

If a wildcard is specified by an anonymous-wildcard, the wildcard-counter's value is incremented by 1. A locator is created by concatenating the wildcard-iri, the string #$__ and the value of the wildcard counter.

If a wildcard is specified by a named-wildcard, and that named-wildcard has not been processed before, the wildcard-counter's value is incremented by 1. A locator is created by concatenating the wildcard-iri, the string #$__, the value of the wildcard-counter, the . character and the identifier part of the named-wildcard.

If the named-wildcard has been processed before, the locator produced by its first occurrence is reused.

3.3.9 Scope

The scope construct is used to assign a scope to an information item.

During deserialization, each topic-ref element is processed according to the procedure described in 3.3.7. These topic items are gathered into a set that is assigned as the value of the [scope] property of the statement being processed.

3.3.10 Reifier

The reifier construct is used to refer from the statement on which it appears to the topic reifying that construct. The reference is a topic-ref as described in 3.3.7

During deserialization the topic-ref is resolved into a topic item following the procedure in 3.3.7. The topic item is set as the value of the [reifier] property of the statement being processed.

3.3.11 Type

The type construct is used to assign a type to the information item in which it occurs. The type is always a topic, indicated by topic-ref.

During deserialization the topic-ref produces a topic item following the procedure in 3.3.7, which is set as the value of the [type] property of the information item being processed.

3.4 Literals

3.4.1 General

A literal is a string value with an optional datatype.

For convenience, this part of ISO/IEC 13250 supports integers, decimals, IRIs, dates and dateTime values, which shall be valid according to XSDT, directly.

The following implicit datatypes are associated with the above mentioned literals:

iri
http://www.w3.org/2001/XMLSchema#anyURI

integer
http://www.w3.org/2001/XMLSchema#integer

decimal
http://www.w3.org/2001/XMLSchema#decimal

ctm-integer
http://psi.topicmaps.org/iso13250/ctm-integer

date
http://www.w3.org/2001/XMLSchema#date

date-time
http://www.w3.org/2001/XMLSchema#dateTime

string
http://www.w3.org/2001/XMLSchema#string

Any literal can be expressed by representing the value as a string and appending the datatype qualifier (^^) and a iri-ref which indicates the datatype.

If the literal occurs inside an occurrence (3.10), the occurrence [value] property is set to the value of the literal and the occurrence [datatype] property is set to the datatype of the literal.

If the literal occurs inside a variant (3.12), the variant [value] property is set to the value of the literal and the variant [datatype] property is set to the datatype of the literal.

Variables shall occur only within templates.

```
42  # equivalent to "42"^^xsd:integer

"09-19"^^xsd:gMonthDay
```

3.4.2 String Escape Sequences

String escape sequences can be used to either escape characters which have special meaning in the grammar (i.e. ") or to enter characters which are not available on the keyboard by providing their Unicode codepoint.

The CTM processor shall replace binary sequences which start with the backslash character (\) with the Unicode code point equivalent: \\ becomes U+005C, \" becomes U+0022, \n becomes U+000A, \r becomes U+000D, and \t is replaced with U+0009.

Sequences which are introduced by \u and \U represent a Unicode code point as hexadecimal number, where the hexdigit part refers to the Unicode character encoding.

The hexdigit part shall be decoded into the Unicode character they represent and the \u and \U prefixes shall be removed from the byte stream.

3.5 Topic Map

The topicmap component acts as a container for the topic map but has no further significance. The syntax is as follows:

The optional encoding directive shall occur at the very beginning of the CTM stream. Neither whitespaces nor any other character shall be allowed in front of the encoding directive (see Clause 3 for details).

Directives are used to define the environment for a CTM processor.

3.6 Encoding Directive

The encoding directive specifies the character encoding used by the document.

If the encoding declaration is omitted, UTF-8 encoding shall be assumed unless the byte order mark indicates a different encoding (Clause 3).

The name of the encoding shall be given as a string in the form recommended by IANA-CHARSETS. The " delimiters do not belong to the character name.

If the encoding is provided, it shall occur at the very beginning of the character stream (neither whitespaces nor any other content is allowed in front of this directive)

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

3.7 Version Directive

The version directive states the version number of the CTM syntax, which is currently "1.0". The syntax is as follows:

The version directive 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.

Its usage is recommended for future compatibility.

```
%version 1.0
```

3.8 Topics

The topic construct is used to declare a topic item, assign identifiers to it, and make statements about it, through the assignment of names and occurrences, or the invocation of templates which generate associations (and/or additional names and occurrences). It starts with a topic identity and ends with a period.

During deserialization a topic item is produced and given an identifier by processing the topic-identity in accordance with the procedure described in 3.3.6.

The topic-tail (if any) is processed according to the procedure given in 3.9 with the produced topic item as input.

```
john-lennon.    # a topic with an item identifier and nothing else

lennon. mccartney. harrison. starr. # four topics on one line

# A topic defined by its subject locator
= http://www.isotopicmaps.org/
  - "The ISO Topic Maps Web Site".

# A topic defined by a subject identifier
http://psi.example.org/John_Lennon
```

```
  - "John Lennon".

# A topic with a local identifier and a subject identifier
john http://psi.example.org/John_Lennon;
  - "John Lennon".
```

3.9 Topic Tail

The topic tail is used to assign additional identities to a topic item and to make statements about it. It takes a topic item as input.

For a subject identifier, subject locator, or item identifier an IRI is created according to the procedure described in 3.3.6 and added to the [subject identifiers], or [subject locators], or [item identifiers] property of the topic item.

If either of these procedures makes the topic equal to another topic item (c.f. ISO/IEC 13250-2 5.3); the two topic items are merged according to the procedure given in ISO/IEC 13250-2.

The instance-of production creates a *type-instance* relationship according to ISO/IEC 13250-2 7.2 where the topic on the right hand side plays the role http://psi.topicmaps.org/iso13250/model/type and the input topic item plays the role http://psi.topicmaps.org/iso13250/model/instance.

The kind-of production establishes a *supertype-subtype* relationship according to ISO/IEC 13250-2 7.3 where the input topic item plays the role http://psi.topicmaps.org/iso13250/model/subtype and the topic item on the right hand side plays the role http://psi.topicmaps.org/iso13250/model/supertype.

3.10 Occurrences

The occurrence construct is used to add occurrences to a topic item. The syntax is as follows:

During deserialization the occurrence construct causes an occurrence item to be created, and added to the [occurrences] property of the topic item created by the procedure decribed in 3.8.

```
Paul_McCartney
  birthday: 1942-02-18;
  webpage:  http://en.wikipedia.org/wiki/Paul_McCartney .
```

3.11 Names

The name construct is used to add a topic name to a topic. The syntax is as follows:

During deserialization the name construct causes a topic name item to be created, and added to the [topic names] property of the topic item created by the procedure decribed in 3.8.

If the type is not specified, 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.

```
john-lennon
  - "John Lennon";                # Name with the default name type
  - fullname: "John Winston Lennon".   # Name of type 'fullname'
```

3.12 Variants

The variant construct is used to add a variant name to a topic name item. The syntax is as follows:

During deserialization the variant construct causes a variant item to be created and added to the [variants] property of the topic name item created by the procedure described in 3.11. After the scope has been processed, the topics in

the [scope] property of the topic name item created by the parent name construct are added to the [scope] property of the variant name item.

```
%prefix tm http://psi.topicmaps.org/iso13250/model/
# a topic with a sort name variant
john
  - "John Lennon" ("lennon, john" @ tm:sort).
```

3.13 Associations

The association construct is used to add associations to the topic map. The syntax of associations is as follows:

During deserialization an association item is created for each association and added to the [associations] property of the topic map item.

During deserialization an association role item is created for each role. The role item is added to the [roles] property of the association item.

```
member_of(group: The_Beatles, member: John_Lennon)
```

3.14 Templates

Templates are containers for arbitrary topics and statements and provide a mechanism to reuse statements within different contexts.

The template body consists of ordinary topics and associations and allows topic references (3.3.7) and literals (3.4) to be replaced by variables. The syntax for templates is defined as follows:

In the template-body variables are allowed wherever topic-refs or literals are allowed.

The declaration of a template does not change the topic map until the template has been invoked by a template-invocation (3.15).

It is an error if a template with the same identifier and the same number of arguments is already defined.

```
# Declaration of a template that sets the topic type to 'person', creates an
# occurrence of type 'birthday', and creates an association of type 'born-in'
def born($person, $date, $place)
   $person isa person;
   birthday: $date .
   born-in(person : $person, birthplace : $place).
end

# Invocation of the template inside a topic block
mccartney
  born(1942-06-12, Liverpool).

# Invocation of the template outside topic block
born(mccartney, 1942-06-12, Liverpool)

# Both of the above have the same effect as the following
mccartney
  isa person;
  birthday: 1942-06-12 .
born-in(person: mccartney, birthplace: Liverpool)
```

If a named-wildcard is used inside the template body, it is accessible only within the template. A named-wildcard causes the creation of a topic *each* time the template is invoked. It shall be handled like it has never been processed before.

Template names and topic identifiers do not share the same namespace. If a template with name *A* exists, it is still possible to declare a topic with the identifier *A*.

3.15 Template Invocation

A template invocation requires the following steps:

- If the template invocation occurs within a topic-tail (3.9), insert the topic-identity of the topic which was used as input for the topic-tail as first argument.
- Search for a template definition with the template-name and the number of provided arguments. It is an error if such a template is not defined.
- Bind each variable of the template to the argument according to the argument's position (the first variable is bound to the first argument etc.)
- Add each statement of the template's body to the topic map where any variable is replaced by the bound argument.

NOTE It is possible to refer within templates to other templates which have not been defined yet (forward reference). If the template is invoked, the referenced templates shall have been defined.

EXAMPLE 1

```
# Template invocation within a topic declaration
mccartney
  plays-for(The_Beatles, piano);
  has-shoesize(42).

# Template invocation outside of topic declarations
plays-for(john, The_Beatles, guitar)
has-shoesize(john, 45)
```

EXAMPLE 2 The meaning of iri-refs depend on the context.

```
def wiki-entry($iri, $name)
  $iri              # Used as subject identifier
  - $name;
  description: $iri . # Used as occurrence value
end

wiki-entry(<http://en.wikipedia.org/wiki/John_Lennon>, "John Lennon")

# The following template invocation fails, because a subject locator is
# provided which cannot be used as value of an occurrence.

wiki-entry(=<http://en.wikipedia.org/wiki/Paul_McCartney>, "Sir Paul McCartney")  # -> Error!
```

EXAMPLE 3 Templates can be used to "decorate" other templates.

```
def is-beatle($beatle)
  is-member-of($beatle, The-Beatles)
end

def is-member-of($member, $group)
  member-of(member: $member, group: $group)
end
```

```
is-member-of(sting, The-Police) # Invoking the 'raw' is-member-of template

is-beatle(john)
```

3.16 Directives

3.16.1 Prefix Directive

The prefix directive is used to associate an IRI with an identifier. The syntax is as follows:

During deserialization the prefix component binds the identifier to the IRI.

It is an error if the identifier is already bound unless the identifier is bound to one and the same reference.

It is an error if the concatenation of the iri and the local part of a QName does not produce a valid IRI.

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

wiki:John_Lennon              # QName used as a subject identifier
- "John Lennon" .
```

3.16.2 Include Directive

The include directive is used to include another CTM source into the currently processed CTM instance. The other source is referenced by an IRI.

The IRI of the external information resource indicated by iri-ref is resolved and the resource is parsed with a CTM processor following the rules defined by this part of ISO/IEC 13250 taking the provided iri-ref as document IRI as input. The wildcard-iri of the CTM processor is initialized with the document IRI of this CTM source and the wildcard-counter initialized with the current wildcard-counter's value.

It is an error if the information resource is not a valid CTM resource.

The new data model instance (B) is then merged into the current one (A) by

- Iterating through all topic items in B's [topics] property and identifying those topics which contain locators in their [item identifiers] whose string value start with B's document IRI.

  For each of the these item identifiers a string S is created by subtracting the string value of B's document IRI from the string value of the item identifier. The string S is resolved against the document IRI of A. The resulting string is added to the [item identifiers] property of the topic.
- 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 Adding topics and associations to A may trigger further merges, as described in ISO/IEC 13250-2.

The current wildcard-counter's value is set to wildcard-counter's value of the processor which was responsible to create the data model instance (B).

All templates are imported and can be used by the current CTM character stream as they had been declared at the point of inclusion.

3.16.3 Mergemap Directive

The mergemap directive is used to merge an external topic map into the topic map produced by deserializing the CTM topic map.

The syntax of the mergemap directive is as follows:

This part of ISO/IEC 13250 mandates that each conforming CTM processor shall support the following syntaxes.

http://psi.topicmaps.org/iso13250/ctm
Identifier for the Compact Topic Maps Syntax (CTM) decribed by this part of ISO/IEC 13250

http://psi.topicmaps.org/iso13250/xtm
Identifier for the XML Topic Maps Syntax (XTM) decribed by ISO/IEC 13250-3: 2007

Aside from the normative syntaxes mentioned above, a CTM processor may support more syntaxes. It is an error if the notation is not supported by the CTM processor.

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 Adding topics and associations to A may trigger further merges, as described in ISO/IEC 13250-2.