ISO/IEC JTC1/SC34N0

ISO/IEC JTC1/SC34

Information Technology —

Document Description and Processing Languages

Title:	Topic Maps — XML Syntax
Source:	Lars Marius Garshol, Graham Moore, JTC1 / SC34
Project:	ISO 13250: Topic Maps
Project editor:	Lars Marius Garshol, Graham Moore
Status:	Final Draft International Standard
Action:	For review
Date:	2006-06-19
Summary:	
Distribution:	SC34 and Liaisons
Refer to:	http://www.isotopicmaps.org/sam/sam-xtm/2006-06-19/
Supercedes:	http://www.isotopicmaps.org/sam/sam-xtm/2006-05-02/
Reply to:	Dr. James David Mason (ISO/IEC JTC1/SC34 Chairman) Y-12 National Security Complex Information Technology Services Bldg. 9113 M.S. 8208 Oak Ridge, TN 37831-8208 U.S.A. Telephone: +1 865 574-6973 Facsimile: +1 865 574-1896 E-mail: mailto:mxm@y12.doe.gov http://www.y12.doe.gov/sgml/sc34/sc34oldhome.htm Mr. G. Ken Holman (ISO/IEC JTC 1/SC 34 Secretariat - Standards Council of Canada) Crane Softwrights Ltd. Box 266, Kars, ON K0A-2E0 CANADA Telephone: +1 613 489-0999 Facsimile: +1 613 489-0995 Network: jtc1sc34@scc.ca

Topic Maps — XML Syntax

Contents

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 13250-3 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 *Topic Maps*:

- Part 1: *Overview and Basic Concepts*
- Part 2: *Data Model*
- Part 3: *XML Syntax*
- Part 4: *Canonicalization*
- Part 5: *Reference Model*

Introduction

> **NOTE:**
> The text of this document is identical to that of the official standard as published by ISO. However, the official version is that published by ISO, and not this document.

XTM (XML Topic Maps) 2.0 is a syntax for the interchange of Topic Maps. The syntax is not designed to be extended or modified. Ease of human authoring was not prioritized during the design of XTM, and consequently it is not recommended to edit the syntax directly.

This part of ISO/IEC13250 should be read in conjunction with *[ISO/IEC 13250-2]* since the interpretation of the XTM syntax is defined through a mapping from the syntax to the data model there defined. Informative guidance on how to serialize instances of the data model to the XTM syntax is also provided.

XTM 2.0 is a revision of the XTM 1.0 syntax defined in *[ISO/IEC 13250:2003]* and *[XTM1.0]*. A description of the differences between the two versions can be found in Annex D.

Topic Maps — XML Syntax

1 Scope

This part of ISO/IEC13250 defines an XML-based interchange syntax for Topic Maps,

which can be used to interchange instances of the data model defined in *[ISO/IEC 13250-2]*. It also defines a mapping from the interchange syntax to the data model. The syntax is defined with a RELAX-NG schema, and 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.

> **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.

ISO/IEC 13250-2, *Topic Maps — Data Model*, http://www.isotopicmaps.org/sam/

W3C XML, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, 4 February 2004, http://www.w3.org/TR/2004/REC-xml-20040204

W3C XML-Names, *Namespaces in XML*, W3C Recommendation, 14 January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

W3C XPointer, *XPointer Framework Version 1.0*, W3C Recommendation, 25 March 2003, http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

W3C XML-Infoset, *XML Information Set (Second Edition)*, W3C Recommendation, 4 February 2004, http://www.w3.org/TR/2004/REC-xml-infoset-20040204

W3C Canonical XML, *Canonical XML Version 1.0*, W3C Recommendation, 15 March 2001, http://www.w3.org/TR/2001/REC-xml-c14n-20010315

ISO/IEC 19757-2, *Document Schema Definition Languages (DSDL) — Part 2: Grammar-based validation — RELAX NG*, http://www.relaxng.org

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

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

3 Terms and definitions

For the purposes of this part of ISO/IEC13250, the following terms and definitions apply.

3.1
XTM

the syntax defined in this part of ISO/IEC13250

4 Syntax definition

4.1 About the syntax

The acronym **XTM** is often used to refer to the syntax defined in this part of ISO/IEC13250. Its full name is XML Topic Maps. The namespace for the XTM syntax is `http://www.topicmaps.org/xtm/`.

An XTM document is an XML document that conforms to the XTM syntax. This clause defines the syntax of XTM documents using a RELAX-NG schema in compact syntax *[ISO/IEC 19757-2]*, and their semantics using prose describing the mapping from XTM documents to *[ISO/IEC 13250-2]*. The full schema can be found in Annex A, a DTD in Annex B, and a W3C XML Schema in Annex C.

4.2 Deserialization

The process of exporting Topic Maps 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 XTM syntax are deserialized into instances of the data model defined in *[ISO/IEC 13250-2]*. Serialization is only implicitly defined, but implementations should guarantee that for any data model instance the XTM 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 *[ISO 13250-4]*.

The input to the deserialization process is:

- A document item as defined by *[W3C XML-Infoset]*, representing an XTM document. (Information item properties from *[W3C XML-Infoset]* are referred to using [[property name]], in order to distinguish them from properties from *[ISO/IEC 13250-2]*.)
- An absolute IRI. This is the IRI from which the XTM document 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 XTM document was not read from any particular IRI the application is responsible for providing an IRI considered suitable.

Deserialization is done by processing each element item in the document item in document order. For each element item encountered the operations specified in the clause for that element type are performed. An input element item matches a clause in this document when the [[namespace uri]] property is set to `"http://www.topicmaps.org/xtm/"`, and the [[local name]] matches the element type name given in the title of that clause.

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.

> **NOTE:**
> This part of ISO/IEC13250 requires an instance of the XML Information Set as input

to the deserialization process, but in most cases the actual input will be an XML document. This part of ISO/IEC13250 does not constrain how XML Information Set instances are built from XML documents, but assumes that in most cases this will be done by simply using an XML processor.

XML processors conformant to the XML Recommendation may produce different results given the same XML document, depending on whether they are validating or non-validating, and depending on which optional features they support. Reliance on any particular behaviour in the XML processors used by recipients is strongly discouraged.

4.3 Common syntactical constructs

4.3.1 Common declarations

The following declarations are used throughout the schema for brevity.

```
default namespace = "http://www.topicmaps.org/xtm/"
namespace xtm = "http://www.topicmaps.org/xtm/"

datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = topicMap

reifiable = attribute reifier { xsd:anyURI }?, itemIdentity*
href = attribute href { xsd:anyURI }

any-markup = (text | element * - xtm:* { attribute * { text }*, any-markup* })*
```

4.3.2 The reifier attribute

The `reifier` attribute is used to refer from the topic map construct on which it appears to the topic reifying that construct. The reference is an IRI matching one of the topic's item identifiers.

During deserialization the value in the [[normalized value]] property of the attribute node representing the `reifier` attribute is resolved into an absolute IRI following the procedure in 4.3.4. If there is a topic item with that IRI in its [item identifiers] property, that topic item is set as the value of the [reifier] property of the topic map construct being processed. If no such topic item exists, a new topic item is created, the IRI added to its [item identifiers] property, and that topic item is set as the value of the [reifier] property of the topic map construct being processed.

4.3.3 The href attribute

The `href` attribute always references an information resource using a relative or absolute IRI valid according to *[IETF RFC 3986]* and *[IETF RFC 3987]*, but the meaning of the reference depends on context.

During deserialization the value in the [[normalized value]] property of the attribute node representing the `href` attribute is turned into an IRI following the procedure in 4.3.4.

4.3.4 Creating IRIs from strings

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.

4.4 The topicMap element

The `topicMap` element type is the document element of all XTM documents. It acts as a container for the topic map, and can be used to reify it, but has no further significance. It is declared as follows:

```
topicMap = element topicMap { reifiable, version, mergeMap*,
                   (topic | association)* }

version = attribute version { "2.0" }
```

The `version` attribute is used to specify which version of XTM the document conforms to. For XTM 2.0 documents this shall be "2.0".

During deserialization the `topicMap` element causes a topic map item to be created.

4.5 The topic element

The `topic` element type is used to represent topics, and acts as a container and point of reference for topic information. The child elements of the `topic` element provide identification as well as names and occurrences, while association roles played by the topic are specified outside the `topic` element.

The `topic` element type is declared as follows:

```
topic = element topic { id,
                        (itemIdentity | subjectLocator | subjectIdentifier)*,
                        instanceOf?, (name | occurrence)* }

id = attribute id { xsd:ID }
```

The `id` attribute provides a unique identifier within the document for the topic, which is used to refer to it.

During deserialization the `topic` element causes a topic item to be created and inserted into the [topics] property of the topic map item.

A locator is created by concatenating the document IRI, a `"#"` character, and the value of the [[normalized value]] property of the attribute item in the [[attributes]] property of that element item whose [[local name]] property is `"id"`. This locator is added to the [item identifiers] property of the topic item. If this makes the topic item equal to another topic item the two topic items are merged according to the procedure given in *[ISO/IEC 13250-2]*.

4.6 The itemIdentity element

The `itemIdentity` element is used to assign an item identifier to the topic map construct

represented by its parent element. It is declared as follows:

```
itemIdentity = element itemIdentity { href }
```

During deserialization the `itemIdentity` element causes a locator to be created from its `href` attribute as specified in 4.3.3. This locator is added to the [item identifiers] property of the information item created by the parent element. If the parent element is a `topic` element and this makes the topic item equal to another topic item the two topics are merged according to the procedure given in *[ISO/IEC 13250-2]*.

4.7 The subjectLocator element

The `subjectLocator` element is used to assign a subject locator to the topic that is represented by its parent `topic` element. It is declared as follows:

```
subjectLocator = element subjectLocator { href }
```

During deserialization the `subjectLocator` element causes a locator to be created from its `href` attribute as specified in 4.3.3 and added to the [subject locators] property of the topic item created by the parent `topic` element. If this makes the topic item equal to another topic item the two topic items are merged according to the procedure given in *[ISO/IEC 13250-2]*.

4.8 The subjectIdentifier element

The `subjectIdentifier` element is used to assign a subject identifier to the topic that is represented by its parent `topic` element.

The `subjectIdentifier` element is declared as follows:

```
subjectIdentifier = element subjectIdentifier { href }
```

During deserialization the `subjectIdentifier` element causes a locator to be created from its `href` attribute as specified in 4.3.3. This locator is added to the [subject identifiers] property of the topic item created by the parent `topic` element. If this makes the topic item equal to another topic item the two topic items are merged according to the procedure given in *[ISO/IEC 13250-2]*.

4.9 The instanceOf element

The `instanceOf` element type is used to assign one or more types to the topic represented by its parent element. The types are always topics, indicated by the `instanceOf` element's child elements. The `instanceOf` element type is declared as follows:

```
instanceOf = element instanceOf { topicRef+ }
```

For each child element of the `instanceOf` element a topic item is produced following the procedure in 4.20. For each topic item the following steps are then taken:

- A new association item is created, with two association role items in its [roles] property, and a topic item representing the type-instance association type (described in *[ISO/IEC 13250-2]*, 7.2) in its [type] property. If no such topic item exists already, one is created, and the subject identifier added to its [subject identifiers] property.
- The first association role item has its [type] property set to the topic item representing the `type` role in the same association (see the section referenced above), while the [player] property is set to the topic produced by the child `topicRef` element.
- The second association role item has its [type] property set to the topic item representing the `instance` role in the same association (see the section referenced above), while the [player] property is set to the topic produced by the parent element (that is, the current topic).

4.10 The name element

The `name` element type is used to add topic names to the topic represented by the parent `topic` element. The child elements of the `name` element provide the property values of the topic name item.

The `name` element type is declared as follows:

```
name = element name { reifiable, type?, scope?, value, variant* }
```

During deserialization the `name` element causes a topic name item to be created, and added to the [topic names] property of the topic item created by the parent `topic` element.

If the `name` element has a `type` child element it is processed according to the procedure in 4.14. 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.

4.11 The value element

The `value` element type is used to provide the value of the topic name. It is declared as follows:

```
value = element value { text }
```

During deserialization the information items in the [[children]] property of the `value` element are traversed, and for each character information item the Unicode character specified by the [[character code]] property is appended to the [value] property of the topic name item created by the parent `name` element.

4.12 The variant element

The `variant` element type is used to add a variant name to a topic name. It is declared as follows:

```
variant = element variant { reifiable, scope, (resourceRef | resourceData) }
```

During deserialization the `variant` element causes a variant item to be created and added to the [variants] property of the topic name item created by the `name` parent element. After the `scope` child element has been processed, the topics in the [scope] property of the topic name item created by the `name` parent element are added to the [scope] property of the variant name item.

4.13 The scope element

The `scope` element type is used to assign a scope to the statement represented by the parent element. It is declared as follows:

```
scope = element scope { topicRef+ }
```

During deserialization the `scope` element has no direct effect on the information set being produced, but changes the interpretation of its child elements. Each `topicRef` child element is processed according to the procedure in 4.20 to produce a topic item. These topic items are gathered into a set that is assigned as the value of the [scope] property of the information item produced by the parent element.

4.14 The type element

The `type` element type is used to assign a type to the topic map construct represented by its parent element. The type is always a topic, indicated by the `type` element's child element.

The `type` element type is declared as follows:

```
type = element type { topicRef }
```

During deserialization the child element produces a topic item following the procedure in 4.20, which is set as the value of the [type] property of the information item produced by the parent element.

4.15 The occurrence element

The `occurrence` element type is used to assign an occurrence to the topic defined by the parent element. It is declared as follows:

```
occurrence = element occurrence { reifiable,
   type, scope?, ( resourceRef | resourceData ) }
```

During deserialization the `occurrence` element causes an occurrence item to be created and added to the [occurrences] property of the topic item created by the parent `topic` element.

4.16 The resourceData element

4.16.1 General

The `resourceData` element type represents an information resource in the form of content contained within the XTM document. This information resource may be either a variant name or an occurrence, and it can have a datatype.

The `resourceData` element type is declared as follows:

```
datatype = attribute datatype { xsd:anyURI }

resourceData = element resourceData { datatype?, any-markup }
```

The `datatype` attribute contains an absolute IRI identifying the datatype of the resource that is represented by the `resourceData` element.

4.16.2 Deserialization

The `resourceData` element sets the [value] property of the information item created by the parent element. If the `datatype` attribute is not present the [datatype] property is set to `"http://www.w3.org/2001/XMLSchema#string"`; if the attribute is present the [datatype] property is set to the value of the attribute.

If the [datatype] property is set to `"http://www.w3.org/2001/XMLSchema#anyType"` the procedure in 4.16.3 is followed.

If the [datatype] property is set to `"http://www.w3.org/2001/XMLSchema#anyURI"` the procedure in 4.3.4 is followed to produce the value of the [value] property from the content of the `resourceData` element. In this case it is an error for the `resourceData` element to have child elements.

Otherwise the information items in the [[children]] property of the element item are traversed, and for each character information item the Unicode character specified by the [[character code]] property is added to the [value] property of the information item created by the parent element. In this case it is an error for the `resourceData` element to have child elements.

4.16.3 Canonicalizing embedded XML

XTM documents may contain arbitrary markup inside `resourceData` elements, and this markup is represented in the data model as a string. A string representation is produced from the embedded markup by applying the canonicalization process described in *[W3C Canonical XML]*. The input to the canonicalization process is an XPath node set (as *[W3C Canonical XML]* requires this). The node set is produced as described below:

- Add XPath nodes for all element, attribute, and character information items that are descendants of the `resourceData` element.
- Remove all namespace nodes attached to these element nodes where there is not at least one element or attribute node in the set with this namespace IRI and namespace prefix.

The second parameter to the canonicalization process is false (that is, comments are not included).

NOTE:
The output of the *[W3C Canonical XML]* is defined as a UTF-8-encoded octet sequence, but the output of the process defined above should be the equivalent string.

4.17 The resourceRef element

The `resourceRef` element type refers to an information resource. The information resource can be an occurrence, if the parent element is `occurrence`, or a variant name, if the parent element is `variant`.

The `resourceRef` element type is declared as follows:

```
resourceRef = element resourceRef { href }
```

During deserialization the `resourceRef` element causes a locator to be produced following the procedure in 4.3.3 and inserted into the [value] property of the information item created by the parent element. The [datatype] property of the information item is also set to `"http://www.w3.org/2001/XMLSchema#anyURI"`.

4.18 The association element

The `association` element type represents associations. The `role` child elements provide the association roles of the association. It is declared as follows:

```
association = element association { reifiable, type, scope?, role+ }
```

During deserialization the `association` element causes an association item to be created, and added to the [associations] property of the topic map item.

4.19 The role element

The `role` element type is used to assign an association role to the association created by the `association` parent element. It is declared as follows:

```
role = element role { reifiable, type, topicRef }
```

During deserialization the `role` element causes an association role item to be created, and added to the [roles] property of the association item created by the parent `association` element. The `topicRef` child element is resolved to a topic according to the procedure in 4.20, and that topic is set as the value of the [player] property of the association role item.

4.20 The topicRef element

The `topicRef` element type refers to a topic, either within the same XML document or externally. The signficance of the topic reference depends on the context. The element type is declared as follows:

```
topicRef = element topicRef { href }
```

The `href` attribute contains the absolute or relative IRI reference that is the topic reference. This IRI reference shall have a fragment identifier which shall be what *[W3C XPointer]* calls a shorthand pointer (formerly barename).

During deserialization a locator is produced from the `topicRef` element according to the rules in 4.3.3. If the data model has a topic item whose [subject identifiers] or [item identifiers] properties contain an equal locator that topic item is the one produced by this `topicRef` element. If no such topic item exists, a topic item is created, and the locator added to its [item identifiers] property. That topic item is then the one produced by this `topicRef` element.

4.21 The mergeMap element

The `mergeMap` element type refers to an external XTM document that is to be merged into the topic map that contains the `mergeMap` element. It is declared as follows:

```
mergeMap = element mergeMap { href }
```

The `href` attribute contains an absolute or relative IRI referring to the XTM document that is to be merged in. The IRI shall not have a fragment identifier.

During deserialization an absolute IRI is produced from the `mergeMap` element's `href` attribute, following the procedure in 4.3.3. The IRI of the external information resource is resolved and the resource is parsed with an XML processor according to *[W3C XML]* to produce an XML Information Set according to *[W3C XML-Infoset]*. It is an error if the resource is not a well-formed XML document. The XML Information Set is then deserialized into a data model instance using the procedure in Clause 4 with the document item and the IRI of the information resource as input.

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]*.

5 Conformance

An XTM document conforms to this part of ISO/IEC13250 provided it:

- Is a well-formed XML document *[W3C XML]*.
- Conforms to *[W3C XML-Names]*.
- Conforms to the schema in Annex A.
- Is deserializable according to the procedure defined in Clause 4 without causing any errors or violating any data model constraints.

An XTM processor conforms to this part of ISO/IEC13250 provided it meets all the requirements given below.

- The XTM processor shall reject any input which is not a conforming XTM document.
- The XTM processor shall produce a representation that is isomorphic to the data model instance created by the procedure given in Clause 4 for all XTM documents.

A A RELAX-NG schema for XTM 2.0 (normative)

```
# ========================================================================
#
# XML Topic Maps 2.0
#
# This is the normative RELAX-NG schema for the XTM 2.0 syntax, as
# defined in ISO 13250-3.
#
# See http://www.isotopicmaps.org/sam/sam-xtm/
#
# ========================================================================

# --- Common declarations

default namespace = "http://www.topicmaps.org/xtm/"
namespace xtm = "http://www.topicmaps.org/xtm/"

datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = topicMap

reifiable = attribute reifier { xsd:anyURI }?, itemIdentity*
href = attribute href { xsd:anyURI }

any-markup = (text | element * - xtm:* { attribute * { text }*, any-markup* })*


# --- The schema

topicMap = element topicMap { reifiable, version, mergeMap*,
                              (topic | association)* }

version = attribute version { "2.0" }

topic = element topic { id,
                        (itemIdentity | subjectLocator | subjectIdentifier)*,
                        instanceOf?, (name | occurrence)* }
id = attribute id { xsd:ID }

name = element name { reifiable, type?, scope?, value, variant* }

value = element value { text }

variant = element variant { reifiable, scope, (resourceRef | resourceData) }

scope = element scope { topicRef+ }

instanceOf = element instanceOf { topicRef+ }

type = element type { topicRef }

occurrence = element occurrence { reifiable,
  type, scope?, ( resourceRef | resourceData ) }

datatype = attribute datatype { xsd:anyURI }

resourceData = element resourceData { datatype?, any-markup }

association = element association { reifiable, type, scope?, role+ }

role = element role { reifiable, type, topicRef }
```

```
topicRef = element topicRef { href }
resourceRef = element resourceRef { href }
subjectLocator = element subjectLocator { href }
subjectIdentifier = element subjectIdentifier { href }
itemIdentity = element itemIdentity { href }

mergeMap = element mergeMap { href }

# --- End of schema
```

B The XTM 2.0 DTD (informative)

```
<!-- ..................................................... -->
<!-- XML Topic Map DTD  ................................... -->

<!-- XML Topic Map (XTM) DTD, Version 2.0

     This is XTM 2.0, an XML interchange syntax for ISO 13250 Topic
     Maps, defined by ISO 13250-3.

     Use this URI to identify the XTM namespace:

         "http://www.topicmaps.org/xtm/"

     The formal public identifier for this DTD is:

         "ISO/IEC 13250-3:2005//DTD XML Topic Maps (XTM) 2.0//EN"

     See http://www.isotopicmaps.org/sam/sam-xtm/

-->

<!-- topicMap ............................................. -->

<!ELEMENT topicMap
   ( itemIdentity*, mergeMap*, ( topic | association )* )
>
<!ATTLIST topicMap
   version         CDATA     #FIXED '2.0'
   xmlns           CDATA     #FIXED 'http://www.topicmaps.org/xtm/'
   reifier         CDATA     #IMPLIED
>

<!-- topic ................................................ -->

<!ELEMENT topic
   ( ( itemIdentity | subjectLocator | subjectIdentifier )*,
     instanceOf?, ( name | occurrence )* )
>
<!ATTLIST topic
   id              ID        #REQUIRED
>

<!-- itemIdentity ......................................... -->

<!ELEMENT itemIdentity
   EMPTY
>
<!ATTLIST itemIdentity
   href            CDATA     #REQUIRED
>

<!-- subjectLocator........................................ -->

<!ELEMENT subjectLocator
   EMPTY
>
```

```
<!ATTLIST subjectLocator
   href            CDATA     #REQUIRED
>

<!-- subjectIdentifier..... -->

<!ELEMENT subjectIdentifier
   EMPTY
>
<!ATTLIST subjectIdentifier
   href            CDATA     #REQUIRED
>

<!-- name ..... -->

<!ELEMENT name
   ( itemIdentity*, type?, scope?, value, variant* )
>
<!ATTLIST name
   reifier         CDATA     #IMPLIED
>

<!-- value ..... -->

<!ELEMENT value  ( #PCDATA ) >

<!-- variant ..... -->

<!ELEMENT variant
   ( itemIdentity*, scope, ( resourceRef | resourceData ) )
>
<!ATTLIST variant
   reifier         CDATA     #IMPLIED
>

<!-- scope ..... -->

<!ELEMENT scope
   ( topicRef )+
>

<!-- instanceOf ..... -->

<!ELEMENT instanceOf
   ( topicRef )+
>

<!-- type ..... -->

<!ELEMENT type
   ( topicRef )
>

<!-- occurrence ..... -->

<!ELEMENT occurrence
   ( itemIdentity*, type, scope?, ( resourceRef | resourceData ) )
>
<!ATTLIST occurrence
   reifier         CDATA     #IMPLIED
>

<!-- resourceData ..... -->

<!ELEMENT resourceData
   ANY
>
<!ATTLIST resourceData
   datatype        CDATA     #IMPLIED
>
```

```
<!-- resourceRef ..................................................... -->

<!ELEMENT resourceRef
   EMPTY
>
<!ATTLIST resourceRef
   href            CDATA     #REQUIRED
>

<!-- association ..................................................... -->

<!ELEMENT association
   ( itemIdentity*, type, scope?, role+ )
>
<!ATTLIST association
   reifier         CDATA     #IMPLIED
>

<!-- role ............................................................ -->

<!ELEMENT role
   ( itemIdentity*, type, topicRef )
>
<!ATTLIST role
   reifier         CDATA     #IMPLIED
>

<!-- topicRef ........................................................ -->

<!ELEMENT topicRef
   EMPTY
>
<!ATTLIST topicRef
   href            CDATA     #REQUIRED
>

<!-- mergeMap ........................................................ -->

<!ELEMENT mergeMap
   EMPTY
>
<!ATTLIST mergeMap
   href            CDATA     #REQUIRED
>

<!-- end of XML Topic Map (XTM) 2.0 DTD .......................... -->
```

C An XML Schema schema for XTM 2.0 (informative)

```
<!-- ............................................................... -->
<!-- XML Topic Map Schema  ........................................ -->

<!-- XML Topic Map (XTM) Schema, Version 2.0

     This is XTM 2.0, an XML interchange syntax for ISO 13250 Topic
     Maps, defined by ISO 13250-3.

     See http://www.isotopicmaps.org/sam/sam-xtm/

-->
<xs:schema targetNamespace="http://www.topicmaps.org/xtm/"
           elementFormDefault="qualified"
           xmlns="http://www.topicmaps.org/xtm/"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:dc="http://purl.org/dc/elements/1.1/">
  <xs:annotation>
```

```
  <xs:appinfo>
    <dc:title>W3C XML Schema for XTM 2.0</dc:title>
    <dc:owner>ISO/IEC JTC1 SC34</dc:owner>
    <dc:contributor>Max Voskob</dc:contributor>
    <dc:contributor>Lars Marius Garshol</dc:contributor>
    <dc:contributor>Ann Wrightson</dc:contributor>
  </xs:appinfo>
</xs:annotation>

<!-- any-markup ..................................................... -->
<xs:complexType name="any-markup" mixed="true">
  <xs:complexContent mixed="true">
    <xs:restriction base="xs:anyType">
      <xs:sequence>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="datatype" type="xs:anyURI"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<!-- topicMap ....................................................... -->
<xs:element name="topicMap">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="mergeMap" minOccurs="0" maxOccurs="unbounded"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="topic"/>
        <xs:element ref="association"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="reifier" type="xs:anyURI"/>
    <xs:attribute name="version" fixed="2.0"/>
  </xs:complexType>
</xs:element>

<!-- topic .......................................................... -->
<xs:element name="topic">
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="itemIdentity"/>
        <xs:element ref="subjectLocator"/>
        <xs:element ref="subjectIdentifier"/>
      </xs:choice>
      <xs:element ref="instanceOf" minOccurs="0" maxOccurs="1"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="name"/>
        <xs:element ref="occurrence"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>

<!-- itemIdentity ................................................... -->
<xs:element name="itemIdentity">
  <xs:complexType>
    <xs:attribute name="href" type="xs:anyURI" use="required"/>
  </xs:complexType>
</xs:element>

<!-- subjectLocator ................................................. -->
<xs:element name="subjectLocator">
  <xs:complexType>
    <xs:attribute name="href" type="xs:anyURI" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<!-- subjectIdentifier ..................................... -->
<xs:element name="subjectIdentifier">
  <xs:complexType>
    <xs:attribute name="href" type="xs:anyURI" use="required"/>
  </xs:complexType>
</xs:element>

<!-- name .................................................. -->
<xs:element name="name">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="type" minOccurs="0"/>
      <xs:element ref="scope" minOccurs="0"/>
      <xs:element ref="value"/>
      <xs:element ref="variant" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="reifier" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>

<!-- value ................................................. -->
<xs:element name="value" type="xs:string"/>

<!-- variant ............................................... -->
<xs:element name="variant">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="scope"/>
      <xs:choice>
        <xs:element ref="resourceData"/>
        <xs:element ref="resourceRef"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="reifier" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>

<!-- scope ................................................. -->
<xs:element name="scope">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="topicRef" maxOccurs="unbounded"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<!-- instanceOf ............................................ -->
<xs:element name="instanceOf">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="topicRef" maxOccurs="unbounded"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<!-- type .................................................. -->
<xs:element name="type">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="topicRef"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<!-- occurrence ............................................ -->
<xs:element name="occurrence">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="type"/>
        <xs:element ref="scope" minOccurs="0"/>
        <xs:choice>
          <xs:element ref="resourceRef"/>
          <xs:element ref="resourceData"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="reifier" type="xs:anyURI"/>
    </xs:complexType>
  </xs:element>

  <!-- resourceData ..................................................... -->
  <xs:element name="resourceData" type="any-markup"/>

  <!-- association ...................................................... -->
  <xs:element name="association">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="type"/>
        <xs:element ref="scope" minOccurs="0"/>
        <xs:element ref="role" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="reifier" type="xs:anyURI"/>
    </xs:complexType>
  </xs:element>

  <!-- role ............................................................. -->
  <xs:element name="role">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="itemIdentity" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="type"/>
        <xs:element ref="topicRef"/>
      </xs:sequence>
      <xs:attribute name="reifier" type="xs:anyURI"/>
    </xs:complexType>
  </xs:element>

  <!-- topicRef ......................................................... -->
  <xs:element name="topicRef">
    <xs:complexType>
      <xs:attribute name="href" type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>

  <!-- resourceRef ...................................................... -->
  <xs:element name="resourceRef">
    <xs:complexType>
      <xs:attribute name="href" type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>

  <!-- mergeMap ......................................................... -->
  <xs:element name="mergeMap">
    <xs:complexType>
      <xs:attribute name="href" type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

D Differences with XTM 1.0 (informative)

This annex describes the differences between the syntax defined in this edition of ISO

13250 and that given in *[ISO/IEC 13250:2003]*.

The differences are:

- The namespace URI has changed.
- The `version` attribute has been added to the `topicMap` element.
- The `parameters` element has been replaced by `scope`.
- The `roleSpec` element has been replaced by `type`.
- The `member` element has been replaced by `role`.
- A single topic reference is now required as the child of `role`.
- The `baseName` element has been replaced by `name`.
- The `instanceOf` element has been replaced by `type` everywhere except inside `topic`.
- The `type` element is now allowed inside `name`.
- The `variantName` and `subjectIdentity` elements have been removed.
- The `variant` element can no longer be nested.
- The `type` element is now required inside `occurrence`, `association`, and `role`.
- The `mergeMap` element no longer supports added scope.
- The `id` attribute has been removed from all elements except `topic`, and the `reifies` attribute has been added on some elements.
- The `itemIdentity`, `subjectLocator`, and `subjectIdentifier` elements have been added.
- The `subjectIndicatorRef` and `resourceRef` elements have been removed.
- XTM no longer uses XLink and XML Base.
- The `mergeMap` element must now come before all `topic` and `association` elements.
- The `datatype` attribute has been added to `resourceData`, which also now supports embedded markup.

E Subject identifiers for defined terms (informative)

This annex defines one subject identifier for each formally defined term in Clause 3. The subject identifiers are defined for the sole purpose of enabling unambiguous reference to the subjects they identify, for example in order to enable the collation of information about those subjects. This part of ISO/IEC13250 attaches no processing semantics of any kind to these identifiers, over and above those associated with subject identifiers in general.

XTM
http://psi.topicmaps.org/iso13250/glossary/XTM

Bibliography

ISO/IEC 13250:2003, *Topic Maps*, 2003, http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf

ISO 13250-4, *Topic Maps — Canonicalization*, http://www.isotopicmaps.org/sam/cxtm/

XTM1.0, *XML Topic Maps (XTM) 1.0 Specification*, Steve Pepper, Graham Moore, TopicMaps.Org, 2001, http://www.topicmaps.org/xtm/1.0/