**ISO/IEC JTC 1/SC 34**

Date: 2007-02-22

**ISO/IEC WD N458**

ISO/IEC JTC 1/SC 34/WG 3

Secretariat: SCC

# Topic Maps Constraint Language

Document type: International Standard
Document subtype:
Document stage: (20) Preparatory
Document language: E

## 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.

ISO/IEC N458 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 34, Document Description and Processing Languages.

# Introduction

This International Standard defines a means to express constraints on topic maps conforming to the Topic Map Data Model *[TMDM]*.

# Topic Maps Constraint Language

## 1 Scope

This International Standard defines a data model for representing constraints on instance of the topic map data model and the formal semantics for the interpretation of different constraint types. It also defines a syntactic form that can be used to represent the model structures. This International Standard expresses constraints using topic map constructs and the interpretation of these constraints as TMQL.

## 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.

Unicode, *The Unicode Standard, Version 4.0*, The Unicode Consortium, Boston, MA, USA, Addison-Wesley , 2003, ISBN 0-321-18578-1

TMDM, *ISO 13250-2 Topic Maps — Data Model*, ISO, 2005, available at <http://www.isotopicmaps.org/sam/sam-model/>

TMQL, *ISO Topic Maps Query Language Working Draft*, ISO, 2005, available at <http://www.isotopicmaps.org/tmql/>

CTM, *ISO Compact Topic Map Notation Working Draft*, ISO, 2006, available at <http://www.isotopicmaps.org/ctm/>

XML 1.0, *Extensible Markup Language (XML) 1.0*, W3C, Third Edition, W3C Recommendation, 04 February 2004, available at <http://www.w3.org/TR/REC-xml/>

RFC3986, *RFC 3986 - Uniform Resource Identifiers (URI): Generic Syntax*, The Internet Society, 2005, available at <http://www.ietf.org/rfc/rfc3986.txt>

RFC3987, *RFC 3987 - Internationalized Resource Identifiers (IRIs)*, The Internet Society, 2005, available at <http://www.ietf.org/rfc/rfc3987.txt>

## 3 Notation and Conventions

### 3.1 Notation and Syntax

TMCL constraints are represented as topic map structures this International Standard adopts TMDM as the formalism for representing the TMCL data model.

### 3.2 Formal Semantics

The representation of TMCL is expressed in terms of TMDM.

The formal interpretation of TMCL is defined in terms of TMQL. TMQL is in turn grounded on both the TMRM and the related path model.

# 4 TMCL

TMCL defines constraint types and an interpretation of instances of those types. The interpretation indicates in an unambiguous fashion what it means for an instance of a given constraint type to be evaluated in the context of a TMDM instance. The result of the evaluation of any given constraint is either true or false.

The TMCL constraint types are defined in terms of the topic map data model. The formal interpretation of each constraint type is defined in terms of TMQL.

All constraint types defined follow a common pattern. They are all defined as subtypes of the topic type 'Constraint' and they all have an occurrence of type 'Constraint Evaluation Function'. Thus, it is possible to define new constraint types that address specific domain requirements while still fitting into the overall TMCL validation framework. The constraint types defined in TMCL are intended for use in a 'traditional' entity constraint language fashion, such as ERM, UML etc. They are intended to be used to define the set of identities, occurrences, names and played association roles that a topic of a given type must have in order to been deemed valid e.g. topics of type person must have one unscoped name and be related to one other topic of type person who is their mother. The generalised constraint model, that of 'Constraint' and 'Constraint Evaluation Function', is intended to facilitate map wide constraints that are not centered on one particular type. e.g. If a person works in department X then they must also be the author of at least 5 research papers.

## 4.1 Validation Semantics

A TMCL constraint topic MUST have a 'Constraint Evaluation Function' occurrence. This occurrence contains the TMQL expression that is evaluated in the context of a given topic map. The evaluation of this expression MUST return true or false. If the expression returns true, validation is said to have succeeded.

TMCL allows constraints to be grouped into what is called a schema. A schema is defined as a topic of type 'Schema' and connected to constraint topics via associations (see later for details).

If a given topic map is valid in respect to a constraint, then validation is said to have succeeded. More formally it can be said that :

```
Given:
  TopicMap: t
  Constraint  : c
Then:
  Validate(t, c) => true  | false
```

Furthermore, a schema is defined as a collection of constraints and if all constraints in that schema are valid then the schema can be said to be valid for a given topic map. More formally:

```
Given:
  TopicMap : t
  Schema : s

Then:
  Validate(t, s) =>
  ForAll Constraint c In s
    Validate(t, c)
  End
=> true | false
```

Constraints are completely independent from each other and can be evaluated in any order.

## 4.2 TMCL Syntax

TMCL does not define its own syntax. Each constraint type has a corresponding TMDM representation and while these representations could be authored as XTM or CTM directly this would be a painful and error prone activity.

TMCL has defined a set of CTM templates that construct instances of the constraint types. These templates can be used in any CTM document to express a schema.

### 4.2.1 TMCL Include Directive

To include the TMCL templates the following directive must be used at the begining of the CTM document

```
#include tmcl as http://psi.isotopicmaps.org/tmcl/templates
```

### 4.2.2 TMCL Template Usage Bootstrapping

The TMCL templates construct an instance of a given constraint type and by default will connect it to a schema topic. The schema topic that the constraint template connects the constraint topic to is *schema.

The CTM document that is using the constraint templates MUST create the *schema topic before invoking any template. An example of the usage follows:

```
#include tmcl as http://psi.isotopicmaps.org/tmcl/templates

*schema http://psi.example.com/myschema

// template invocations follow
```

### 4.2.3 TMCL Add Constraint to Schema Template

There is one template used by all the CTM templates. This template connects the current schema, as defined by *schema, to a constraint. It is defined as follows and MUST be included in any CTM based authoring.

```
soc http://psi.isotopicmaps.org/tmcl/associationtypes/schema_owns_constraint
soc_schema http://psi.isotopicmaps.org/tmcl/associationtypes/schema_owns_constraint/schema_role
soc_constraint http://psi.isotopicmaps.org/tmcl/associationtypes/schema_owns_constraint/constraint_role

def AddConstraintToTemplate($schema, $constraint)
  soc($schema : soc_schema, $constraint : soc_constrainnt)
end
```

## 4.3 TMCL MetaModel

TMCL requires a metamodel in order to define constraints types using TMDM. This section lists these topics.

### 4.3.1 Schema

Instances of this topic are used to group a set of constraints that comprise a schema.

The topic 'Schema' has the following subject identifier *http://psi.isotopicmaps.org/tmcl/schema*. It has the following representation using the CTM notation:

```
schema
  http://psi.isotopicmaps.org/tmcl/schema
  isa http://psi.isotopicmaps.org/tmcl/topictype
```

### 4.3.2 Constraint

This topic is used as the base type for all topics that are to be considered as constraint types.

The topic 'Constraint' has the following subject identifier *http://psi.isotopicmaps.org/tmcl/constraint*. It has the following representation using the CTM notation:

```
constraint
  http://psi.isotopicmaps.org/tmcl/constraint
  isa http://psi.isotopicmaps.org/tmcl/topictype
```

### 4.3.3 Constraint Evaluation Function

This topic is an occurrence type that contains the TMQL expression used for the evaluation of instances of the given constraint type.

The topic 'Constraint Evaluation Function' has the following subject identifier *http://psi.isotopicmaps.org/tmcl/constraint_eval_function*. It has the following representation using the CTM notation:

```
constraint_eval_function
  http://psi.isotopicmaps.org/tmcl/constraint_eval_function
  isa http://psi.isotopicmaps.org/tmcl/occurrencetype
```

## 4.4 TMCL Constraint Types

The following sections contain the constraint types defined by TMCL. Each constraint type is defined in terms of a CTM template. This template acts as the syntax for authoring constraints of that type and also for defining the TMDM instance that represents a constraint. The formal interpretation of the constraint is defined as an occurrence in the form of TMQL.

Each constraint type may also define some support topics. These topics are defined in terms of their subject identifiers, the 'meaning' of these topics is confered in the TMQL validation expression.

### 4.4.1 Topic Type Constraint Type

The *topictype constraint type* provides a way to constrain that only topics explicitly defined as topic types can have instances. To indicate a topic as a topic type TMCL defines the topic for topictype.

Supporting topics for this constraint type are:

```
topictype http://psi.topicmaps.org/tmcl/topictype

topicTypeConstraint http://psi.topicmaps.org/tmcl/topictypeconstraint,
                   iko constraint

constraint_eval_function http://psi.isotopicmaps.org/tmcl/constraint_eval_function
                        isa http://psi.isotopicmaps.org/tmcl/occurrencetype
```

The CTM template for constructing instances of this constraint type is defined as:

```
def TopicTypeConstraint()
  *ttc1 isa topicTypeConstraint
  *ttc1 constraint_eval_function """
                              TMQL for if any topics used as a topic type and not instance of topictype then fail.
                             """
  AddConstraintToSchema(*schema, *ttc1)
end
```

Usage Example

This constraint type provides a way to ensure that all topics that are used as topic types are expliticly defined as topic types. This is acheived by making topics that are considered topic types instances of the well known topic, topictype.

```
// define meta model topics
topictype http://psi.topicmaps.org/tmcl/topictype

// define type topic
person isa topictype

// create topic type constraint topic
tmcl:topictypeconstraint()
```

### 4.4.2 Association Type Constraint Type

The *associationtype constraint type* provides a way to constrain that only topics explicitly defined as association types are used to type associations. To indicate a topic as an association type TMCL defines the topic for associationtype. This topic should be used to unambiguously indicate that a topic is an association type.

The topic for association type is defined as follows:

```
associationtype http://psi.topicmaps.org/tmcl/associationtype
```

The CTM template for constructing instances of this constraint type is defined as follows:

```
associationtype http://psi.topicmaps.org/tmcl/associationtype

associationTypeConstraint http://psi.topicmaps.org/tmcl/associationtypeconstraint
                         iko constraint

constraint_eval_function http://psi.isotopicmaps.org/tmcl/constraint_eval_function
                        isa http://psi.isotopicmaps.org/tmcl/occurrencetype

// template for constructing instances of AssociationTypeConstraint
def tmcl:associationtypeconstraint()
  *atc1 isa associationTypeConstraint
  *atc1 constraint_eval_function """
                              Any topics used as an association type and not instance of associationtype then fail.
                             """
  // connect to schema
  AddConstraintToSchema(*schema, *atc1)
end
```

Usage Example

This constraint is a way to ensure that all topics that are used as association types are expliticly defined as association types. This is acheived by making topics that are considered association types instances of the well known topic, associationtype.

```
// define meta model topics
associationtype http://psi.topicmaps.org/tmcl/associationtype

// define association type topic
person_works_for_company isa associationtype

// create association type constraint topic
tmcl:associationtypeconstraint()
```

### 4.4.3 AssociationRoleType Constraint Type

The *associationroletype constraint type* provides a way to constrain that only topics explicitly defined as association role types are used in such a fashion. To indicate a topic as an association role type TMCL defines the topic for associationroletype. This topic should be used to unambiguously indicate that a topic is an association role type.

The topic for associationroletype is defined as follows:

```
associationroletype http://psi.topicmaps.org/tmcl/associationroletype
```

The CTM template for constructing instances of this constraint type is defined as follows:

```
associationroletype http://psi.topicmaps.org/tmcl/associationroletype
associationRoleTypeConstraint http://psi.topicmaps.org/tmcl/associationroletypeconstraint,
                              iko constraint

constraint_eval_function http://psi.isotopicmaps.org/tmcl/constraint_eval_function
                         isa http://psi.isotopicmaps.org/tmcl/occurrencetype

// template for constructing instances of AssociationRoleTypeConstraint
def tmcl:associationroletypeconstraint()
  *artc1 isa associationRoleTypeConstraint
  *artc1 constraint_eval_function """
                                  Any topics used as an association role type and not instance of associationroletype then fail.
                                 """
  // connect to schema
  AddConstraintToSchema(*schema, *artc1)
end
```

Usage Example
This constraint is a way to ensure that all topics that are used as association role types are expliticly defined as association role types. This is acheived by making topics that are considered association role types instances of the well known topic, associationroletype.

```
// define meta model topics
associationtype http://psi.topicmaps.org/tmcl/associationtype
associationroletype http://psi.topicmaps.org/tmcl/associationroletype

// define association type
person_works_for_company isa associationtype

// define association role types
employer isa associationroletype
```

```
employee isa associationroletype

// create association type constraint topic
tmcl:associationroletypeconstraint()
```

### 4.4.4 OccurrenceType Constraint Type

The *occurrencetype constraint type* provides a way to constrain that only topics explicitly defined as occurrence types are used in such a fashion. To indicate a topic as an occurrence type TMCL defines the topic for occurrencetype. This topic should be used to unambiguously indicate that a topic is an occurrence type.

The topic for occurrencetype is defined as follows:

```
occurrencetype http://psi.topicmaps.org/tmcl/occurrencetype
```

The CTM template for constructing instances of this constraint type is defined as follows:

```
occurrencetype http://psi.topicmaps.org/tmcl/occurrencetype
occurrenceTypeConstraint http://psi.topicmaps.org/tmcl/occurrencetypeconstraint,
                iko constraint

constraint_eval_function http://psi.isotopicmaps.org/tmcl/constraint_eval_function
                isa http://psi.isotopicmaps.org/tmcl/occurrencetype

// template for constructing instances of OccurrenceTypeConstraint
def tmcl:occurrencetypeconstraint()
  *otc1 isa occurrenceTypeConstraint
  *otc1 constraint_eval_function """
                    Any topics used as an occurrence type and not an instance of topictype then fail.
                    """
  // connect to schema
  AddConstraintToSchema(*schema, *otc1)
end
```

Usage Example
This constraint is a way to ensure that all topics that are used as occurrence types are expliticly defined as occurrence types. This is acheived by making topics that are considered occurrence types instances of the well known topic, occurrencetype.

```
// define meta model topics
occurrencetype http://psi.topicmaps.org/tmcl/occurrencetype

// define occurrence type topic
age isa occurrencetype

// create occurrence type constraint topic
occurrencetypeconstraint()
```

### 4.4.5 Abstract Topic Type Constraint Type

The *abstract topic type constraint type* provides a way to express that a given topic type is abstract and must therefor not have any instances.

1 **[topictype]** : Topic. The topic type that is being defined as abstract.

The CTM template for constructing instances of this constraint type is defined as follows:

```
AbstractTopicTypeConstraint http://psi.topicmaps.org/tmcl/isabstractconstraint
                        iko constraint

constraint_eval_function http://psi.isotopicmaps.org/tmcl/constraint_eval_function
                      isa http://psi.isotopicmaps.org/tmcl/occurrencetype

catt http://psi.isotopicmaps.org/tmcl/constraintappliestotopictype
     isa associationtype

catt_constraint_role http://psi.isotopicmaps.org/tmcl/constraintappliestotopictype_constraint_role
                   isa associationroletype

catt_topictype_role http://psi.isotopicmaps.org/tmcl/constraintappliestotopictype_topictype_role
                  isa associationroletype

// template for constructing instances of AbstractTopicTypeConstraint
def isAbstract($topictype)
  *attc1 isa AbstractTopicTypeConstraint
  constraint_eval_function """
                        if $this->catt->$topictype has any instances then fail.
                        """

  catt(*attc1 : catt_constraint_role, $topictype : catt_topictype_role)
      AddConstraintToSchema(*schema, *attc1)
end
```

Usage Example
This constraint provides a way to express that the specified topic type must not have any instances. It is most useful when defining a type hierarchy to ensure that only the most specialised types have instances.

```
// define topic type
vehicle isa topictype
car iko vehicle

isAbstract(vehicle)
```

### 4.4.6 Disjoint Constraint

A *disjoint constraint* provides a way to express that a topic can only be an instance of one of the referenced types..

1 **[atype]** : Topic. One of the types of which instances must be disjoint.

2 **[btype]** : Topic. One of the types of which instances must be disjoint.

The Topic Map representation of instances of this constraint type is as follows:

```
djc isa disjointconstraint

djc_includes_type(djc   : djc_includes_type_constraintrole,
ATYPE : djc_includes_type_typerole)

djc_includes_type(djc   : djc_includes_type_constraintrole,
BTYPE : djc_includes_type_typerole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(fn:count(select $x where $x isa ATYPE & $y isa BTYPE & $x=$y) == 0) => true
```

The CTM template for this constraint is defined as follows:

```
ctm:template disjointCostraint(atype, btype) =>
djc isa disjointconstraint

djc_includes_type(djc   : djc_includes_type_constraintrole,
atype : jc_includes_type_typerole)

djc_includes_type(djc   : djc_includes_type_constraintrole,
btype : jc_includes_type_typerole)
end
```

The example usage is as follows:

```
disjointConstraint(atype, btype)
```

### 4.4.7 Subject Identifier Constraint

A *subject identifier constraint* provides a way to constrain the shape and number of topic subject identifiers.

1 **[topictype]** : Topic. The topic type to which this constraint applies.

2 **[cardmin]** : Integer. The mininum allowed number of identifiers

3 **[cardmax]** : Integer. The maximum allowed number of identifiers

4 **[identifiervaluepattern]** : RegularExpression. A regular expression that must match the resource value of the subject identifier.

The subject identifier constraint type is defined as follows:

```
sic isa subjectidentifierconstraint
mincardinalityfacet: CMIN
maxcardinalityfacet: CMAX
identifiervaluepattern: REGEXP

sic_constrains_type(SIC   : sic_constrains_type_constraintrole,
ATYPE : sic_constraint_type_typerole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

The example usage is as follows:

```
subjectIdentifierConstraint(atype, 1, 1, '*')
```

### 4.4.8 Subject Locator Constraint

A *subject locator constraint* provides a way to constrain the shape and number of topic subject locators.

1 **[type]** : Topic. The topic type to which this constraint applies.

2 **[cardmin]** : Integrer. The mininum allowed number of identifiers

3 **[cardmax]** : Integrer. The maximum allowed number of identifiers

4 **[identifiervaluepattern]** : RegularExpression. A regular expression that must match the resource value of the subject identifier.

The Topic Map representation of instances of this constraint type is as follows:

```
sic isa subjectlocatorconstraint
mincardinalityfacet: CMIN
maxcardinalityfacet: CMAX
identifiervaluepattern: REGEXP

slc_constrains_type(SIC   : slc_constrains_type_constraintrole,
ATYPE : slc_constraint_type_typerole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(
(fn:count(
select $t where $t isa ATYPE & (fn:regexp(identifiervaluepattern, $t / locators) = 1)
>= CMIN)
&
(fn:count(
select $t where $t isa ATYPE & (fn:regexp(identifiervaluepattern, $t / locators) = 1)
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template subjectLocatorConstraint(atype, cmin, cmax, regexp) =>
sic isa subjectlocatorconstraint

sic
mincardinalityfacet: cmin
maxcardinalityfacet: cmax
valueexp: regexp

sic_constrains_type(SIC   : slc_constrains_type_constraintrole,
ATYPE : slc_constraint_type_typerole)
```

The example usage is as follows:

```
subjectLocatorConstraint(atype, 1, 1, '*')
```

### 4.4.9 TopicName Constraint

A *topicname constraint* provides a way to constrain the type and cardinality of topic names for instances of a given topic type.

1 **[type]** : Topic. The topic type to which this constraint applies.

2 **[nametype]** : Topic. The expected nametype of valid names.

3 **[cardmin]** : An integer. The expected minimum number of names on a topic that is allowed to match this pattern.

4 **[cardmax]** : An integer. The expected maximum number of names on a topic that is allowed to match this pattern.

The TMDM representation of instances of this constraint type is as follows:

```
tnc isa topicnameconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

tncappliestonametype(tnc : tncappliestonametype_constraintrole,
NAMETYPE : tncappliestonametype_nametype)

constraintappliestotopictype(tnc       : constraintappliestotopictype_constraintrole,
ATYPE      : constraintappliestotopictype_topictyperole)
```

Using the topic map representations above, the interpretations of this constraint is as follows:

```
(
(fn:count(
select $t where $t isa ATYPE & (fn:regexp(identifiervaluepattern, $t / bn / NAMETYPE) = 1)
>= CMIN)
&
(fn:count(
select $t where $t isa ATYPE & (fn:regexp(identifiervaluepattern, $t / bn / NAMETYPE) = 1)
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template topicNameConstraint(topictype, nametype, cmin, cmax) =>
tnc isa topicnameconstraint
mincardinalityfacet: cmin
maxcardinalityfacet: cmax

tncappliestonametype(tnc : tncappliestonametype_constraintrole,
nametype : tncappliestonametype_nametype)

constraintappliestotopictype(tnc       : constraintappliestotopictype_constraintrole,
topictype  : constraintappliestotopictype_topictyperole)
```

The example usage is as follows:

```
topicNameConstraint(person, surname, 1, 1)
```

### 4.4.10 NameTypeScope Constraint

A *nametypescope constraint* provides a way to constrain the allowed scoping topics for a name of a given type.

1 **[nametype]** : Topic. The name type to which this constraint applies.

2 **[scopetopictype]** : Topic. A topictype, instances of which can be in the scope of topic names of the given name type.

3 **[cardmin]** : An integer. The expected minimum number of topics of the specified type expected to be found on

topic names of the given name type.

4 **[cardmax]** : An integer. The expected maximum number of topics of the specified type expected to be found on topic names of the given name type.

The TMDM representation of instances of this constraint type is as follows:

```
tnsc isa topicnamescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

tnscappliestonametype(tnc : tnscappliestonametype_constraintrole,
NAMETYPE : tnscappliestonametype_nametype)

constraintappliestotopictype(tnc       : constraintappliestotopictype_constraintrole,
ATYPE      : constraintappliestotopictype_topictyperole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(
(fn:count(
select $t where $t isa ATYPE & $p / bn / NAMETYPE @ $t)
) >= CMIN)
&
(fn:count(
select $t where $t isa ATYPE & $p / bn / NAMETYPE @ $t)
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template topicNameScopeConstraint(nametype, scopetype, cmin, cmax) =>
tnsc isa topicnamescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

tnscappliestonametype(tnc : tnscappliestonametype_constraintrole,
nametype : tnscappliestonametype_nametype)

constraintappliestotopictype(tnc       : constraintappliestotopictype_constraintrole,
scopetype      : constraintappliestotopictype_topictyperole)
```

The example usage is as follows:

```
nameTypeScopeConstraint(displayName, language, 0, 1)
```

### 4.4.11 OccurrenceTypeScope Constraint

A *occurrencetypescope constraint* provides a way to constrain the allowed scoping topics for an occurrence of a given type.

1 **[occurrencetype]** : Topic. The occurrence type to which this constraint applies.

2 **[scopetopictype]** : Topic. A topictype, instances of which can be in the scope of topic occurrences of the

given occurrence type.

3 **[cardmin]** : An integer. The expected minimum number of topics of the specified type expected to be found on topic occurrences of the given occurrence type.

4 **[cardmax]** : An integer. The expected maximum number of topics of the specified type expected to be found on topic occurrences of the given occurrence type.

The TMDM representation of instances of this constraint type is as follows:

```
tnsc isa topicoccurrencescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

tnoscappliestooccurrencetype(tnc : tnoscappliestooccurrencetype_constraintrole,
OCTYPE : tnoscappliestooccurrencetype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
ATYPE       : constraintappliestotopictype_topictyperole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(
(fn:count(
select $s in // OCTYPE [ ^ tm:occurrence ] @ where $s isa ATYPE
) >= CMIN)
&
(fn:count(
select $s in // OCTYPE [ ^ tm:occurrence ] @ where $s isa ATYPE
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template topicOccurrenceScopeConstraint(occurrencetype, scopetype, cmin, cmax) =>
tnsc isa topicoccurrencescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

tnoscappliestooccurrencetype(tnc : tnoscappliestooccurrencetype_constraintrole,
occurrencetype : tnoscappliestooccurrencetype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
scopetype       : constraintappliestotopictype_topictyperole)
```

The example usage is as follows:

```
occurrenceTypeScopeConstraint(content_reference, security_level, 1, 1)
```

### 4.4.12 OccurrenceTypeScope Constraint

A *associationtypescope constraint* provides a way to constrain the allowed scoping topics for an association of a given type.

1 **[associationtype]** : Topic. The association type to which this constraint applies.

2 **[scopetopictype]** : Topic. A topictype, instances of which can be in the scope of associations of the given association type.

3 **[cardmin]** : An integer. The expected minimum number of topics of the specified type expected to be found on associations of the given association type.

4 **[cardmax]** : An integer. The expected maximum number of topics of the specified type expected to be found on associations of the given association type.

The TMDM representation of instances of this constraint type is as follows:

```
atsc isa associationtypescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

atscappliestoassociationtype(tnc : atscappliestoassociationtype_constraintrole,
ASSOCTYPE : atscappliestoassociationtype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
ATYPE      : constraintappliestotopictype_topictyperole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(
(fn:count(
select $s in // ASSOCTYPE [ ^ tm:association ] @ where $s isa ATYPE
) >= CMIN)
&
(fn:count(
select $s in // ASSOCTYPE [ ^ tm:association ] @ where $s isa ATYPE
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template associationTypeScopeConstraint(associationtype, scopetype, cmin, cmax) =>

atsc isa associationtypescopeconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX

atscappliestoassociationtype(tnc : atscappliestoassociationtype_constraintrole,
associationtype : atscappliestoassociationtype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
scopetype      : constraintappliestotopictype_topictyperole)
```

The example usage is as follows:

```
associationTypeScopeConstraint(works_for, timeperiod, 1, 1)
```

### 4.4.13 VariantName Constraint

A *variantname constraint* provides a way to constrain the type, cardinality and value of variant names for instances of a name type..

1 **[nametype]** : Topic. The name type to which this constraint applies.

2 **[cardmin]** : An integer. The minimum number of variant names on a topic name that is allowed to match this pattern.

3 **[cardmax]** : An integer. The maximum number of variant names on a topic name that is allowed to match this pattern.

4 **[datatype]** : URI. An identifier for the datatype of the variant name value.

5 **[resourcevaluepattern]** : RegularExpression. A repexp that must match the resource value of the variant name value.

The TMDM representation of instances of this constraint type is as follows:

```
vnc isa variantnameconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX
datatypevalue: DATATYPE_VALUE
resourcevaluepattern: REGEXP

vncappliestonametype(vnc : vncappliestonametype_constraintrole,
NAMETYPE : vncappliestonametype_nametyperole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(
(fn:count(
select $vname in // topics / name [ NAMETYPE ] / variant
where ((fn:regexp ($vname / value, resourcevaluepattern) = 1) &
$vname / datatype = datatypevalue)
) >= CMIN)
&
(fn:count(
select $vname in // topics / name [ NAMETYPE ] / variant
where ((fn:regexp ($vname / value, resourcevaluepattern) = 1) &
$vname / datatype = datatypevalue)          ) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template variantNameConstraint(nametype, datatype, resourceexp, cmin, cmax) =>

vnc isa variantnameconstraint
mincardinalityfacet: cmin
maxcardinalityfacet: cmax
datatypevalue: datatype
resourcevaluepattern: resourceexp

vncappliestonametype(vnc : vncappliestonametype_constraintrole,
nametype : vncappliestonametype_nametyperole)
```

The example usage is as follows:

```
variantNameConstraint(displyName, "xsd:uri", "*", 1, 1)
```

### 4.4.14 Occurrence Type Constraint

An *occurrence type constraint* provides a way to constrain the allowed datatype of an occurrence type..

1 **[occurrencetype]** : Topic. The topic which is an occurrence type.

2 **[datatype]** : IRI. An IRI that identifies the allowed datatypes for this occurrence type.

The TMDM representation of instances of this constraint type is as follows:

```
otc isa occurrencetypeconstraint
datatypevalue: DATATYPE_VALUE

otcappliestooccurrencetype(vnc : otcappliestooccurrencetype_constraintrole,
OCTYPE : otcappliestooccurrencetype_nametyperole)
```

Using the topic map representation above, the interpretation of this constraint is as follows:

```
(fn:count(select $o in // OCTYPE [ ^ tm:occurrence ] / datatype != DATATYPE_VALUE) = 0) =>
```

### 4.4.15 Topic Occurrence Constraint

A *topic occurrence constraint* defines a way to constrain the type, cardinality and value of occurrences connected to a topic of a given type.

1 **[type]** : Topic. The topic type to which this constraint applies.

2 **[occurrencetype]** : Topic. The occurrence type of valid occurrences.

3 **[resourcevaluepattern]** : RegularExpression. A regular expression that must match the value of the occurrence.

4 **[cardmin]** : An integer. The expected minimum number of names on a topic that is allowed to match this pattern.

5 **[cardmax]** : An integer. The expected maximum number of names on a topic that is allowed to match this pattern.

The TMDM representation of instances of this constraint type is as follows:

```
toc isa topicoccurrenceconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX
occurrencevaluepattern: valuepattern

tocappliestooccurrencetype(tnc : tncappliestonametype_constraintrole,
OCTYPE : tncappliestonametype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
ATYPE      : constraintappliestotopictype_topictyperole)
```

Using the topic map representations above, the interpretations of this constraint is as follows:

```
(
(fn:count(
select $t in // topics where (fn:regexp ($t / occurrence [ OCTYPE ] / value, valuepattern) = 1) &
)
>= CMIN)
&
(fn:count(
select $t in // topics where (fn:regexp ($t / occurrence [ OCTYPE ] / value, valuepattern) = 1) &
) <= CMAX)
)
=> true
```

The syntax for this constraint is defined as a CTM template:

```
ctm:template topicOccurrenceConstraint(topictype, occurrencetype, regexp, cmin, cmax) =>
toc isa topicoccurrenceconstraint
mincardinalityfacet: CARDMIN
maxcardinalityfacet: CARDMAX
occurrencevaluepattern: regexp

tocappliestooccurrencetype(tnc : tncappliestonametype_constraintrole,
occurrencetype : tncappliestonametype_nametype)

constraintappliestotopictype(tnc        : constraintappliestotopictype_constraintrole,
topictype      : constraintappliestotopictype_topictyperole)
```

The example usage is as follows:

```
topicOccurrenceConstraint(person, email, "*@*", 0, *)
```

### 4.4.16 AssociationRole Constraint

A *associationrole constraint* defines the nature of allowed roles in an association of given a type..

1 **[AssociationType]** : Topic. The selector that defines which types of associations are governed by this constraint.

2 **[RoleType]** : Topic. The selector that defines which types of association roles are governed by this constraint.

3 **[cardMin]** : An integer value.. The minimum allowed number of roles within a given matching this pattern.

4 **[cardMax]** : An integer value.. The maximum allowed number of roles matching this pattern.

### 4.4.17 RolePlayer Constraint

A *roleplayer constraint* defines a way to constraint the allowed role players of a given role..

1 **[AssociationType]** : Topic. The selector that defines which types of associations are governed by this constraint.

2 **[RoleType]** : Topic. The selector that defines which types of association roles are governed by this constraint.

3 **[RoleType]** : Topic. The selector that defines which types of association roles are governed by this constraint.

4 **[cardMin]** : An integer value.. The minimum allowed number of roles within a given matching this pattern.

5 **[cardMax]** : An integer value.. The maximum allowed number of roles matching this pattern.

### 4.4.18 OtherRole Constraint

A *otherrole constraint* defines a way to constraint the allowed role given the existence of a certain role in an association..

1 **[AssociationType]** : Topic.

2 **[RoleType]** : Topic.

3 **[OtherRoleType]** : Topic.

4 **[cardMin]** : An integer value.. The minimum allowed number of other allowed roles within the association.

5 **[cardMax]** : An integer value.. The maximum allowed number of other allowed roles within the association.

#### 4.4.19 UniqueRole Constraint

A *uniquerole constraint* defines a way to constraint that a given topic can only play a certain role once in any given map..

1 **[AssociationType]** : Topic.

2 **[RoleType]** : Topic.

#### 4.4.20 UniqueOccurrence Constraint

A *uniqueoccurrence constraint* defines a way to constraint that occurrences of a given type with the same value can only be present on one topic in a given map..

1 **[TopicType]** : Topic.

2 **[OccurrenceType]** : Topic.

# 5 TMCL Extensions

The TMCL data model in its most basic form supports any TMQL expression that returns true or false to be used as a constraint predicate.

# 6 TMCL Conformance

There are two levels of conformance defined by this International Standard.

## 6.1 TMCL Level One Conformance

Level one confir

## 6.2 TMCL Level Two Conformance

# 7 TMCL as TMCL

To illustrate the use of TMCL and to prove its self referential capability this annex defines a TMCL schema for TMCL constructs.

to do

# Bibliography

[1] *TMCL Requirements*, ISO, 2004

[2] *TMCL Use Cases*, ISO, 2004

[3] *TMQL Requirements*, ISO, 2003