INTERNATIONAL STANDARD

**ISO/IEC 15938-12**

First edition
2008-12-15

**AMENDMENT 2**
2011-08-01

# Information technology — Multimedia content description interface —

## Part 12: **Query format**

## AMENDMENT 2: Semantic enhancement

*Technologies de l'information — Interface de description du contenu multimédia —*

*Partie 12: Format de requête*

*AMENDEMENT 2: Amélioration sémantique*



Reference number
ISO/IEC 15938-12:2008/Amd.2:2011(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

 

# 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.

Amendment 2 to ISO/IEC 15938-12:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

# Information technology — Multimedia content description interface —

## Part 12: Query format

## AMENDMENT 2: Semantic enhancement

*In 3.2, add the following:*

RDF: Resource Description Framework (W3C, http://www.w3.org/RDF/)

SPARQL: SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/

*After 6.7, add the following subclause:*

### 6.8 SemanticFieldType

The SemanticFieldType supports the addressing of individual Subject/Object parts of an RDF triple which are involved in a comparison expression or the sorting operation.

#### 6.8.1 Syntax

```
<complexType name="SemanticFieldType">
  <sequence>
    <element name="Var" type="string"/>
  </sequence>
</complexType>
```

#### 6.8.2 Semantics

Semantics of the `SemanticFieldType`:

| *Name* | *Definition* |
|---|---|
| `SemanticFieldType` | Specifies the representation of a semantic field within the query format. A semantic field addresses a node (subject/object) according to the RDF data model (triple). The syntax is similar to the SPARQL query language and starts with a question mark and a descriptive identifier or a specific URI for the variable. |
| `Var` | Defines a place holder for the subject of a RDF triple. The place holder must start with a question mark and a descriptive identifier or a specific URI. |

### 6.8.3 Example

The main purpose of the SemanticFieldType (the variable element) is the addressing of RDF nodes to be used in sorting and comparison expressions. For instance, one want to have the result set sorted after a specific node information or want to filter the result set according to specific literal values. An example might be the price of a hotel which is modeled like hotel->hasPrice->value (e.g., Bloom->hasPrice->80 which results to a requesting triple: ?hotel -> hasPrice -> ?price), then one want to filter all hotels whose price is above a certain threshold. In this case, the GreaterThan condition (comparison expression already defined in MPQF) is used to filter the identified literal to a specific bound.

```
<MpegQuery mpqfID="">
    <Query>
        <Input>
            <QueryCondition>
                <Condition xsi:type="AND">
                    <Condition xsi:type="SemanticRelation" anchor="true">
                        <Subject>?hotel</Subject>
                        <Property>hasPrice</Property>
                        <Object>?price</Object>
                    </Condition>
                    <Condition xsi:type="GreaterThan">
                        <SemanticArithmeticField>
                            <Var>?price</Var>
                        </SemanticArithmeticField>
                        <DoubleValue>100.0</DoubleValue>
                    </Condition>
                </Condition>
            </QueryCondition>
        </Input>
    </Query>
</MpegQuery>
```

*In 8.1 Introduction, add the following paragraph:*

[…]
Its structure is defined by the `QFDeclarationType` type, which consists of a sequence of `DeclaredField` and `Resource` elements **and Prefix information declaring used namespaces**. A `DeclaredField` element allows declaring a reusable data path within an item's metadata. A `Resource` element allows declaring a reusable resource description using one of the subtypes of the `ResourceType` (the `MediaResourceType` type or the `DescriptionResourceType` type). **A `Prefix` element allows declaring a shortcut for a namespace which is used in semantic expressions.**

*In 8.2 Syntax, add the following (highlighted):*

```
<complexType name="QFDeclarationType">
  <sequence>
    <element name="DeclaredField" type="mpqf:DeclaredFieldType"
      minOccurs="0" maxOccurs="unbounded"/>
    <element name="Resource" type="mpqf:ResourceType" minOccurs="0"
      maxOccurs="unbounded"/>
    <element name="Prefix" minOccurs="0" maxOccurs="unbounded">
      <complexType>
          <attribute name="name" type="string" use="required" />
          <attribute name="uri" type="anyURI" use="required" />
      </complexType>
    </element>
```

```
  </sequence>
  <!-- Declaration of entities that can be reused in OutputDescription   or
QueryCondition -->
</complexType>
```

### 8.3 Semantics

Semantics of the `Prefix` element:

| *Name* | *Definition* |
|---|---|
| `Prefix` | Defines the name and the unique URI of a namespace, e.g. *http://www.w3.org/1999/02/22-rdf-syntax-ns#* for RDF. |
| `name` | Defines the name or shortcut of the namespace. For instance: *xsd*. |
| `uri` | Defines the assigned URI for this prefix and shortcut. For instance: *http://www.w3.org/2001/XMLSchema#* |

*In 9.2 Syntax, add the following (highlighted):*

```
<complexType name="OutputDescriptionType">
  <sequence>
    <element name="ReqField" type="mpqf:FieldType" minOccurs="0"
             maxOccurs="unbounded"/>
    <element name="ReqAggregateID" type="IDREF" minOccurs="0"
             maxOccurs="unbounded"/>
    <!-- Semantic Integration by MD -->
    <element name="ReqSemanticField" type="string" minOccurs="0"
             maxOccurs="unbounded"/>
    <element name="GroupBy" minOccurs="0">
      <complexType>
        <sequence>
          <element name="GroupByField" type="mpqf:FieldType"
                   maxOccurs="unbounded"/>
          <element name="Aggregate"
                   type="mpqf:AggregateExpressionType"
                   minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="SortBy" type="mpqf:AbstractSortByType"
             minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="maxPageEntries" type="positiveInteger"
             use="optional"/>
  <attribute name="maxItemCount" type="positiveInteger"
             use="optional"/>
  <attribute name="freeTextUse" type="boolean" use="optional"/>
  <attribute name="thumbnailUse" type="boolean" use="optional"/>
  <attribute name="mediaResourceUse" type="boolean" use="optional"/>
  <attribute name="outputNameSpace" type="anyURI" use="optional"/>
  <attribute name="distinct" type="boolean" use="optional"
             default="false"/>
</complexType>
```

*In 9.3 Semantics, add the following at the end of the table:*

Enhanced Semantics of the `OutputDescriptionType`:

| *Name* | *Definition* |
|---|---|
| `ReqSemanticField` | Describes the place holder of the desired semantic variable which is filtered in the Semantic Expression part of the QueryCondition. The place holder follows the SPARQL syntax beginning with a question mark and a descriptive identifier: e.g. ?person |

*In 9.5.2, add the following:*

### 9.5.2 Syntax

```
<complexType name="SortByFieldType">
  <complexContent>
    <extension base="mpqf:AbstractSortByType">
      <choice>
       <element name="Field" type="mpqf:FieldType"/>
       <element name="SemanticField" type="mpqf:SemanticFieldType"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

*In 9.5.3, add the following:*

### 9.5.3 Semantics

| *Name* | *Definition* |
|---|---|
| `SemanticField` | Describes the place holder of the desired semantic variable which should be used for sorting. The semantic field is filtered in the Semantic Expression part of the QueryCondition or stems from the SPARQL query type. The place holder follows the SPARQL syntax beginning with a question mark and a descriptive identifier: e.g. ?person |

*After 11.9, add the following subclause:*

## 11.10 Semantic Expression Types

### 11.10.1 Introduction

### 11.10.2 Syntax

```
<complexType name="SemanticExpressionType" abstract="true">
  <complexContent>
    <extension base="mpqf:BooleanExpressionType">
      <attribute name="anchorDistance"
                  type="nonNegativeInteger" use="required" />
      <attribute name="anchor" type="boolean" use="optional" />
    </extension>
  </complexContent>
</complexType>

<complexType name="SubClassOf">
  <complexContent>
    <extension base="mpqf:SemanticExpressionType">
       <attribute name="var" type="string"/>
       <attribute name="class" type="string"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="TypeOf">
  <complexContent>
    <extension base="mpqf:SemanticExpressionType">
       <attribute name="var" type="string"/>
       <attribute name="class" type="string"/>
     </extension>
  </complexContent>
</complexType>
<complexType name="EquivalentClass">
  <complexContent>
    <extension base="mpqf:SemanticExpressionType">
       <attribute name="var" type="string"/>
       <attribute name="class" type="string"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="ComplementOf">
  <complexContent>
    <extension base="mpqf:SemanticExpressionType">
       <attribute name="var" type="string"/>
       <attribute name="class" type="string"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="IntersectionOf">
  <complexContent>
    <extension base="mpqf:SemanticExpressionType">
       <attribute name="var" type="string"/>
       <attribute name="class" type="string"/>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="UnionOf">
  <complexContent>
   <extension base="mpqf:SemanticExpressionType">
      <attribute name="var" type="string"/>
      <attribute name="class" type="string"/>
   </extension>
  </complexContent>
</complexType>
<complexType name="InverseOf">
  <complexContent>
   <extension base="mpqf:SemanticExpressionType">
     <attribute name="var" type="string"/>
      <attribute name="class" type="string"/>
   </extension>
  </complexContent>
</complexType>
<complexType name="DisjointWith">
  <complexContent>
   <extension base="mpqf:SemanticExpressionType">
      <attribute name="var" type="string"/>
      <attribute name="class" type="string"/>
   </extension>
  </complexContent>
</complexType>

<complexType name="SemanticRelation">
  <complexContent>
   <extension base="mpqf:SemanticExpressionType">
     <sequence>
      <element name="Subject" type="string"/>
      <element name="Property" type="string"/>
      <element name="Object" type="string"/>
     </sequence>
   </extension>
  </complexContent>
</complexType>
```



### 11.10.3 Semantics

Semantics of the `SemanticExpressionType`:

| *Name* | *Definition* |
|---|---|
| `SemanticExpressionType` | An abstract type based on the BooleanExpressionType type that is the base type for defining semantic expressions. |
| `anchorDistance` | Defines the number of hops the identified triple is reachable through the RDF graph from the given starting RDF triple. The starting RDF triple is indicated by enabling the anchor flag. An anchorDistance of 0 indicates that the addressed node can be within any distance to the anchor node. |
| `anchor` | Defines a marker for a RDF triple within a query graph which serves as source for other triples. The main idea is to simplify the generation of semantic queries and to allow optimizers to improve resulting SPARQL queries. |

Semantics of the `class specific` types:

| *Name* | *Definition* |
|---|---|
| `SubClassOf` | A complex type based on the SemanticExpressionType type for defining a sub class relationship. This operation has two attributes that are required to represent the following pattern: var SubClassOf class. The var and class parameter can be URIs or place holder based on the SPARQL notation (starting with a question mark and a descriptive identifier). |
| `TypeOf` | A complex type based on the SemanticExpressionType type for defining a type of relationship. The attributes are defined as for the SubClassOf operation. |
| `EquivalentClass` | A complex type based on the SemanticExpressionType type for defining an equivalent class relationship. The attributes are defined as for the SubClassOf operation. |
| `ComplementOf` | A complex type based on the SemanticExpressionType type for defining a complement of relationship. The attributes are defined as for the SubClassOf operation. |
| `IntersectionOf` | A complex type based on the SemanticExpressionType type for defining an intersection of relationship. The attributes are defined as for the SubClassOf operation. |
| `UnionOf` | A complex type based on the SemanticExpressionType type for defining a union of relationship. The attributes are defined as for the SubClassOf operation. |
| `InverseOf` | A complex type based on the SemanticExpressionType type for defining an inverse of relationship. The attributes are defined as for the SubClassOf operation. |
| `DisjointWith` | A complex type based on the SemanticExpressionType type for defining a disjoint with relationship. The attributes are defined as for the SubClassOf operation. |

Semantics of the `SemanticRelationType` type:

| *Name* | *Definition* |
|---|---|
| `SemanticRelation` | Specifies the representation of a semantic relation within the query format. A semantic relation describes a triple in the RDF data model. The syntax is similar to the SPARQL query language and starts with a question mark and a descriptive identifier or a specific URI for the subject and object as well as a respective value representing the property. |
| `Subject` | Defines a place holder for the subject of a RDF triple. The place holder must start with a question mark and a descriptive identifier or a specific URI. |

| *Name* | *Definition* |
|---|---|
| `Property` | Defines the predicate (RDF notation) or property (OWL notation) of a RDF triple. The name space can be predefined in the QFDeclaration section. |
| `Object` | Defines a place holder for the object of a RDF triple.<br><br>The place holder must start with a question mark and a descriptive identifier or a specific URI. |

### 11.10.4 Usage guidance

#### 11.10.4.1 General

This subclause is an informative section.

#### 11.10.4.2 Linking structural (XML) and semantic data (RDF/XML)

Multimedia annotation comes in a large diversity. On the one side XML based metadata have been introduced (MPEG-7, etc.) which have strengthens in expressing structural information of the described multimedia content. For instance, low level features (like color descriptors) or temporal (spatial) behavior (video segments, image regions) are described by XML based annotations. On the other side, there are RDF based descriptions which support the annotation of high-level and semantic related annotations. Both concepts have advantages in their specific domain and drawbacks if they are used in the opposite domain (e.g., RDF for structural design and XML for semantic design).

To address both models in one search request, a linking between the two worlds need to be provided. The following paragraph describes such a possible linking for supporting MPQF requests that target on a global data set.

In order to accomplish a linkage between the extracted semantic concepts and its structured counterparts the following definition is introduced:

#### 11.10.4.3 Definition

Let A = CONCEPT#value the segment id value of a corresponding semantic concept expressing e.g. a temporal event description and B = //ELEMENT [@ID = value] the segment ID extracted via the given XPath expression of the structured description (e.g., the VideoSegment of an MPQF-7 description). Then, the linkage within the hybrid data model can be derived by the following Equi-Join:

$$E \otimes_{A=B} VS := \{e \in E \wedge vs \in VS \wedge e_A = vs_B\}$$

where E and V S are the semantic and structured annotations, respectively.

The definition foresees a unique key system that is accomplished by instantiating semantic and structured information (see Figure AMD2.1). Here, at the semantic side a unique URI containing a fragment identifier [4] can be attached to every instantiation of a concept (e.g., TemporalEntity). For instance at a TemporalEntity instance the following URI may be used: http://domain.tld/TemporalEntity#Segment_ID_1. Then, at the respective structured counterpart, an ID attribute is attached containing the same identifier value. In our example the MPEG-7 VideoSegment element would be instantiated with the following attribute id=SegmentID_1. By evaluating the given definition with value = SegmentID_1, the linkage of the corresponding information can be solved.

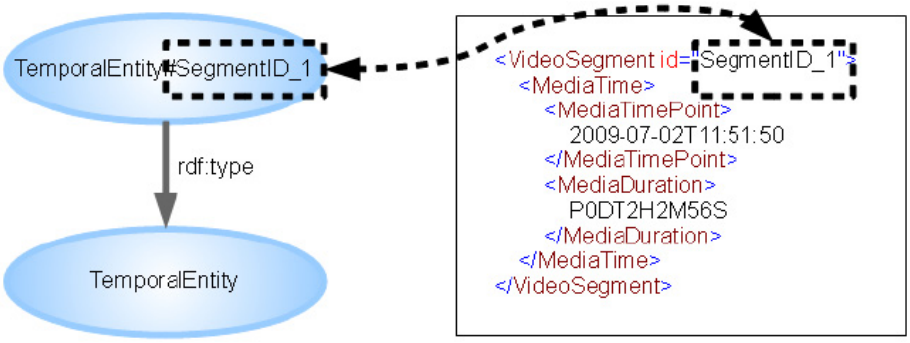


**Figure AMD2.1 — Linking Semantic and XML Data**

#### 11.10.4.4 EvaluationPath

The EvaluationPath allows focusing the scope of the query evaluation to a specific subset. For instance by using the MPEG-7 standard, one is able to concentrate on either Videos or VideoSegments during the evaluation. The scope is represented by an XPath pointing to the desired element e.g., //Video or //VideoSegment.

The use of the EvaluationPath for semantic retrieval is implicitly defined by the linkage to XML based metadata. Although it is possible to use XPath for RDF/XML data, it is not recommended as different mappings would result in different XPath expressions. By this, as defined, the focus should set by targeting on the structured data which is indented to provide a linking to the semantic descriptions. However, note, the exact linking between structured and semantic data is an implementation issue.

Furthermore, there are additional restrictions of the EvaluationPath already existing in the current version of the MPEG Query Format. First, the EvaluationPath is only as useful as an available XML Schema (the metadata description of the multimedia data) provides a deep hierarchy and deep descriptive level. In case there is only a flat description or no XML Schema available, the EvaluationPath is hardly beneficial. In addition, there also exist query types (e.g., QueryByMedia) where an EvaluationPath has no further meaning. A query using a single QueryByMedia condition targeting to an example image might not use the EvaluationPath at all.

#### 11.10.4.5 Evaluation of semantic conditions

The amendment for semantic enhancement to MPQF comes with two different ways of expressing semantic conditions. First of all, there is a QueryBySPARQL query condition which supports expressing SPARQL expressions. Those expressions are denoted to be executed by SPARQL engines, where the content of the QueryBySPARQL is restricted to an ASK SPARQL query. The evaluation processes (similar to the QueryByXQuery condition) the occurrence (matching) of the modeled subgraph in the RDF repository and results in values on a true/false basis.

Second, semantic filtering can be expressed by a set of semantic relations, expressions and comparison expressions enhanced by semantic means. This approach follows the same concept as the introduction of the Comparison expressions. Those expressions have been introduced in case that no full XQuery engine is available. In the same manner, the semantic relations can be used for the semantic case. In sum the semantic relations need to describe a subgraph for filtering against the semantic data set.

### 11.10.5 Example

The following example illustrates the use of semantic expressions on an imaginary semantic annotation in a surveillance system scenario. The example searches for persons and its location and time information (see OutputDescription) that matches the drawn RDF subgraph (see condition part). The example demonstrates the use of the integrated anchor and anchorDistance attributes. Here the first SemanticRelation-Condition defines the anchor meaning that all other specified conditions build the search graph pattern related to their given anchorDistance. The RDF subgraph assumes as data model the annotation of a closed (all persons are

known and can be tracked by specific sensors) video surveillance scenario containing locations (region of interests (ROI) or rooms) which are monitored by cameras. At runtime of the system, automatic events are triggered when persons are entering rooms or specific areas (ROI, e.g. a coffee area). In this context, the RDF subgraph and therefore the query searches for all persons that have been monitored entering a specific ROI which is controlled by a camera located in *Room240.* Besides the use of the anchor and anchorDistance attributes supports the ease of query creation by specifying one SemanticRelation as root (anchor is set to true) and all other relations are located within the distance specified by the anchorDistance flag. On the one side a user does not need to specify all relations and on the other side a possible query optimizer is able to produce an optimized subgraph. However, it has to be noted that query optimization is an implementation issue and that the semantic of the query has to be evaluated.

```
<MpegQuery mpqfID="">
  <Query>
    <Input>
      <QFDeclaration>
        <Prefix name="xsd"
                uri="http://www.w3.org/2001/XMLSchema#"/>
        <Prefix name="rdf"
                uri="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
        <Prefix name="ontology"
                uri="http://www.example.de/ontology#"/>
        <Prefix name="config"
                uri="http://www.example.de/configuration#"/>
      </QFDeclaration>
      <OutputDescription>
        <ReqSemanticField>?person</ReqSemanticField>
        <ReqSemanticField>?location</ReqSemanticField>
        <ReqSemanticField>?temp</ReqSemanticField>
      </OutputDescription>
      <QueryCondition>
        <Condition xsi:type="AND">
          <Condition          xsi:type="SemanticRelation"          anchor="true"
anchorDistance="0">
            <Subject>?person</Subject>
            <Property>ontology:hasName</Property>
            <Object>?name</Object>
          </Condition>
          <Condition xsi:type="SemanticRelation" anchorDistance="3">
            <Subject>?cam</Subject>
            <Property>ontology:isLocated</Property>
            <Object>?location</Object>
          </Condition>
          <Condition xsi:type="SemanticRelation" anchorDistance="3">
            <Subject>?location</Subject>
            <Property>ontology:hasName</Property>
            <Object>"Room240"^^xsd:string</Object>
          </Condition>
        </Condition>
      </QueryCondition>
    </Input>
  </Query> </MpegQuery>
```

*After 12.13, add the following subclause:*