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>
```