
**Information technology — Multimedia
content description interface —**

**Part 12:
Query format**

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

Partie 12: Format de requête



Reference number
ISO/IEC 15938-12:2012(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2012



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Page

Foreword	v
Introduction.....	vi
1 Scope	1
1.1 Organization of the document	1
1.2 Overview of the Query Format	1
2 Normative references	1
3 Terms, definitions, abbreviated terms and conventions.....	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	3
3.3 Conventions	3
4 Structure and Data Model	5
4.1 Structure.....	5
4.2 Data Model	8
5 Root Element	8
5.1 Introduction.....	8
5.2 Syntax	9
5.3 Semantics.....	10
5.4 Example.....	10
6 Datatypes	11
6.1 MediaLocatorType.....	11
6.2 MimeType	13
6.3 XPathType	13
6.4 ZeroToOneType	14
6.5 TermType	14
6.6 ClassificationScheme	15
6.7 RelationType	16
6.8 SemanticFieldType.....	17
7 Input Query Format	18
7.1 Introduction.....	18
7.2 Syntax	19
7.3 Semantics.....	19
7.4 Example.....	20
8 QFDeclaration	21
8.1 Introduction.....	21
8.2 Syntax	21
8.3 Semantics.....	22
8.4 Example	24
9 Output Description	25
9.1 Introduction.....	25
9.2 Syntax	25
9.3 Semantics.....	26
9.4 Example.....	28
9.5 SortBy.....	30
10 Query Condition	33
10.1 Introduction.....	33
10.2 Syntax	33
10.3 Semantics.....	33

10.4	Example	34
11	Expression Types	35
11.1	Introduction	35
11.2	Syntax	35
11.3	Semantics	35
11.4	Operands	36
11.5	Boolean Expression Types	38
11.6	Comparison Expression Types	42
11.7	Arithmetic Expression Types	47
11.8	Aggregate Expression Types	50
11.9	String Expression Types	52
11.10	Semantic Expression Types	54
12	Query Types	60
12.1	Introduction	60
12.2	Syntax	60
12.3	Semantics	60
12.4	QueryByMedia	60
12.5	QueryByDescription	62
12.6	QueryByFreeText	63
12.7	QueryByFeatureRange	65
12.8	SpatialQuery	69
12.9	TemporalQuery	72
12.10	Query By XQuery	74
12.11	Query By Relevance Feedback	75
12.12	Query By ROI	76
12.13	Join Query	80
12.14	QueryBySPARQL	82
13	Output Query Format	84
13.1	Output	84
13.2	ResultItem	86
13.3	SystemMessageType	89
14	Query Management Tools	90
14.1	Introduction	90
14.2	InputManagementType	90
14.3	OutputManagementType	95
15	MPEG Query Format Reference Software	98
15.1	Introduction	98
15.2	MPQF Reference Software specific terms, definitions and conventions	98
15.3	Overview of the architecture of the 15938-12 reference software	99
15.4	MPQF Validator	100
15.5	MPQF Parser	107
15.6	Basic Interpreter	112
15.7	MPQF Semantic Enhancement	118
Annex A (normative)	System Messages	122
Annex B (normative)	Classification Scheme	124
Bibliography		153

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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

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

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

This second edition cancels and replaces the first edition (ISO/IEC 15938-12:2008), which has been technically revised.

It also incorporates ISO/IEC 15938-12:2008/Cor.1:2009, ISO/IEC 15938-12:2008/Cor.2:2010, ISO/IEC 15938-12:2008/Amd.1:2011 and ISO/IEC 15938-12:2008/Amd.2:2011.

ISO/IEC 15938 consists of the following parts, under the general title *Information technology — Multimedia content description interface*:

- *Part 1: Systems*
- *Part 2: Description definition language*
- *Part 3: Visual*
- *Part 4: Audio*
- *Part 5: Multimedia description schemes*
- *Part 6: Reference software*
- *Part 7: Conformance testing*
- *Part 8: Extraction and use of MPEG-7 descriptions*
- *Part 9: Profiles and levels*
- *Part 10: Schema definition*
- *Part 11: MPEG-7 Profile schemas*
- *Part 12: Query format*

Introduction

The MPEG-7 standard, also known as the "Multimedia Content Description Interface", aims at providing standardized core technologies allowing the description of audiovisual data content in multimedia environments. This is a challenging task given the broad spectrum of requirements and targeted multimedia applications, and the broad number of audiovisual features of importance in such a context. In order to achieve this broad goal, MPEG-7 standardizes:

- Datatypes that are description elements not specific to the audiovisual domain that corresponds to reusable basic types or structures employed by multiple Descriptors and Description Schemes.
- Descriptors (D) to represent Features. Descriptors define the syntax and the semantics of each feature representation. A Feature is a distinctive characteristic of the data, which signifies something to somebody. It is possible to have several descriptors representing a single feature, i.e. to address different relevant requirements. A Descriptor does not participate in many-to-one relationships with other description elements.
- Description Schemes (DS) to specify the structure and semantics of the relationships between their components, which may be both Ds and DSs. A Description Scheme shall have descriptive information and may participate in many-to-one relationships with other description elements.
- A Description Definition Language (DDL) to allow the creation of new DSs and, possibly, Ds and to allow the extension and modification of existing DSs.
- Systems tools to support multiplexing of descriptions or description and data, synchronization issues, transmission mechanisms, file format, etc.

The standard is subdivided into twelve parts:

1. Systems: Architecture of the standard, tools that are needed to prepare MPEG-7 Descriptions for efficient transport and storage, and to allow synchronization between content and descriptions. Also tools related to managing and protecting intellectual property.
2. Description Definition Language: Language for defining new DSs and eventually also new Ds, binary representation of DDL expressions.
3. Visual: Visual description tools (Ds and DSs).
4. Audio: Audio description tools (Ds and DSs).
5. Multimedia Description Schemes: Description tools (Ds and DSs) that are generic, i.e. neither purely visual nor purely audio.
6. Reference Software: Software implementation of relevant parts of the MPEG-7 Standard.
7. Conformance: Guidelines and procedures for testing conformance of MPEG-7 implementations.
8. Extraction and use of MPEG-7 descriptions.
9. Profiles and Levels.
10. Schema Definition.

11. MPEG-7 Profile Schemas.

12. Query Format.

This part of ISO/IEC 15938 contains the tools of the MPEG Query Format (MPQF). It addresses the normative aspects of the MPQF and also illustrates some non-normative examples. The syntax of the Query Format is defined using the guidelines of DDL ISO/IEC 15938-2.

Information technology — Multimedia content description interface —

Part 12: Query format

1 Scope

1.1 Organization of the document

This part of ISO/IEC 15938 describes the query format tools which may be used independently or in combination with other parts of ISO/IEC 15938. Each query format tool is described in two normative sections:

- Syntax: Normative specification of the query and management format.
- Semantic: Normative definition of the semantics of all the components of the corresponding query format specification.

In some instances the query format level tool is also described using either one or two informative sections:

- Examples: Optionally an informative section dealing with examples is appended.
- Definitions: Optionally an informative section dealing with definitions is appended.

1.2 Overview of the Query Format

The query format provides a standardized interface for multimedia content information retrieval systems (e.g. MPEG-7 databases) in three aspects which are input query format, output query format, and query managements. The input query format specifies the interface through which the users can describe their search criteria with a set of precise input parameters in addition to a set of preferred output parameters to depict the return result sets. The output query format specifies the interface format for the result set. The query management provides means for selecting services (e.g. MPEG-7 database) or aggregated services (e.g. service provider that administers a set of different services) based on service properties (e.g. supported query format).

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.

XQuery 1.0 and XPath 2.0 Data Model (XDM). W3C Recommendation, 23 January 2007. <http://www.w3.org/TR/xpath-datamodel/>

XML Path Language (XPath) 2.0. W3C Recommendation, 23 January 2007. <http://www.w3.org/TR/xpath20/>

3 Terms, definitions, abbreviated terms and conventions

For the purposes of this document, the following terms and definitions, abbreviated terms and conventions apply.

3.1 Terms and definitions

3.1.1

content

data and the associated **metadata**

3.1.2

evaluation item

EI

unit, against which the query condition is tested

NOTE By default, an EI is a **multimedia content** of the multimedia repository, but other types of EI are also possible. An EI can be:

- 1) A **multimedia content**;
- 2) A **segment** of a **multimedia resource**;
- 3) An **XPath-item** related to the **multimedia content's metadata** XML tree.

3.1.3

input query format

interface format going from a requester to one or more responders with two functionalities

- functionality providing a combination of syntax and semantics of the interface, through which the requester assigns search criteria and associated data;
- functionality providing syntax and semantics of the interface, through which the requester specifies the format of the result data

NOTE The second functionality of the input query format provides tools by which the requester can express desired output format that should be conformant to the specification of **output query format**.

3.1.4

metadata

data expressed as a schema valid XML instance to carry additional information describing a **multimedia resource**, where the schema defines the information model of the data

3.1.5

multimedia content

coded representation of the information contained in or related to a **multimedia resource** in a formalized manner suitable for interpretation by human means

3.1.6

multimedia resource

URI identifiable portion of raw data of a video, an image, an audio or text in any format, that is associated with a MIME Content-Type

3.1.7

output query format

interface format going from the responder to the requester as a response to the request specified by the **input query format**

NOTE Output query format defines all possible structures of return from responder to the requester. The structure of an actual return shall be decided by OutputDescription element in **input query format**.

3.1.8**query management tools**

tools to support the functionality required to manage the query transaction between the requesters and the responders

NOTE The query management tools do not include tools that are supported by network protocols. The query management tools intend to be network agnostic and media agnostic.

3.1.9**segment**

spatial, temporal, or spatio-temporal unit of multimedia, for example, a temporal segment of video, or a spatial segment of an image

3.1.10**XPath-item**

either a node from the **multimedia content's metadata** XML tree or an atomic value

NOTE Details about the different types of nodes and atomic values can be found in the W3C Recommendation on XQuery 1.0 and XPath 2.0 Data Model. An XPath-item of a **multimedia content's metadata** may or may not be related to a **multimedia content's segment**. Also, a **multimedia content** or a **multimedia content's segment** may or may not be related to XML **metadata**. Within MPQF queries, XPath can be used to select a sequence of **multimedia content's segments** and/or **metadata** XPath-items. According to the W3C Recommendation on XQuery 1.0 and XPath 2.0 Data Model, a sequence is an ordered collection of zero or more XPath-items.

3.2 Abbreviated terms

MPEG: Moving Picture Experts Group

MPEG-7: ISO/IEC 15938

MPQF: MPEG Query Format, ISO/IEC 15938-12

URI: Uniform Resource Identifier (IETF Standard is RFC 2396)

URL: Uniform Resource Locator (IETF Standard is RFC 2396)

XML: Extensible Markup Language (W3C, <http://www.w3.org/XML/>)

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/>

3.3 Conventions**3.3.1 Query and query management tools**

This part of ISO/IEC 15938 specifies the format for query and query management tools using XML-Schema.

- Query – A set of tools supporting the definition of the query request as well as the query response defined for the MPEG query format. The structure defined for the `Query` element provides a container for input query format or output query format.
 - Input query format – `Input` and `FetchResult` elements are defined for the query. The input query format structure provides a container for a query request. Such a request should contain a set of conditions and/or the output description which specifies the structure and content of the output query format and/or a set of declarations.

- Output query format – The `Output` element is defined for the query. It provides a container for all the responses from a responder to a requester. It may contain not only query results but also any messages such as error and exceptions.
- Management tools – A set of tools for the query management defined for the MPEG query format including service discovery, querying service capability, and service capability description. The structure defined for the management tools provides a container for input management tools or the output management tools.
 - Input management tools – An `Input` element is defined for the management tools intended to be sent from a requester to one or more responders.
 - Output management tools – An `Output` element is defined for the management tools intended to be sent from a responder to one or more requesters.

3.3.2 Naming convention

In order to specify tools for the query format, this part of ISO/IEC 15938 uses constructs provided by the language specified in ISO/IEC 15938-2 [1], such as "element", "attribute", "simpleType" and "complexType". The names associated to these constructs are created on the basis of the following conventions:

- If the name is composed of various words, the first letter of each word is capitalized. The rule for the capitalization of the first word depends on the type of construct and is described below.
- Element naming: the first letter of the first word is capitalized (e.g. `TimePoint` element of `TimeType`).
- Attribute naming: the first letter of the first word is not capitalized (e.g. `timeUnit` attribute of `IncrDurationType`).
- `complexType` naming: the first letter of the first word is capitalized, the suffix "Type" is used at the end of the name, except the concrete types inherited from abstract types.
- `simpleType` naming: the first letter of the first word is not capitalized, the suffix "Type" may be used at the end of the name (e.g. `timePointType`).

NOTE The full name of the `complexType` or `simpleType` is used when referencing a `complexType` or `simpleType` in the definition of the query format.

3.3.3 Documentation convention

The syntax of each datatype, descriptor and description scheme is specified using the constructs provided by ISO/IEC 15938-2 [1], and is shown in this part of ISO/IEC 15938 using a specific font and background:

```
<complexType name="ExampleType">
  <sequence>
    <element name="Element1" type="string" minOccurs="1" maxOccurs="1"/>
  </sequence>
  <attribute name="attribute1" type="string" use="default" value="attrvalue1"/>
</complexType>
```

The semantics of each datatype, descriptor and description scheme is specified using a table format, where each row contains the name and a definition of a type, element or attribute:

<i>Name</i>	<i>Definition</i>
ExampleType	Specifies an ...
element1	Describes the ...
attribute1	Describes the ...

Non-normative examples are included in separate sections, and are shown in this part of ISO/IEC 15938 using a separate font and background:

```
<Example attribute1="example attribute value">
  <Element1>example element content</Element1>
</Example>
```

Moreover, the schema defined in this part of ISO/IEC 15938 follows a type-centric approach. As a result, almost no elements (in the XML schema sense) are defined. Most of the description tools are specified only by defining a complexType or a simpleType. In order to create a description, it has to be assumed that an element of a given type (complexType or simpleType) has been declared somewhere in the schema, for example as a member of another complexType or simpleType.

The examples in the informative sections assume that the following declaration has been made:

```
<element name="Example" type="mpqf:MyToolType">
```

Therefore, the example shown above is a valid description.

3.3.4 Wrapper convention of the schema

The Syntax defined in this part of ISO/IEC 15938 assumes the following schema wrapper.

```
<schema xmlns:mpqf="urn:mpeg:mpqf:schema:2008"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:mpeg:mpqf:schema:2008" elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

4 Structure and Data Model

4.1 Structure

The Query Format defines the interface between a requester and a responder. The requester may be any requester and the according responder might be the service provider, which can in turn additionally be a requester to a number of other databases. Additionally, one or more service providers can utilize the Query Format acting as responder from the requester side and as requester, when forwarding the messages to a number of databases. Furthermore, the service provider may have the capability to aggregate the results from different databases and to reply a combined result to the requester. Figure 1 depicts a possible setup for the use of the MPEG Query Format.

The part of the query format related to the messages from the requester to the responder, describing their search criteria, is called Input Query Format; the part of the query format related to the messages in return is called Output Query Format. Additionally, the format for the management messages provides means for selecting services (e.g., MPEG-7 database) or aggregated services (e.g., service provider that administers a set of different services) based on service properties (e.g., supported query format, etc.).

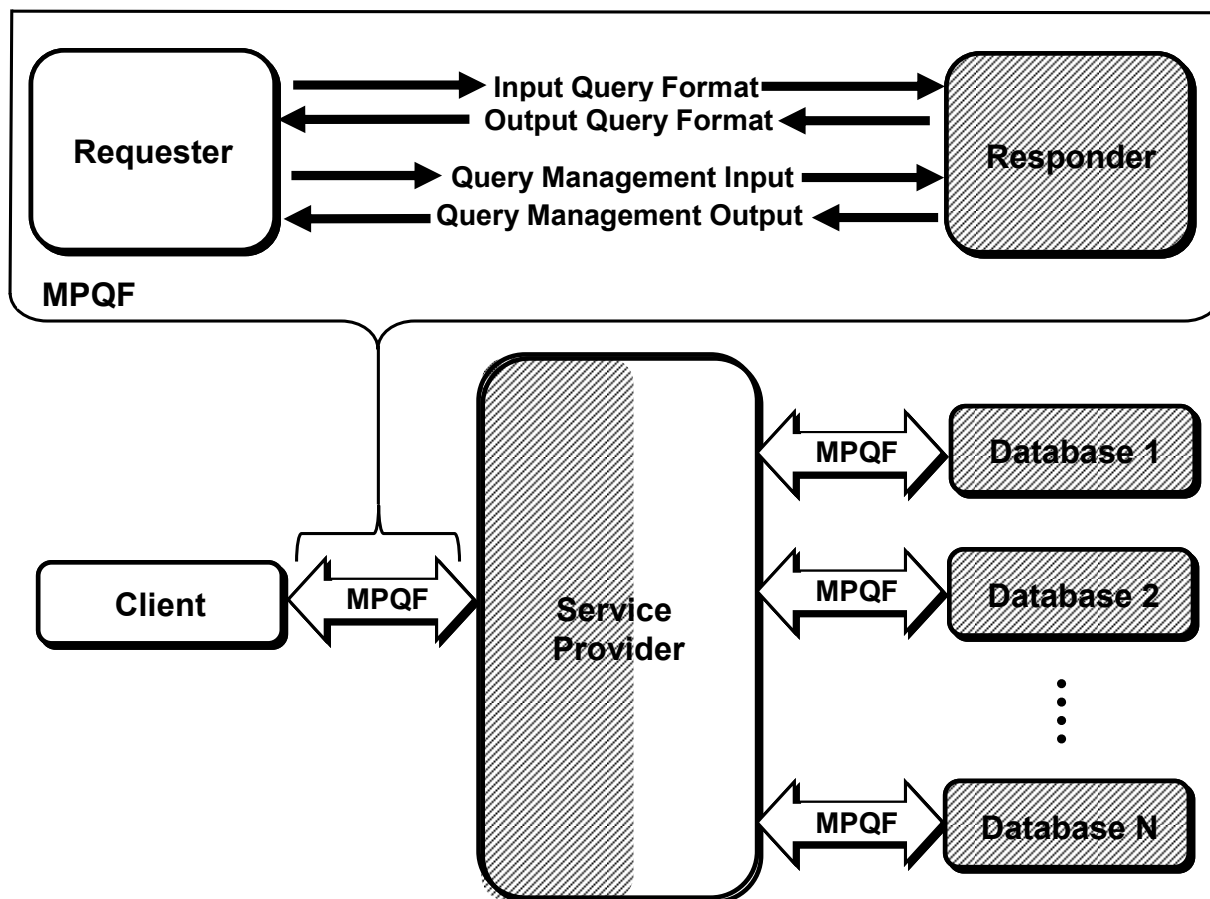


Figure 1 — Possible scenario for the use of the MPEG Query Format

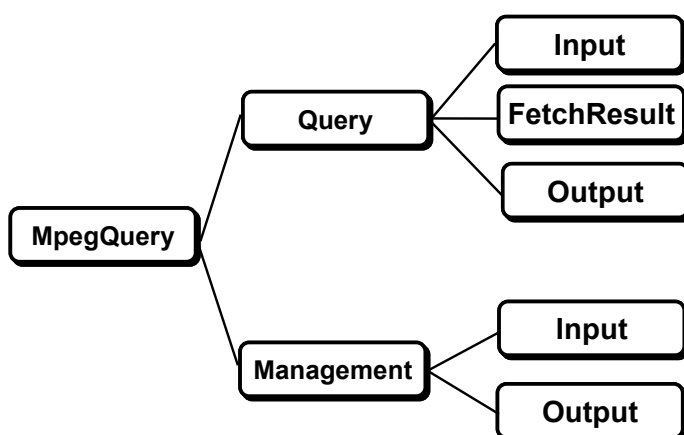


Figure 2 — Schema overview of the uppermost elements of the MPEG Query Format

The root element of the MPEG Query Format is named “MpegQuery” and it contains a sequence of `Query` and `Management` elements. Table 1 explains both elements in more detail:

Table 1 — Query and Management elements

<code>Query</code>	A set of tools are defined for the query and response for the MPEG Query Format. The structure defined for the <code>Query</code> element provides a container for Input Query Format or Output Query Format.
<code>Management</code>	A set of tools for the query management defined for the MPEG Query Format including service discovery, querying service capability, and service capability description. The structure defined for the <code>Management</code> element provides a container for Management Input Tools or the Management Output Tools.

The `Query` element of the MPEG Query Format defines a set of tools for the query and its response. Table 2 depicts the containing elements in more detail. Note that the `Input` and the `FetchResult` elements belong to the Input Query Format, which provides a container for describing a query request (e.g., the query condition and/or the output description which specifies the structure and content of the output query format).

Table 2 — Elements in Query Tools

<code>Input</code>	The <code>Input</code> element is part of the Input Query Format. Its structure provides a container for describing a query request. Such a request can consist of a query condition and/or the output description which specifies the structure and content of the output query format and/or some declaration parts.
<code>FetchResult</code>	The <code>FetchResult</code> element is also part of the Input Query Format. It allows the user to request the results of a previous query issued using e.g., the asynchronous mode.
<code>Output</code>	The <code>Output</code> element describes the Output Query Format. It provides a container for all the responses from a responder to a requester. It may contain not only query results but also any messages such as errors, exceptions or comments.

The `Management` element of the MPEG Query Format represents the Query Management Tools of the Query Format. It describes a set of tools for the query management including service discovery, querying service capability, and service capability description. Similar to the query tools (Input Query Format and Output Query Format), the management part distinguishes the tools for the request (Input Management Type) and the response (Output Management Type). Table 3 depicts the containing elements in more detail.

Table 3 — Elements in Query Management Tools

<code>Input</code>	The <code>Input</code> element, which is defined for the management tools, is intended to be sent from a requester to one or more responders.
<code>Output</code>	The <code>Output</code> element, which is defined for the management tools, is intended to be sent from a responder to one requester.

4.2 Data Model

Processing and evaluation of an MPQF query is executed against one or more multimedia repositories. Note, all introduced terms used within this subclause are explained in 3.1.

From the point-of-view of the MPQF, a multimedia repository is an unordered set of multimedia contents (MCs). An MPQF query specifies a search condition tree composed by conditions which may be interconnected by Boolean expressions. Conditions within the condition tree operate on one evaluation-item (EI) at a given time, or two EIs if a Join operation is used. By default, an evaluation-item (EI) is a multimedia content in the multimedia repository, but other types of EIs are also possible. An EI can be either a multimedia content (MC), a segment of a multimedia resource, or an XPath-item related to the MC's metadata XML tree. A segment of a multimedia content is a spatial and/or temporal unit of multimedia resource, for example, a temporal segment of a video, or a segment of an image.

According to the XQuery 1.0 and XPath 2.0 Data Model (XDM), an XPath-item is either a node from the MC's metadata XML tree or an atomic value. Details about the different types of nodes and atomic values can be found in the W3C Recommendation of the XQuery 1.0 and XPath 2.0 Data Model (XDM). An XPath-item of an MC's metadata may or may not be related to an MC's segment. A multimedia content or an MC's segment may or may not be related to XML metadata. Within MPQF queries, XPath can be used to select a sequence of MC's segments and/or metadata XPath-items.

The scope of query evaluation and the granularity of the result set shall be determined by the `EvaluationPath` element specified in the `QueryCondition` element. For example, if the `EvaluationPath` element points to a certain level of a video segment, then the result set is a collection of video segments with the requested metadata. If the `EvaluationPath` element points to a video resource, then the result set consists of a collection of video resources with the requested metadata. If the `EvaluationPath` element in the `QueryCondition` element is not specified, the output result is provided as a collection of multimedia contents, as stored in the repository, each satisfying the query condition.

XPath expressions appearing in MPQF can be absolute or relative. When absolute, independently of the kind of evaluation item being addressed, an XPath expression will always refer to the root of the metadata document related to the multimedia content which the evaluation-item belongs to (even if the evaluation-item is related just to a fragment of this document).

When relative, an XPath expression will always refer to the root node of the metadata fragment (which can be the root of the metadata document or cannot) related to the evaluation item being addressed.

5 Root Element

5.1 Introduction

The `MpegQuery` element serves as the root element of the MPEG Query Format. The root element shall be used as the topmost element in all messages transmitted. This applies on the one side to the input query format and the output query format of a query request/response as well as on the other side to the query management messages of the input/output.

5.2 Syntax

```

<element name="MpegQuery" type="mpqf:MpegQueryType"/>

<complexType name="MpegQueryType">
  <choice>
    <element name="Query">
      <complexType>
        <choice>
          <element name="Input" type="mpqf:InputQueryType"/>
          <element name="FetchResult">
            <complexType>
              <attribute name="queryID" type="anyURI"/>
              <attribute name="retrievePageNum" type="positiveInteger"/>
            </complexType>
          </element>
          <element name="Output" type="mpqf:OutputQueryType"/>
        </choice>
      </complexType>
    </element>
    <element name="Management">
      <complexType>
        <choice>
          <element name="Input" type="mpqf:InputManagementType"/>
          <element name="Output" type="mpqf:OutputManagementType"/>
        </choice>
      </complexType>
    </element>
  </choice>
  <attribute name="mpqfID" type="anyURI" use="required"/>
</complexType>

```

5.3 Semantics

Semantics of the `MpegQueryType` type:

Name	Definition
MpegQuery	Serves as the root element of the MPEG Query Format. The MpegQuery element shall be used as the topmost element in any instance of MPEG Query Format.
MpegQueryType	Specifies the syntax of the root element. Within this element, either one of the Input, FetchResult or Output element of the Query or one of the Input or Output element of the Management element shall be instantiated, respectively.
Query	Wraps the user query request (an Input or a FetchResult element) or the responder response to a user query request (the Output element).
Input	Wraps the user query specified using the Input Query Format, which is to be sent from the requester to the responder.
FetchResult	Allows the user to request the result of a previous query. In case of a query request issued using the asynchronous mode (with the immediateResponse attribute of the InputQueryType type set to false), the FetchResult element shall be used to indicate the preparedness of the user to receive results. In case of a query request issued using the synchronous mode, the FetchResult element shall be used to receive individual pages of the result set (if paging is activated).
queryID	The requester is looking for a the result of a certain identifier in asynchronous mode. The attribute queryID specifies this identifier.
retrievePageNum	Specifies the number of page the requester wants the responder to return (optional) The default value is '1.'
Output	Wraps the responder response to a user query as described using the Output Query Format. The output is sent from the responder to the requester.
Management	Describes the Query Management tools which may be exchanged between responders and requesters. Within this element only one of Input or Output element shall be instantiated.
Input	Wraps a user request for service capabilities.
Output	Wraps the response to a user request for service capabilities.
mpqfID	Specifies a unique identifier which is assigned to every message sent between a requester and a responder.

5.4 Example

The following examples show the use of the root element for the MPEG Query Format. The first three examples demonstrate the use of the `Query` element. First, a user query request is simulated by using the `Input` element. Note, that no details have been specified. Second, the result of a previous asynchronous request is going to be fetched by the user. Finally, the result set of a query request is demonstrated by using the `Output` element.

The last example shows the use of a service discovery request by using the `Management` element followed by an `Input` element. In this case the response would contain a list of all available services which are registered.

```

<MpegQuery mpqfID="idForMyRequest1">
  <Query>
    <Input>
      . . .
    </Input>
  </Query>
</MpegQuery>

```

```

<MpegQuery mpqfID="idForMyRequest2">
  <Query>
    <FetchResult queryID="idFromAsyncRequest1"/>
  </Query>
</MpegQuery>

```

```

<MpegQuery mpqfID="idForMyResponse1">
  <Query>
    <Output>
      . . .
    </Output>
  </Query>
</MpegQuery>

```

```

<MpegQuery mpqfID="idForMyRequest3">
  <Management>
    <Input>
      . . .
    </Input>
  </Management>
</MpegQuery>

```

6 Datatypes

6.1 MediaLocatorType

6.1.1 Introduction

The `MediaLocatorType` type serves as a basic datatype for locating a multimedia resource in the `MediaResource` element.

The `MediaLocatorType` type shall be composed of either a `MediaUri` element or `InlineMedia` element for locating a multimedia resource. The `InlineMedia` element enables an alternative method to a `MediaUri` for locating multimedia resource, by embedding the multimedia resource into the description.

6.1.2 Syntax

```
<complexType name="MediaLocatorType">
  <sequence>
    <choice>
      <element name="MediaUri" type="anyURI"/>
      <element name="InlineMedia" type="mpqf:InlineMediaType"/>
    </choice>
  </sequence>
</complexType>
<complexType name="InlineMediaType">
  <choice>
    <element name="MediaData16" type="hexBinary"/>
    <element name="MediaData64" type="base64Binary"/>
  </choice>
  <attribute name="type" type="mpqf:mimeType" use="required"/>
</complexType>
```

6.1.3 Semantics

Semantics of the `MediaLocatorType` type:

Name	Definition
<code>MediaLocatorType</code>	Specifies the <code>MediaLocatorType</code> type. Describes the location of a multimedia resource in <code>MediaResource</code> element. The location of a multimedia resource should be specified either by reference to the external multimedia resource using the <code>MediaUri</code> element or by embedding the multimedia resource within the description as an <code>InlineMedia</code> instance.
<code>MediaUri</code>	Points to the actual resource that is used for querying.
<code>InlineMedia</code>	Describes the actual resource that is used for querying.

Semantics of the `InlineMediaType` type:

Name	Definition
<code>InlineMediaType</code>	Specifies the multimedia resource embedded in the resource. The <code>InlineMediaType</code> type is a container for the actual binary multimedia resource.
<code>MediaData16</code>	Describes the actual resource that is used for querying, encoded as a textual string in base-16 format.
<code>MediaData64</code>	Describes the actual resource that is used for querying encoded as a textual string in base-64 format.
<code>Type</code>	This attribute specifies the <code>mimeType</code> of the multimedia resource.

6.2 mimeType

6.2.1 Introduction

The `mimeType` type is defined to specify the syntax for a `mimeType` element which shall be used to identify the media type of a multimedia resource.

6.2.2 Syntax

```
<simpleType name="mimeType">
  <restriction base="string">
    <whiteSpace value="collapse"/>
    <pattern value=' ([Xx]\-)?[!#$%&#x27;*+.0-9A-Z\^~&#x7f;]+/ ([Xx]\-
      )?[!#$%&#x27;*+.0-9A-Z\^~&#x7f;\-]+' />
  </restriction>
</simpleType>
```

6.2.3 Semantics

Semantics of the `mimeType` type:

Name	Definition
<code>mimeType</code>	Specifies the syntax of the <code>mimeType</code> type, which is used to identify the type of multimedia resource.
pattern value	Specifies the allowed pattern for <code>mimeType</code> s and corresponds to RFC 2045 MIME (Part 1).

6.3 XPathType

6.3.1 Introduction

XPath expressions are used to identify elements and attributes in the corresponding responder description (specified by the namespace of the description, e.g., MPEG-7) in Input Query or Output Query. The instance of `xPathType` shall be compliant to the specification XML Path Language (XPath) 2.0, W3C Recommendation 23 January 2007.

6.3.2 Syntax

```
<simpleType name="XPathType">
  <restriction base="token"/>
</simpleType>
```

6.3.3 Semantics

Semantics of the `xPathType` type:

Name	Definition
<code>xPathType</code>	Describes the valid XPath expression.

6.4 ZeroToOneType

6.4.1 Introduction

The `zeroToOne` datatype represents a float value in the range of 0.0 to 1.0.

6.4.2 Syntax

```
<simpleType name="zeroToOneType">
  <restriction base="float">
    <minInclusive value="0.0"/>
    <maxInclusive value="1.0"/>
  </restriction>
</simpleType>
```

6.4.3 Semantics

Semantics of the `zeroToOneType` type:

Name	Definition
<code>zeroToOneType</code>	Describes a floating value in the range of 0.0 to 1.0 inclusive.

6.5 TermType

6.5.1 Introduction

The `TermType` type is to be used in the classification scheme and other tools to control the use of the terms inside the query format. The `Name` element allows the specification of the label of the term. The `Description` element allows the definition and the description in text format. The `href` attribute is required to point to the classification scheme, in which the terms are defined. A hierarchy of nested terms can be expressed by successive applying of the `Term` element.

6.5.2 Syntax

```
<complexType name="TermType">
  <sequence>
    <element name="Name" type="string" minOccurs="0"/>
    <element name="Description" type="string" minOccurs="0"/>
    <element name="Term" type="mpqf:TermType" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
  <attribute name="href" type="mpqf:SimpleTermType" use="required"/>
</complexType>

<simpleType name="SimpleTermType">
  <restriction base="anyURI"/>
</simpleType>
```

6.5.3 Semantics

Semantics of the `TermType` type:

<i>Name</i>	<i>Definition</i>
<code>TermType</code>	Specifies a <code>TermType</code> type which is used for describing capabilities and its references to classification schemes.
<code>Name</code>	Specifies the name of the term.
<code>Description</code>	Specifies a detailed description of the term.
<code>Term</code>	Specifies a new term which supports nested term definitions.
<code>href</code>	Specifies the <code>SimpleTermType</code> type pointing to a specific term of a classification scheme.
<code>SimpleTermType</code>	Specifies the URN of the term.

6.5.4 Example

For the examples, check the classification schemes in Annex B.

6.6 ClassificationScheme

6.6.1 Introduction

The `ClassificationScheme` element is an element, which contains the classification scheme. It makes use of the `TermType` type, which allows a nested definition of terms.

6.6.2 Syntax

```
<element name="ClassificationScheme">
  <complexType>
    <sequence>
      <element name="Term" type="mpqf:TermType" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="uri" type="anyURI" use="required"/>
    <attribute name="domain" use="optional">
      <simpleType>
        <list itemType="mpqf:xPathType"/>
      </simpleType>
    </attribute>
  </complexType>
</element>
```

6.6.3 Semantics

Semantics of the `ClassificationScheme` element:

Name	Definition
<code>ClassificationScheme</code>	This element contains the actual classification scheme
<code>Term</code>	Describes the content of the classification scheme (the single terms including a description).
<code>Uri</code>	Identifies the classification scheme with a Uniform Resource Identifier (URI). This URI shall uniquely identify a classification scheme. The use of a URN as a classification scheme URI is preferred.
<code>Domain</code>	Indicates a set of description elements to which this <code>ClassificationScheme</code> applies. The set is specified using an XPath expression. This attribute is a hint on how the <code>ClassificationScheme</code> can be used in a description. One or more values may be specified. The XPath in this attribute is defined relative to a <i>description root</i>, not the DDL schema. For example: a value of <code>"../Classification/Genre"</code> for domain indicates that the <code>ClassificationScheme</code> can be used for "Genre" elements inside of all "Classification" elements in the description.

6.6.4 Example

See Annex B for classification scheme examples.

6.7 RelationType

6.7.1 Introduction

The `RelationType` type describes a relation among DS instances and controlled terms. The arguments of a relation are categorized into the source (first) and the target (second) arguments. Both the source and target arguments for a relation are ordered. A relation may have a variable number of arguments in each of the source and the target.

6.7.2 Syntax

```
<complexType name="RelationType">
  <sequence/>
  <attribute name="relationType" type="anyURI" use="required"/>
  <attribute name="sourceResource" type="IDREF" use="required"/>
  <attribute name="targetResource" type="IDREF" use="optional"/>
</complexType>
```

6.7.3 Semantics

Semantics of the `RelationType` type:

<i>Name</i>	<i>Definition</i>
RelationType	Specifies the RelationType type of a spatial or temporal relation.
relationType	Describes the type of the Relation. Depending on the used QueryType (SpatialQueryType, TemporalQueryType) the allowed classification scheme types differ. The allowed classification schemes are defined at the corresponding query types
sourceResource	Specifies the source element of the specific relation. The source element shall be defined in the declaration part of type DescriptionResourceType type and point to a valid description according to the used query type (e.g., for a SpatialQueryType type only a MPEG-7 StillImage or comparable (of other metadata standards) description is allowed)
targetResource	Specifies the target element of the specific relation. The target element shall be defined in the declaration part of type DescriptionResourceType type and point to a valid description according to the used query type (e.g., for a SpatialQueryType type only a MPEG-7 StillImage or comparable (of other metadata standards) description is allowed)

6.7.4 Example

See the SpatialQuery and TemporalQuery type for the examples.

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:

<i>Name</i>	<i>Definition</i>
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

Name	Definition
	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" anchorDistance="0">
            <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>
```

7 Input Query Format

7.1 Introduction

This clause specifies the `Input` element defined for the Input Query Format. The Input Query Format structure is defined by the `InputQueryType` type. It provides a container for describing a query request containing a set of conditions and/or the specification of the structure and content of the output query format and a declaration part. In general, this type provides all necessary constructs in order to express a multimedia query request to a set of databases.

7.2 Syntax

```
<complexType name="InputQueryType">
  <sequence>
    <element name="QFDeclaration" type="mpqf:QFDeclarationType" minOccurs="0"/>
    <element name="OutputDescription" type="mpqf:OutputDescriptionType"
minOccurs="0"/>
    <element name="QueryCondition" type="mpqf:QueryConditionType" minOccurs="0"/>
    <element name="ServiceSelection" type="mpqf:ServiceSelectionType"
      minOccurs="0"/>
  </sequence>
  <attribute name="previousAnswerID" type="anyURI" use="optional"/>
  <attribute name="immediateResponse" type="boolean" use="optional"
default="true"/>
  <attribute name="timeout" type="duration" use="optional"/>
</complexType>
<complexType name="ServiceSelectionType">
  <sequence>
    <element name="ServiceID" type="anyURI" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

7.3 Semantics

Semantics of the `InputQueryType` type:

Name	Definition
InputQueryType	Specifies the syntax of the Input element, which describes the input query request to be sent from the requester to the responder.
QFDeclaration	Defines resources that can be referenced for describing query conditions and output data fields in the QueryCondition or OutputDescription element, respectively. A resource is either a multimedia resource or a description resource. A multimedia resource is specified either by the inlined raw data of an image, video, etc. or a pointer (URL) to its location. The description resource contains information of low level (e.g., color) or high level (e.g, objects) features of an example media. The description can be given by any valid XML-based multimedia metadata schema. Furthermore, one has the possibility to declare specific fields (e.g., MPEG-7 Creator element) of any XML-based multimedia metadata schema in order to reuse this information within the OutputDescription element or the QueryCondition element.
OutputDescription	Defines the content as well as the structure of the returned value from the responder. Here, the requester has the possibility to select the desired information which the result set should contain. For instance, in case of searching for songs, one would like to know the song title, location, etc. Depending on the used multimedia metadata standard (of the target service) the XML elements can vary. Furthermore, by using absolute path entries, the requester has the possibility to force the structure of the returned value.
QueryCondition	Defines the search and filter criteria within a single query.
ServiceSelection	Specifies the set of services where the query should be evaluated. This element is supposed to be used in case of communicating with an aggregation service in order to signalize which subset of the available services should evaluate the request. The syntax is specified by the ServiceSelectionType type.

Name	Definition
previousAnswerID	Specifies a unique identifier to determine a query response already received (optional). This attribute is intended to be used in operations (e.g., relevance feedback) such as a search within the previous result set.
immediateResponse	Defines the mode of operation. The default value of this optional attribute is "true". When either the value is "true" or the attribute is not present, the requester expects to receive the response immediately (until timeOut). If the value of this attribute is "false", the requester is expected to receive only a "mpqfID" as response (asynchronous mode). In the asynchronous mode, the requester will come later (after the timeOut) to harvest the response (sending a message including the FetchResult element). In case of a given value for the expirationDate attribute, the result set is required to be cached. In case of no expirationDate attribute is given, it is an implementation issue.
timeOut	Specifies the temporal duration during which the responder can perform searching the relevant databases. In the immediate response mode of operation, it can be interpreted as a time limit within which an aggregation responder can wait for the individual service provider's response. In the asynchronous mode of operation, it can be interpreted as a way of specifying a time point, after which the requester intends to issue a FetchResult query.

Semantics of the ServiceSelectionType type:

Name	Definition
ServiceSelectionType	Specifies the syntax of the ServiceSelection element, which determines the set of services where the request should be sent to.
ServiceID	Defines one service by its ID of type anyURI. This information is the same as given by the service's capability description and denotes the service's entry point.

7.4 Example

Examples for the Input element are given in the following sub-clauses of QFDeclaration, OutputDescription and QueryCondition. An example for a service selection is presented below. The ServiceSelection element in the input element determines the set of databases that have been chosen for retrieval. The use of the ServiceSelection element assumes the existence of an aggregation service which takes care of on the one side distributing the query to the selected target databases and on the other side aggregating the individual result sets in a sophisticated manner. In case of a direct query request to a single database, the service selection element may be omitted.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <ServiceSelection>
        <ServiceID>http://www.test-retrieval.com</ServiceID>
        <ServiceID>http://www.test-retrievall.com</ServiceID>
      </ServiceSelection>
    </Input>
  </Query>
</MpegQuery>
```

8 QFDeclaration

8.1 Introduction

The optional `QFDeclaration` element allows declaring reusable definitions of data paths and resources that can be referenced multiple times within a query. 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.

8.2 Syntax

```
<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>

<complexType name="DeclaredFieldType">
  <simpleContent>
    <extension base="mpqf:xPathType">
      <attribute name="id" type="ID" use="required"/>
      <attribute name="typeName" type="string" use="optional"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="ResourceType" abstract="true">
  <attribute name="resourceID" type="ID" use="required"/>
</complexType>

<complexType name="MediaResourceType">
  <complexContent>
    <extension base="mpqf:ResourceType">
      <sequence>
        <element name="MediaResource" type="mpqf:MediaLocatorType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="DescriptionResourceType">
  <complexContent>
    <extension base="mpqf:ResourceType">
      <sequence>
        <element name="AnyDescription">
          <complexType mixed="true">
            <sequence>
              <any namespace="##any"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

        </complexType>
    </element>
</sequence>
</extension>
</complexContent>
</complexType>

```

8.3 Semantics

Semantics of the `QFDeclarationType` type:

Name	Definition
<code>QFDeclarationType</code>	Specifies the syntax of the <code>QFDeclaration</code> element, which allows declaring reusable definitions of data paths and resources that can be referenced multiple times within a query.
<code>DeclaredField</code>	Allows declaring a reusable data path within an item's metadata. Its structure is defined by the <code>DeclaredFieldType</code> type.
<code>Resource</code>	Allows declaring a reusable resource description using one of the subtypes of the abstract <code>ResourceType</code> type (the <code>MediaResourceType</code> type or the <code>DescriptionResourceType</code> type).

Semantics of the `Prefix` element:

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

Semantics of the `DeclaredFieldType` type:

Name	Definition
<code>DeclaredFieldType</code>	Complex type which allows declaring a reusable data path within an item's metadata the same way as the <code>FieldType</code> type does, but adding an ID that can be referenced multiple times within a query. As the <code>FieldType</code> type, the <code>DeclaredFieldType</code> type is derived by extension from the <code>xPathType</code> type which is itself derived from the <code>token</code> type. It serves to specify a data path within an item's metadata using an XPath expression. XPath expressions appearing in MPQF can be absolute or relative. When absolute, independently of the kind of evaluation item being addressed (see the Terminology section), an XPath expression will always refer to the root

<i>Name</i>	<i>Definition</i>
	of the metadata document related to the multimedia content to which the evaluation item belongs (even if the evaluation-item is related just to a fragment of this document). When relative, an XPath expression will refer to the root node of the metadata fragment (which can be the root of the metadata document or cannot) related to the evaluation item being addressed. If a <code>typeName</code> is specified, the XPath expression will refer to all the elements belonging to the named type appearing anywhere within the evaluation item being addressed. In this case only relative XPath expressions are allowed.
<code>Id</code>	Specifies a unique identifier for the declared field. The attribute <code>id</code> can be used for referencing the field multiple times within the query.
<code>typeName</code>	Specifies the name of a complex data type defined in MPEG-7 or any other metadata schema, which will serve as the starting point of a relative XPath expression.

Semantics of the `ResourceType` type:

<i>Name</i>	<i>Definition</i>
<code>ResourceType</code>	Specifies the root type of all resource types. It is abstract.
<code>resourceID</code>	Specifies a unique identifier for the declared resource. The identifier can be used for referencing the resource multiple times within the query.

Semantics of the `MediaResourceType` type:

<i>Name</i>	<i>Definition</i>
<code>MediaResourceType</code>	Subtype of the <code>ResourceType</code> type which allows describing one or more resources by containing or pointing to their raw media data.
<code>MediaResource</code>	Element of the <code>MediaLocatorType</code> type which allows describing a resource by containing or pointing to its raw media data.

Semantics of the `DescriptionResourceType` type:

Name	Definition
<code>DescriptionResourceType</code>	Specifies the <code>DescriptionResourceType</code> type which is an extension of the <code>ResourceType</code> type.
<code>DescriptionResource</code>	Specifies the container for any description based on a specific schema specified by the namespace declaration within the description.

8.4 Example

In this example, the `QFDeclaration` element allows to describe an image resource for retrieval. The following table shows a `QFDeclaration` example of an image “image.jpg” with the identifier `ImageResourceID`. By referring to `ImageResourceID`, a query condition that searches image files similar to the resource can be described in the `QueryCondition` element.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QFDeclaration>
        <Resource resourceID="ImageResourceID" xsi:type="MediaResourceType">
          <MediaResource>
            <MediaUri>image.jpg</MediaUri>
          </MediaResource>
        </Resource>
      </QFDeclaration>
    </Input>
  </Query>
</MpegQuery>
```

The next example shows that the `QFDeclaration` element allows describing Dublin Core metadata as a resource. Any XML-schema valid metadata can be target with putting its namespace in the `AnyDescription` element.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QFDeclaration>
        <Resource resourceID="DC_title" xsi:type="DescriptionResourceType">
          <AnyDescription xmlns:dc="http://purl.org/dc/elements/1.1/"
            xsi:schemaLocation="http://purl.org/dc/elements/1.1/ dc.xsd">
            <dc:title>World Cup 2006</dc:title>
          </AnyDescription>
        </Resource>
      </QFDeclaration>
    </Input>
  </Query>
</MpegQuery>
```

9 Output Description

9.1 Introduction

The `OutputDescription` element is part of the `Input` element in the `Query` element which describes the structure of the returned data from the responder to the requester. It specifies which data shall be included in the response which is defined by the `Output` element of the `Query` element and how it is structured. Furthermore, it allows limiting the maximum number of items per output page and the overall item count. Besides, it also allows the users to use the aggregation and sorting processes in the `OutputDescription` element.

9.2 Syntax

```
<complexType name="OutputDescriptionType">
  <sequence>
    <element name="ReqField" type="mpqf:FieldType" minOccurs="0"
maxOccurs="unbounded"/>
    <element name="ReqAggregateID" type="IDREF" minOccurs="0"
maxOccurs="unbounded"/>
    <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"/>
  <!-- This indicates that if I want to receive a link to the actual media
resource. -->
</complexType>

<complexType name="FieldType">
  <simpleContent>
    <extension base="mpqf:xPathType">
      <attribute name="typeName" type="string" use="optional"/>
      <attribute name="fragmentResultName" type="string" use="optional"/>
      <attribute name="fromREF" type="IDREF" use="optional"/>
      <attribute name="fieldREF" type="IDREF" use="optional"/>
      <attribute name="resultMode" use="optional" default="structured">
        <simpleType>
          <restriction base="string">
            <enumeration value="flat"/>
            <enumeration value="structured"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </simpleContent>
</complexType>
```

```

        </restriction>
    </simpleType>
</attribute>
</extension>
</simpleContent>
</complexType>

```

9.3 Semantics

Semantics of the `OutputDescriptionType` type:

Name	Definition
<code>OutputDescriptionType</code>	Describes the way how the information related to the selected digital items should be returned in the output pages.
<code>ReqField</code>	Describes a data path within the item's metadata, which a requester asks to be returned. Paths are specified by making use of relative XPath expressions, which refer to the root of the evaluation item's metadata (the one specified by the <code>EvaluationPath</code> element), or optionally using absolute XPath expressions referred to the root of the multimedia content's metadata (to which the evaluation item belongs). Depending on the value of the <code>resultMode</code> attribute, the resulting metadata fragments of the different <code>ReqField</code> elements will appear in several <code>ResultField</code> elements or within a single <code>Description</code> element (or two if a <code>Join</code> operation is used). If the <code>GroupBy</code> element appears in the query, only aggregation results and fields appearing within a <code>GroupByField</code> element can be requested.
<code>ReqAggregateID</code>	Describes the ID of the aggregate operation the requester asks to be returned. When one or more <code>ReqAggregateIDs</code> are used, the aggregate ID should be in the <code>GroupBy</code> element.
<code>ReqSemanticField</code>	Describes the place holder of the desired semantic variable, which is filtered in the Semantic Expression part of the <code>QueryCondition</code> . The place holder follows the SPARQL syntax beginning with a question mark and a descriptive identifier: e.g. <code>?person</code>
<code>GroupBy</code>	Describes the grouping operation the user wants to apply to the query results.
<code>GroupByField</code>	Describes the key for the grouping process. In a usual case, leaf nodes are intended to be used as an instance of the <code>GroupByField</code> . When a structured field (e.g. DS) is defined as a key, all the mandatory elements and attributes included in the pointed node shall be considered for grouping. For an optional field to be considered in the grouping, it should be specified individually. All used keys in the <code>GroupByField</code> element are required to appear as a <code>ReqField</code> element in the <code>OutputDescription</code> element.
<code>Aggregate</code>	Describes the aggregate operation the user wants to apply to the results of the specified grouping operation.
<code>SortBy</code>	Describes the sort operation the user wants to apply to the query results.

<i>Name</i>	<i>Definition</i>
<code>maxPageEntries</code>	Describes the maximum number of items in one output page. (optional) The result items of a given query can be split among different output pages.
<code>maxItemCount</code>	Describes the maximum number of result items. (optional)
<code>freeTextUse</code>	Specifies the desire that the <code>TextResult</code> field appears in each result item. (optional) If the value of this attribute is <code>false</code> , the <code>TextResult</code> field shall not appear in the result items. If the value of this attribute is not specified, it is assumed that the requester does not care whether the values appear or not.
<code>thumbnailUse</code>	Specifies the desire that the <code>Thumbnail</code> field appears in each result item. (optional) If the value of this attribute is <code>false</code> , the <code>Thumbnail</code> field is not allowed to appear in the result items. If the value of this attribute is not specified, it is assumed that the requester does not care whether the values appear or not.
<code>mediaResourceUse</code>	Specifies the desire that the <code>MediaResource</code> field appears in each result item. (optional) If the value of this attribute is <code>false</code> , the <code>MediaResource</code> field is not allowed to appear in the result items. If the value of this attribute is not specified, it is assumed that the requester does not care whether the values appear or not.
<code>outputNameSpace</code>	Specifies the URI of the namespace associated to the desired metadata format to appear within the <code>Description</code> element of each result item. It may be a namespace related to MPEG-7 or to any other metadata specifications.
<code>Distinct</code>	Specifies the desire that records containing exact the same metadata are required to be merged or not.

Semantics of the `FieldType` type:

<i>Name</i>	<i>Definition</i>
<code>FieldType</code>	Specifies a data path within an item's metadata. This type is derived by extension from <code>xPathType</code> type. Paths are specified by making use of absolute XPath expressions, which refer to the root of the item's metadata, or optionally relative XPath expressions, which refer to a complex type of a schema given by the <code>outputNameSpace</code> attribute.
<code>fragmentResultName</code>	If the <code>resultMode</code> attribute is set to "flat", this attribute specifies the value which will appear within the <code>name</code> attribute of the corresponding <code>FragmentResult</code> element in each result item.
<code>typeName</code>	Specifies the name of a complex data type defined in MPEG-7 or any other metadata schema given by the <code>outputNameSpace</code> attribute, which will serve as the starting point of a relative XPath expression. (optional).

Name	Definition
fromREF	Describes a reference to the Join object ID for use in the OutputDescription element for Join operation.
fieldREF	Describes a reference to the ID of a Field of other operations for reuse.
resultMode	If set to flat, specifies the desire that the selected metadata fragment appears in a FragmentResult element in each result item. If the value of this attribute is not specified, or is set to structured, the Description element will be used instead (carrying all the metadata fragments selected by all the ReqField elements).

9.4 Example

The following example illustrates the use of the OutputDescription element for describing the desired output of an MPQF query. In this example, a simple free text query is specified which searches for textual descriptions containing "San Jose". In addition, the target domain is limited to images of the JPEG format. The OutputDescription element is used to select five fields from the metadata of the resulting digital items. Additionally, the OutputDescription element specifies that three result pages or less will be computed, that each one of these pages shall have at most ten items and that the freeTextUse field is desired.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <OutputDescription maxItemCount="30" maxPageEntries="10" freeTextUse="true"
outputNameSpace="urn:mpeg:mpeg7:schema:2004" >
        <ReqField typeName="MediaLocatorType"/>MediaUri</ReqField>
        <ReqField typeName="CreationInformationType"/>Creation/Title</ReqField>
        <ReqField typeName="CreationInformationType"/>Creation/Creator</ReqField>
        <ReqField typeName="MediaFormatType"/>FileFormat</ReqField>
        <ReqField typeName="MediaFormatType"/>FileSize</ReqField>
      </OutputDescription>
      <QueryCondition>
        <TargetMediaType xsi:type="mimeType">image/jpeg</TargetMediaType>
        <Condition xsi:type="QueryByFreeText">
          <FreeText>San Jose</FreeText>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

This second example illustrates the use of resultMode attribute set to "flat" to obtain a flat metadata output. In this example, a simple free text query is specified which searches for textual descriptions containing "San Jose". In addition, the target domain is limited to images of the JPEG format. The OutputDescription element is used to select two fields from the metadata of the resulting digital items (width and height of the image), with the resultMode attribute set to "flat".

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <OutputDescription maxItemCount="30" maxPageEntries="10" freeTextUse="true"
outputNameSpace="urn:mpeg:mpeg7:schema:2004" >
        <ReqField typeName="MediaLocatorType"/>MediaUri</ReqField>
        <ReqField typeName="MediaFormatType"/>FileFormat</ReqField>
      </OutputDescription>
      <QueryCondition>
        <TargetMediaType xsi:type="mimeType">image/jpeg</TargetMediaType>
        <Condition xsi:type="QueryByFreeText">
          <FreeText>San Jose</FreeText>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

```

    <ReqField typeName="width"
resultMode="flat">MediaInformation/MediaProfile/MediaFormat/VisualCoding/Frame/@w
idth</ReqField>
    <ReqField typeName="height"
resultMode="flat">MediaInformation/MediaProfile/MediaFormat/VisualCoding/Frame/@h
eight</ReqField>
  </OutputDescription>
  <QueryCondition>
    <TargetMediaType xsi:type="mimeType">image/jpeg</TargetMediaType>
    <Condition xsi:type="QueryByFreeText">
      <FreeText>San Jose</FreeText>
    </Condition>
  </QueryCondition>
</Input>
</Query>
</MpegQuery>

```

The following is the example of an expected output by the specified OutputDescription above.

```

<MpegQuery>
  <Query>
    <Output currPage="1" totalPages="1" expirationDate="2008-05-30T09:00:00">
      <ResultItem xsi:type="ResultItemType" recordNumber="1">
        <TextResult>Title 01</TextResult>
        <FragmentResult name="width">640</FragmentResult>
        <FragmentResult name="height">480</FragmentResult>
      </ResultItem>
      <ResultItem recordNumber="2">
        <TextResult>Title 02</TextResult>
        <FragmentResult name="width">320</FragmentResult>
        <FragmentResult name="height">200</FragmentResult>
      </ResultItem>
      <ResultItem recordNumber="3">
        <TextResult>Title 03</TextResult>
        <FragmentResult name="width">800</FragmentResult>
        <FragmentResult name="height">1000</FragmentResult>
      </ResultItem>
    </Output>
  </Query>
</MpegQuery>

```

The following is the example of an expected output by the specified OutputDescription above.

```

<MpegQuery>
  <Query>
    <Output currPage="1" totalPages="1" expirationDate="2008-05-30T09:00:00">
      <ResultItem xsi:type="ResultItemType" recordNumber="1">
        <TextResult>Title 01</TextResult>
        <Description xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
          <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="mpeg7:CreationInformationType">
              <mpeg7:Creation>
                <mpeg7:Title>Title 01</mpeg7:Title>
              </mpeg7:Creation>
            </mpeg7:DescriptionUnit>
          </mpeg7:Mpeg7>
          <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="mpeg7:CreationType">

```

```

    <mpeg7:Title>Title 01</mpeg7:Title>
    <mpeg7:Creator xsi:type="mpeg7:CreatorType">
      <mpeg7:Role href="http://www.roles.com/author"/>
      <mpeg7:Agent xsi:type="mpeg7:PersonType">
        <mpeg7:Name>
          <mpeg7:GivenName>John</mpeg7:GivenName>
          <mpeg7:FamilyName>Smith</mpeg7:FamilyName>
        </mpeg7:Name>
      </mpeg7:Agent>
    </mpeg7:Creator>
  </mpeg7:DescriptionUnit>
</mpeg7:Mpeg7>
</Description>
<AggregationResult aggregateID="avgSize">256</AggregationResult>
</ResultItem>
<ResultItem recordNumber="2">
  <TextResult>Title 02</TextResult>
  <AggregationResult aggregateID="avgSize">512</AggregationResult>
</ResultItem>
<ResultItem recordNumber="3">
  <TextResult>Title 03</TextResult>
  <AggregationResult aggregateID="avgSize">1024</AggregationResult>
</ResultItem>
</Output>
</Query>
</MpegQuery>

```

9.5 SortBy

9.5.1 Introduction

This sub clause specifies all the types of **SortBy** tools which are used in the **OutputDescription** element. The **AbstractSortByType** type is defined as an abstract data type from which the **SortByFieldType** type and **SortByAggregateType** type are derived.

9.5.2 Syntax

```

<complexType name="AbstractSortByType" abstract="true">
  <attribute name="order" use="required">
    <simpleType>
      <restriction base="string">
        <enumeration value="ascending"/>
        <enumeration value="descending"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>

<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>

```

```

</complexType>

<complexType name="SortByAggregateType">
  <complexContent>
    <extension base="mpqf:AbstractSortByType">
      <choice>
        <element name="Aggregate" type="mpqf:AggregateExpressionType"/>
        <element name="AggregateID" type="IDREF"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

```

9.5.3 Semantics

Semantics of the `SortBy` types:

Name	Definition
<code>AbstractSortByType</code>	Defines the abstract data type for the two <code>SortBy</code> sub-types.
<code>order</code>	Indicates the order of the sorting. The current version supports either ascending or descending sort order.
<code>SortByFieldType</code>	Describes a sort type in which the <code>Field</code> element is used as a sorting key.
<code>SemanticField</code>	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. <code>?person</code>
<code>SortByAggregateType</code>	Describes a sort type in which the result of an aggregate expression is used as a key.
<code>Aggregate</code>	Describes an aggregate expression, whose result is used as a key in sorting process.
<code>AggregateID</code>	Describes an aggregate expression by id reference, whose result is used as a key in the sorting process.

9.5.4 Example

The following example illustrates the use of the `OutputDescription` element for describing the desired output of an MPQF query in combination with a `GroupBy` and `SortBy` statement. In addition, the use of an aggregate function is demonstrated. Both statements (`ReqAggregateID` and `SortBy`) reference to the introduced `Aggregate` element in the `GroupBy`.

```

<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <OutputDescription outputNameSpace="urn:mpeg:mpeg7:2004">
        <ReqField typeName="CreationInformation"/>/Creation/Title</ReqField>
        <ReqField typeName="Creator"/>/Character/FamilyName</ReqField>
        <ReqAggregateID>avgSize</ReqAggregateID>
        <GroupBy>
          <GroupByField typeName="Creator"/>/Character/FamilyName</GroupByField>
          <Aggregate xsi:type="AVG" aggregateID="avgSize">
            <Field typeName="MediaFormat"/>/FileSize</Field>
          </Aggregate>
        </GroupBy>
        <SortBy xsi:type="SortByAggregateType" order="ascending">
          <AggregateID>avgSize</AggregateID>
        </SortBy>
      </OutputDescription>
    </Input>
  </Query>
</MpegQuery>

```

The following is an example of expected output by the specified `OutputDescription` element above. Note that the actual output depends on the multimedia content stored in the repository.

```

<MpegQuery mpqfID="someID">
  <Query>
    <Output currPage="1" totalPages="1" expirationDate="2008-05-30T09:00:00">
      <ResultItem xsi:type="ResultItemType" recordNumber="1">
        <TextResult>Title 01</TextResult>
        <Description xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004">
          <mpeg7:Mpeg7>
            <mpeg7:DescriptionUnit xsi:type="mpeg7:CreationInformationType">
              <mpeg7:Creation>
                <mpeg7:Title>Title 01</mpeg7:Title>
              </mpeg7:Creation>
            </mpeg7:DescriptionUnit>
          </mpeg7:Mpeg7>
        </Description>
        <AggregationResult aggregateID="avgSize">256</AggregationResult>
      </ResultItem>
      <ResultItem recordNumber="2">
        <TextResult>Title 02</TextResult>
        <AggregationResult aggregateID="avgSize">512</AggregationResult>
      </ResultItem>
      <ResultItem recordNumber="3">
        <TextResult>Title 03</TextResult>
        <AggregationResult aggregateID="avgSize">1024</AggregationResult>
      </ResultItem>
    </Output>
  </Query>
</MpegQuery>

```

10 Query Condition

10.1 Introduction

The `QueryCondition` element, of the `QueryConditionType` type, is the part of the Input Query Format where the user specifies the properties of the media or the metadata to be retrieved. The `QueryCondition` element is defined as a sequence of desired target media types, an `EvaluationPath` element and a condition tree, which can be a simple (`Condition` element) or a Join operation (`Join` element). The sequence of the target media types declares the desired MIME types which the user expects as a result. The `EvaluationPath` element (optional) declares an XPath expression, which specifies the node of the metadata fragment related to the evaluation item being addressed (`//VideoSegment` for instance). It also determines the structure of the output; one result item will be returned for each evaluation item if it matches the condition. If the `EvaluationPath` element is not specified, the output result shall be provided as a collection of multimedia contents, as stored in the repository each of which satisfies the query condition. If the `QueryCondition` element does not appear within the Input element, or if the `QueryCondition` element does not contain neither a `Condition` nor a `Join` element, it is considered to be an Empty Query asking to retrieve all records in the database. The condition tree can be specified using the `Condition` element, of the `BooleanExpressionType` type, which is the parent abstract type of all the types resulting in a value in the range of [0..1]. A more complex condition tree can be specified using the `Join` element. The Join operation allows the definition of filtering conditions which act over two sets of multimedia objects.

10.2 Syntax

```
<complexType name="QueryConditionType">
  <sequence>
    <element name="EvaluationPath" type="mpqf:xPathType" minOccurs="0"/>
    <element name="TargetMediaType" type="mpqf:mimeType" minOccurs="0"
maxOccurs="unbounded"/>
    <choice minOccurs="0">
      <element name="Join" type="mpqf:JoinType"/>
      <element name="Condition" type="mpqf:BooleanExpressionType"/>
    </choice>
  </sequence>
</complexType>
```

10.3 Semantics

Semantics of the `QueryConditionType` type:

Name	Definition
<code>QueryConditionType</code>	Defines a sequence of desired target media types, an <code>EvaluationPath</code> element, and a condition tree, which can be simple (<code>Condition</code> element) or a Join operation (<code>Join</code> element).
<code>EvaluationPath</code>	Optional XPath expression, which specifies the node of the metadata fragment related to the evaluation item being addressed. It also determines the structure of the output; one result item will be returned for each evaluation item if it matches the condition. If the <code>EvaluationPath</code> element is not specified, the output result shall be provided as a collection of multimedia contents, as stored in the repository each of which satisfies the query condition.

Name	Definition
TargetMediaType	The sequence of the target media types declares the desired MIME types which the user expects as a result. This element is required not to be used in case a Join element is stated.
Condition	The condition tree of the query is specified using a BooleanExpressionType type, which is the parent abstract type of all the types resulting in a value in the range of [0..1].
Join	Optionally, a Join element can be used instead of the Condition element. A Join operation allows the definition of filtering conditions which act over two sets of multimedia objects. With the Join operation, it is possible to specify conditions which define two separate filtered sets and combine them (see JoinType type).

10.4 Example

The following example illustrates the usage of the QueryConditionType type. It demonstrates a combined (by AND) retrieval for all items containing the word “Lausanne” somewhere in a textual description and where the annotated Title is equal to the string “Blade Runner”. In addition, the target domain is limited to images of the JPEG compression format (see TargetMediaType element). Note, that the target domain also can be restricted to all images by using *image/**.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <TargetMediaType>image/jpeg</TargetMediaType>
        <Condition xsi:type="AND">
          <Condition xsi:type="QueryByFreeText">
            <FreeText>Lausanne</FreeText>
          </Condition>
          <Condition xsi:type="Equal">
            <StringField typeName="CreationType">Title</StringField>
            <StringValue>Blade Runner</StringValue>
          </Condition>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

The following example illustrates the use of the EvaluationPath element. It shows a query to search for all video segments containing the word “Lausanne” somewhere in a textual description. Note, that for each matching segment, one different result item will be returned.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <EvaluationPath>//VideoSegment</EvaluationPath>
        <Condition xsi:type="QueryByFreeText">
          <FreeText>Lausanne</FreeText>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```

```

        </Condition>
    </QueryCondition>
</Input>
</Query>
</MpegQuery>

```

11 Expression Types

11.1 Introduction

Expression Types allow to build expression trees. There are three main abstract Expression Types: the `BooleanExpressionType` type, which is the base type of all the expressions resulting in a value in the range of [0..1], the `ArithmeticExpressionType` type which is the base type of all the expressions resulting in a numeric value (Add, Subtract, Multiply, Divide, Modulus, Abs, Ceiling, Floor and Round), and the `StringExpressionType` type, which is the base type of all the expressions resulting in a string value (UpperCase and LowerCase).

11.2 Syntax

```

<complexType name="BooleanExpressionType" abstract="true">
  <attribute name="preferenceValue" type="mpqf:zeroToOneType"
    use="optional" default="1"/>
  <attribute name="thresholdValue" type="mpqf:zeroToOneType"
    use="optional" />
</complexType>
<complexType name="ArithmeticExpressionType" abstract="true"/>
<complexType name="StringExpressionType" abstract="true"/>

```

11.3 Semantics

Semantics of the abstract Expression types:

Name	Definition
<code>BooleanExpressionType</code>	An abstract type that is the base type for defining the boolean expressions.
<code>preferenceValue</code>	Used to indicate the importance, priority or weight assigned to a particular user preference component, relative to other components. The range of the preference values is from 0 to 1. Positive float value indicates the degree of desired preference. A default, positive, value of 1 corresponds to a nominal preference. The sum of all assigned preference values at one level is required to be 1.
<code>thresholdValue</code>	Used to indicate the minimum value the score of an evaluation-item based on a specific operation (query type, expression) is required to have. Otherwise the evaluation-item is not considered further during evaluation.
<code>ArithmeticExpressionType</code>	An abstract type that is the base type for defining the arithmetic expressions.
<code>StringExpressionType</code>	An abstract type that is the base type for defining the string expressions.

11.4 Operands

11.4.1 DateTimeOperands

11.4.1.1 Introduction

The `DateTimeOperands` group allows specifying one element which has a date or time value as content. Each element can instantiate one of the four types which result in a date or a time value (the `date` type, the `time` type, the `dateTime` type and the `FieldType` type).

11.4.1.2 Syntax

```
<group name="DateTimeOperands">
  <sequence>
    <choice>
      <element name="DateTimeValue" type="dateTime"/>
      <element name="DateValue" type="date"/>
      <element name="TimeValue" type="time"/>
      <element name="DateTimeField" type="mpqf:FieldType"/>
    </choice>
  </sequence>
</group>
```

11.4.1.3 Semantics

Semantics of the `DateTimeOperands` group:

Name	Definition
<code>DateTimeOperands</code>	Defines the <code>DateTimeOperands</code> group, which allows specifying one element that can instantiate one of the four types which result in a date and/or a time value.
<code>DateTimeValue</code>	Allows specifying a constant date-time value by <code>dateTime</code> type.
<code>DateValue</code>	Allows specifying a constant date value by <code>date</code> type.
<code>TimeValue</code>	Allows specifying a constant time value by <code>time</code> type.
<code>DateTimeField</code>	Allows specifying a path to a date-time value within the metadata of a multimedia item by <code>FieldType</code> type.

11.4.1.4 Example

The following example illustrates how the group `DateTimeOperands` allows that the input of the `Equal` operation can be a constant date-time value and a path to a date-time value within the metadata of the evaluation item.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <Condition xsi:type="Equal">
          <DateTimeField
type="CreationType"/>CreationCoordinates/Date/TimePoint</DateTimeField>
```

```

<DateTimeValue>2002-05-30T09:00:00</DateTimeValue>
  </Condition>
</QueryCondition>
</Input>
</Query>
</MpegQuery>

```

11.4.2 DurationOperands

11.4.2.1 Introduction

The `DurationOperands` group allows specifying one element which consists of a duration value. Each element can instantiate one of the two types that result in a duration value (the `duration` type and the `FieldType` type).

11.4.2.2 Syntax

```

<group name="DurationOperands">
  <sequence>
    <choice>
      <element name="DurationValue" type="duration"/>
      <element name="DurationField" type="mpqf:FieldType"/>
    </choice>
  </sequence>
</group>

```

11.4.2.3 Semantics

Semantics of the `DurationOperands` group:

Name	Definition
<code>DurationOperands</code>	Defines the <code>DurationOperands</code> group, which allows specifying one element from one of the two types that result in a duration value.
<code>DurationValue</code>	Allows specifying a constant duration value of <code>duration</code> type.
<code>DurationField</code>	Allows specifying a path to a duration value within the metadata of an evaluation item by <code>FieldType</code> type.

11.4.2.4 Example

The following example illustrates how the group `DurationOperands` allows that the input of the `Equal` operation can be a constant duration value and a path to a duration value within the metadata of the object for example.

```

<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <Condition xsi:type="Equal">
          <DurationField typeName="SomeVideoDescriptionType">Duration</DurationField>

```

```

<DurationValue>P5Y2M10DT15H</DurationValue>
  </Condition>
</QueryCondition>
</Input>
</Query>
</MpegQuery>

```

11.5 Boolean Expression Types

11.5.1 Introduction

The `BooleanExpressionType` type is an abstract type which is the base type of the Boolean operators (AND, OR, NOT, XOR), the `ComparisonExpressionType` type and the `QueryType` type. These types allow to build expressions evaluating to a value in the range of [0..1]. The `BooleanExpressionType` type provides one attribute that indicates the degree of preference (relative weight of importance) of the query condition. Preference values (if present) are always taken into account within the same hierarchy level of the XML tree (i.e. between siblings of the same sequence). The AND and the OR types should have two or more condition elements. The XOR type should have exactly two condition elements, and the NOT type should have exactly one condition element.

11.5.2 Syntax

```

<complexType name="AND">
  <complexContent>
    <extension base="mpqf:BooleanExpressionType">
      <sequence>
        <element name="Condition" type="mpqf:BooleanExpressionType"
          minOccurs="2" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="scoringFunction" type="mpqf:SimpleTermType"
        use="optional"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="OR">
  <complexContent>
    <extension base="mpqf:BooleanExpressionType">
      <sequence>
        <element name="Condition" type="mpqf:BooleanExpressionType"
          minOccurs="2" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="scoringFunction" type="mpqf:SimpleTermType"
        use="optional"/>
    </extension>
  </complexContent>
</complexType>
<complexType name="NOT">
  <complexContent>
    <extension base="mpqf:BooleanExpressionType">
      <sequence>
        <element name="Condition" type="mpqf:BooleanExpressionType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="XOR">
  <complexContent>

```

```

<extension base="mpqf:BooleanExpressionType">
  <sequence>
    <element name="Condition" type="mpqf:BooleanExpressionType"
      minOccurs="2" maxOccurs="2"/>
  </sequence>
  <attribute name="scoringFunction" type="mpqf:SimpleTermType"
    use="optional"/>
</extension>
</complexContent>
</complexType>

```

11.5.3 Semantics

Semantics of the Boolean Expression Types:

Name	Definition
<code>BooleanExpressionType</code>	An abstract type that is the base type for defining the boolean expressions.
<code>scoringFunction</code>	An attribute referring to a classification scheme, which contains different scoring functions. If combining two different <code>QueryTypes</code> , such as <code>QueryByMedia</code> and <code>QueryByDescription</code> with an AND, a fuzzy logic may be applied. That means that values of type float are aggregated based on the desired scoring function (e.g., Min, Sum, Product, etc.). The terms of supported scoring functions are required to be referred at a classification scheme. In case a responder may not support this attribute, it should be ignored.
AND	A complex type inherited from the <code>BooleanExpressionType</code> type, for using the Boolean “AND” operator in a query and it shall result to a value in the range of [0..1]. If the resulting values (scores) are not exactly 0 or 1, the computation of the result for the AND operation is up to the implementers, who may apply rules as provided in subclause 11.5.4.
OR	A complex type inherited from the <code>BooleanExpressionType</code> type, for using the Boolean “OR” operator in query and it shall result to a value in the range of [0..1]. If the resulting values (scores) are not exactly 0 or 1, the computation of the result for the OR operation is up to the implementers, who may apply rules as provided in subclause 11.5.4.
NOT	A complex type inherited from the <code>BooleanExpressionType</code> type, for using the “NOT” operator in a query and it shall result to a value in the range of [0..1]. If the resulting values (scores) are not exactly 0 or 1, the computation of the result for the NOT operation is up to the implementers, who may apply rules as provided in subclause 11.5.4.
XOR	A complex type inherited from the <code>BooleanExpressionType</code> type, for using the “XOR” operator in query and it shall result to a value in the range of [0..1]. If the resulting values (scores) are not exactly 0 or 1, the computation of the result for the XOR operation is up to the implementers, who may apply rules as provided in subclause 11.5.4.

11.5.4 Definitions

This subclause is an informative section.

It is recommended that the scoring function of the **AND** operator may conform to the rules applied in Definition 1 (t-norm) and to the weighted t-norm in case of a given preference value. The final score is calculated by the (weighted) input values of the respective conditions.

It is recommended that the scoring function of the **OR** operator may conform to the rules applied in Definition 2 (t-conorm) and to the weighted t-conorm in case of a given preference value. The final score is calculated by the (weighted) input values of the respective conditions.

It is recommended that the evaluation of the **NOT** operator may conform to the rules applied in Definition 3. The final score is calculated by the (weighted) input values of the respective conditions.

It is recommended that the scoring function of the **XOR** operator may conform to the rules applied in Definition 4 and to the same weighted rules as for t-conorm in case of a given preference value. The final score is calculated by the (weighted) input values of the respective conditions

In order to evaluate individual scores at the **AND**, **OR** or **XOR** element, this part of ISO/IEC 15938 recommends to use the scoring function $T: [0,1] \times [0,1] \rightarrow [0,1]$ which relates to the t-norms (for **AND**) and t-conorms (for **OR**).

Definition 1: Function $T(\text{AND}): [0,1] \times [0,1] \rightarrow [0,1]$ is a t-norm if it satisfies the following criteria:

1. $T(s_1, s_2) = T(s_2, s_1)$ (commutativity)
2. $T(s_1, T(s_2, s_3)) = T(T(s_1, s_2), s_3)$ (associativity)
3. $s_1 \leq s_2 \wedge s_3 \leq s_4 \rightarrow T(s_1, s_2) \leq T(s_3, s_4)$ (monotonicity)
4. $T(s_1, 1) = s_1$ (border condition)

Definition 2: Function $T(\text{OR}): [0,1] \times [0,1] \rightarrow [0,1]$ is a t-conorm if it satisfies the following criteria:

1. criteria 1-3 of the t-norm
2. $T(s_1, 0) = s_1$ (border condition)

The score evaluation for the **NOT** operation is defined as follows:

Definition 3: Function $T(\text{NOT}): [0,1] \rightarrow [0,1]$ requires the following conditions:

1. $T(\text{NOT})(T(\text{NOT})(s_1)) = s_1$
2. DeMorgan laws are applied:
 - a. $T(\text{NOT})(T(\text{AND})(s_1, s_2)) = T(\text{OR})(T(\text{NOT})(s_1), T(\text{NOT})(s_2))$
 - b. $T(\text{NOT})(T(\text{OR})(s_1, s_2)) = T(\text{AND})(T(\text{NOT})(s_1), T(\text{NOT})(s_2))$

The score evaluation for the **XOR** operation is defined as follows:

Definition 4: Function $T(\text{XOR}): [0,1]^n \rightarrow [0,1]$ is realized by combining the operation **AND**, **OR** and **NOT** as follows:

1. $T(\text{XOR})(s_1, s_2, \dots, s_n) = T(\text{OR})(T(\text{AND}_i))$ with $i=1..n$
2. $T(\text{AND}_i) = T(\text{AND})(T(\text{NOT}) s_1, \dots, T(\text{NOT}) s_{(i-1)}, s_i, T(\text{NOT}) s_{(i+1)}, \dots, T(\text{NOT}) s_n)$

In order to integrate preference values ($p \in [0,1]$), the weighted t-norm and t-conorm are defined:

$T(AND_p): ([0,1] \times [0,1])^n \rightarrow [0,1]$, where the first parameter identifies the preference. In addition, the following rules apply:

1. $T(AND_p)(p_1, s_1, p_2, s_2, \dots, p_n, s_n) = T(AND_p)(p_2, s_2, \dots, p_n, s_n)$
2. If for all $p_1, \dots, p_n$ in the filter condition tree (at one level) it is true that: $p_1 = \dots = p_n$ (all preference values have the same value) then:

$$T(AND_p)(p_1, s_1, p_2, s_2, \dots, p_n, s_n) = T(AND)(s_1, s_2, \dots, s_n)$$

3. The weighted scoring function T within the filter condition tree must be continuous concerning the scores (s_x) as well as its weights (p_x).

Note, the same rules apply to $T(OR_p)$. The function $T(NOT_p)$ is reduced to $T(NOT)$. The function $T(XOR_p)$ is substituted by $T(AND_p)$, $T(OR_p)$ and $T(NOT_p)$ as defined in definition 4.

11.5.5 Example

The following example illustrates compounding some Boolean operations. The top level “AND” operator, includes four nested operations (“AND”, “OR”, “XOR”, and “NOT”). The second level “AND” operator, with preference value “0.1”, includes two kinds of query conditions, and the “OR” and the “XOR” operators, with preference value “0.4”, include two query conditions. The “NOT” operator, with preference value “0.1”, includes only one query condition.

```
<MpegQuery mpqfID="someID">
  <Query>
    <Input>
      <QueryCondition>
        <EvaluationPath>//VideoSegment</EvaluationPath>
        <Condition xsi:type="AND">
          <Condition xsi:type="AND" preferenceValue="0.1">
            <Condition xsi:type="QueryByFreeText" preferenceValue="0.7">
              <FreeText>Lausanne</FreeText>
            </Condition>
            <Condition xsi:type="QueryByXQuery" preferenceValue="0.3">
              <XQuery><![CDATA[
                $a := node()//MediaFormat/Content/Name
                $a/text()="Video" ]]></XQuery>
            </Condition>
          </Condition>
          <Condition xsi:type="OR" preferenceValue="0.4">
            <Condition xsi:type="Equal" >
              <DateTimeField
                typeName="CreationType">/CreationCoordinates/Date/TimePoint</DateTimeField>
              <DateTimeValue>2002-05-30T09:00:00</DateTimeValue>
            </Condition>
            <Condition xsi:type="QueryByFreeText" >
              <FreeText>San Jose</FreeText>
            </Condition>
          </Condition>
          <Condition xsi:type="XOR" preferenceValue="0.4">
            <Condition xsi:type="Equal">
              <BooleanField
                typeName="MediaFormatType">bitrate/@variable</BooleanField>
              <BooleanValue>true</BooleanValue>
            </Condition>
            <Condition xsi:type="QueryByFreeText">
              <FreeText>Marrakech</FreeText>
            </Condition>
          </Condition>
        </Condition>
      </QueryCondition>
    </Input>
  </Query>
</MpegQuery>
```