INTERNATIONAL STANDARD

ISO/IEC 21000-17

First edition
2006-09-15

Information technology — Multimedia framework (MPEG-21) —

Part 17:
Fragment Identification of MPEG Resources

Technologies de l'information — Cadre multimédia (MPEG-21) —

Partie 17: Identification de fragments de ressources MPEG

Reference number
ISO/IEC 21000-17:2006(E)

Contents

Page

Foreword ... iv
Introduction ... vi
1 Scope ... 1
1.1 General ... 1
1.2 Organization of this document ... 1
1.3 Relationship with the MPEG-21 Framework ... 2
2 Normative references ... 2
3 Terms, definitions and abbreviated terms ... 3
3.1 Definitions ... 3
3.2 Abbreviations ... 6
3.3 Namespaces ... 6
4 MPEG URI Fragment Identifier Framework ... 7
4.1 Introduction ... 7
4.2 W3C XPointer Framework ... 7
4.3 Use of namespaces and of the xmlns() pointer scheme ... 7
4.4 Hierarchical combination of pointer parts ... 8
4.5 Syntax ... 8
4.6 Character escaping ... 9
4.7 Processing ... 9
5 Pointer schemes ... 10
5.1 Introduction ... 10
5.2 The `ffp()` pointer scheme ... 10
5.3 The `offset()` pointer scheme ... 12
5.4 The `mp()` pointer scheme ... 12
5.5 The `mask()` pointer scheme ... 25
6 Logical Model representation ... 26
6.1 Introduction ... 26
6.2 Schema for Logical Model representation ... 26
6.3 Semantics ... 27
6.4 Schema for schemas ... 28
6.5 Logical Model for MPEG-4 files ... 28
Annex A (informative) Examples of spatiotemporal fragments ... 31
Annex B (informative) Logical model of a resource - examples ... 34
Annex C (informative) Time scheme syntaxes ... 38

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 21000-17 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 21000 consists of the following parts, under the general title *Information technology — Multimedia framework (MPEG-21)*:

— *Part 1: Vision, Technologies and Strategy* [Technical Report]

— *Part 2: Digital Item Declaration*

— *Part 3: Digital Item Identification*

— *Part 4: Intellectual Property Management and Protection Components*

— *Part 5: Rights Expression Language*

— *Part 6: Rights Data Dictionary*

— *Part 7: Digital Item Adaptation*

— *Part 8: Reference Software*

— *Part 9: File Format*

— *Part 10: Digital Item Processing*

— *Part 11: Evaluation Tools for Persistent Association Technologies* [Technical Report]

— *Part 12: Test Bed for MPEG-21 Resource Delivery* [Technical Report]

— *Part 14: Conformance Testing*

— *Part 15: Event Reporting*

— *Part 16: Binary Format*

— *Part 17: Fragment Identification of MPEG Resources*

The following parts are under preparation:

— *Part 18: Digital Item Streaming*

Introduction

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no “big picture” to describe how these elements, either in existence or under development, relate to each other. The aim for ISO/IEC 21000 (MPEG-21) is to describe how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new International Standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new International Standards as appropriate while other relevant International Standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.

The vision for MPEG-21 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.

A key concept of the multimedia framework is the Digital Item. In MPEG-21 a Digital Item is a structured digital object with a standard representation, identification, and metadata. An equally important concept in the multimedia framework is the notion of the User. In MPEG-21 a User is any entity that interacts with the multimedia framework and as such includes all members of the value chain (e.g. creator, rights holders, distributors and consumers of Digital Items) and include, for example, individuals, consumers, communities, organizations, corporations, consortia and governments.

This part of MPEG-21 specifies a normative syntax for URI Fragment Identifiers to be used for addressing parts of MPEG resources. MPEG URI Fragment Identifier schemes offer comprehensive and flexible mechanisms for addressing fragments of audiovisual content. Therefore, their use may also be extended to other audiovisual Internet Media types.

Information technology — Multimedia framework (MPEG-21) —

Part 17: Fragment Identification of MPEG Resources

1 Scope

1.1 General

This International Standard is titled “Fragment Identification of MPEG Resources” and specifies a normative syntax for URI Fragment Identifiers to be used for addressing parts of any resource whose Internet Media Type is one of:

- audio/mpeg	[RFC3003];
- video/mpeg	[RFC2045, RFC2046];
- video/mp4	[RFC4337];
- audio/mp4	[RFC4337];
- application/mp4	[RFC4337].

MPEG URI Fragment Identifier schemes offer comprehensive and flexible mechanisms for addressing fragments of audiovisual content. Therefore, their use may potentially be extended to other audiovisual Internet Media types.

Such URI Fragment Identifiers are compliant to the generic syntax for URIs defined by IETF RFC 3986 and therefore can be used after the “#” character in a URI. Where appropriate, such Fragment Identifiers can also be used in IRIs as specified by IETF RFC 3987.

The syntax for URI Fragment Identifiers defined in this specification is based on the W3C XPointer Framework Recommendation and adds the ability to address:

- temporal, spatial and spatiotemporal locations of a resource;
- logical units of a resource according to a given Logical Model;
- byte ranges of a resource;
- items or Tracks of an ISO Base Media File;
- a portion of a video through the use of a Mask.

1.2 Organization of this document

This International Standards provides firstly a set of generic principles for addressing fragments of multimedia resources. This set of principles is referred to as MPEG URI Fragment Identifier Framework (Figure 1) and is specified in Clause 4.

Secondly, this International Standards defines a set of normative pointer schemes to be used in the context of the MPEG URI Fragment Identifier Framework. These pointer schemes are specified in Clause 5.

Finally, Clause 6 provides a set of tools for representing Logical Models for various types of media, in order to allow the addressing of fragments of content.

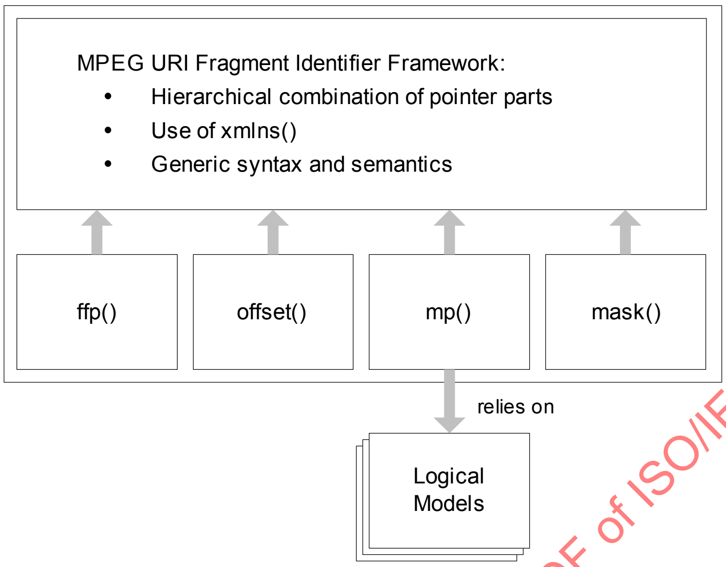
MPEG URI Fragment Identifier Framework:
Hierarchical combination of pointer parts
Use of xmlns()
Generic syntax and semantics
ffp()
offset()
mp()
mask()
relies on
Logical Models

Figure 1 — Scope of this International Standard

This International Standards also provides illustrative (non-normative) examples.

1.3 Relationship with the MPEG-21 Framework

The tools specified in this part of ISO/IEC 21000 allow identification of a part of a resource by providing a format for referencing the part using a Fragment Identifier. The Fragment Identifiers defined in this International Standards are distinct from the Digital Item Identifiers specified in ISO/IEC 21000-3 for identifying the “fundamental units of trade” within the MPEG-21 Multimedia Framework (i.e. Digital Items).

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.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 10646, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 13818-6, *Information technology — Generic coding of moving pictures and associated audio information — Part 6: Extension for DSM-CC*

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 15938-5, *Information technology — Multimedia content description interface — Part 5: Multimedia description schemes*

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

W3C Recommendation *XML Schema Part 0: Primer, Second Edition*, 28 October 2004, http://www.w3.org/TR/xmlschema-0/

W3C Recommendation *XML Schema Part 1: Structures, Second Edition*, 28 October 2004, http://www.w3.org/TR/xmlschema-1/

W3C Recommendation *XML Schema Part 2: Datatypes, Second Edition*, 28 October 2004, http://www.w3.org/TR/xmlschema-2/

W3C Recommendation *XML Path Language, Version 1.0*, 16 November 1999, http://www.w3.org/TR/xpath

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

W3C Recommendation *XPointer xmlns() Scheme*, 25 March 2003, http://www.w3.org/TR/xptr-xmlns/

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

SMPTE (Society of Motion Picture and Television Engineers) 12M-1999, *Television, Audio and Film — Time and Control Code*

IETF RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, http://www.ietf.org/rfc/rfc2045.txt

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, http://www.ietf.org/rfc/rfc2046.txt

IETF RFC 2234, *Augmented BNF for Syntax Specifications: ABNF*, November 1997, including the following core ABNF syntax rules defined by that specification: ALPHA (letters), DIGIT (decimal digits), and HEXDIG (hexadecimal digits), http://www.ietf.org/rfc/rfc2234.txt

IETF RFC 3003, *The audio/mpeg Media Type*, http://www.ietf.org/rfc/rfc3003.txt

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, http://www.ietf.org/rfc/rfc3986.txt

IETF RFC 3987, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/rfc/rfc3987.txt

IETF RFC 4337, *MIME Type Registration for MPEG-4*

3 Terms, definitions and abbreviated terms

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

3.1 Definitions

3.1.1
axis
extension of an XPath axis that specifies the tree relationship between the locations selected by a location step and the context location

NOTE In this International Standard, an axis projects the current context to one of its dimensions to allow a node-test to be performed to select a set of locations that bear a certain relation in that dimension. Each of the XPath axes (e.g. child, descendant, attribute, etc) is considered as one dimension. Dimensions can additionally be spatial, temporal and spatiotemporal.

3.1.2
context
as defined in the XPath recommendation

3.1.3
default namespace
namespace which applies to a scheme name, a logical unit or an attribute of a logical unit when no namespace prefix is specified

3.1.4
fragment identifier
component of a URI that allows identification of a portion of a resource

NOTE Such portion is referred to as fragment in this International Standard. A fragment identifier may refer to the entire resource.

3.1.5
function
in XPath, a function selects, in the current context, the subset of nodes that verify a condition

NOTE In this International Standard, a function is extended to select a subset of locations.

3.1.6
location
generalization of XPath's node that includes times, regions and temporal-regions in addition to XPath nodes

NOTE A location is a Node plus a (possibly empty) set of temporal and/or spatial conditions on the node.

3.1.7
location path
selects a set of locations relative to the context location

NOTE The result of evaluating an expression that is a location path is a location-set containing the locations selected by the location path. Location paths can recursively contain expressions that are used to filter sets of locations.

3.1.8
location-set
unordered list of locations which corresponds to a node-set produced by an XPath expression, except for the generalization to include times, regions and temporal-regions as valid members of the set

3.1.9
location step
in XPath, a location step which selects a set of nodes relative to a context

NOTE In this specification, location steps have been extended to select sets of locations.

3.1.10
logical model
provides an abstraction of the resource that is meaningful to the User of the resource

NOTE Logical models of audiovisual resources consist of a hierarchy of logical units and do not necessarily depend on how the audiovisual resource is physically stored.

3.1.11
logical model description
XML representation of the logical model of a particular resource

3.1.12
logical model schema
XML-schema representation of the logical model of a family of resources

3.1.13
logical unit
unit of an audiovisual resource such as a chapter of a DVD that is semantically meaningful to the User of the resource

NOTE Logical units can be atomic or composed of other logical units. A logical unit is represented as a (namespace qualified) element node in the XPath data model.

3.1.14
node-test
as in XPath, specifies the node type and expanded-name of the nodes to be selected by a location step

NOTE In this International Standard, node-tests have been extended to define temporal and/or spatial conditions on the selected nodes resulting in sets of locations. As such, a node-test selects a set of locations in a given dimension.

3.1.15
pointer
string conforming to this specification or to the XPointer Framework specification

3.1.16
pointer part
portion of a pointer that provides a scheme name and some pointer data that conforms to the definition of that scheme

3.1.17
predicate
as defined in XPath, uses arbitrary expressions to further refine the set of nodes selected by a location step

NOTE In this International Standard, a predicate is extended to return a set of locations.

3.1.18
region
connected 2D or 3D area of a logical unit of an audiovisual resource, for instance, a circular patch of a photo that contains the image of a person's face or a rectangular area of a video frame that contains the image of a car

3.1.19
resource
A resource as defined by IETF RFC 3986

3.1.20
scheme
specialized pointer part data format that has a name and is defined in an International Standard

3.1.21
temporal-region
time sequence of regions of a logical unit of an audiovisual resource, for instance, a sequence of boxes that tracks a moving car in a sequence of video frames

3.1.22
time
point or interval within a logical unit of an audiovisual resource, for instance, 10 seconds from the start of an audio CD track or the first minute of the track

3.2 Abbreviations

For the purpose of this part of ISO/IEC 21000, the following abbreviations apply:

FID	Fragment Identifier
IRI	Internationalized Resource Identifier
LM	Logical Model
MIME	Multipurpose Internet Mail Extensions [IETF RFC 2045]
MPEG	Moving Picture Experts Group
MPEG-4	ISO/IEC 14496
MPEG-21	ISO/IEC 21000
NS	Namespace
URI	Uniform Resource Identifier [IETF RFC 3986]
URL	Uniform Resource Locator [IETF RFC 1738]
URN	Uniform Resource Name [IETF RFC 2141]
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language 1.0 [W3C Recommendation]
XPath	XPath Language 1.0 [W3C Recommendation]
XPointer	XPointer Framework [W3C Recommendation]

3.3 Namespaces

Subclause 4.3 of this specification extends the use of the W3C `xmlns()` pointer to allow interpretation of namespace prefixes in pointers referring to non-XML resources.

Additionally, this specification defines default namespaces for:

- Resolving a scheme name, and
- Resolving Logical Units or attributes used in scheme data.

3.3.1 Default pointer scheme namespace

The pointer schemes that are defined in this specification are defined under the following namespace: `urn:mpeg:mpeg21:2006:01-FID-NS`.

For resources of MPEG Media Types, the specification of a namespace is optional for scheme names. If not specified, the namespace for a pointer scheme is defaulted to be `urn:mpeg:mpeg21:2006:01-FID-NS`.

For instance, in the following URI, the `ffp()` pointer part refers to the normative `ffp()` pointer scheme defined in this specification:

```
myFile.mp21#ffp(item_name='myItem')
```

while in the following URI, the `myPointer()` scheme is a proprietary scheme whose namespace `urn:myNamespace` is specified by the prefix `pref` via an xmlns() pointer part:

```
myFile.mp21#xmlns(pref=urn:myNamespace)pref:myPointer(mySchemeData)
```

3.3.2 Default namespace for pointer data

For resources of MPEG Media Types, the specification of namespace is optional for Logical Units or attributes used in scheme data. If not specified, the namespace for a Logical Unit or an attribute is defaulted to be the namespace for the normative Logical Model of that MPEG Media Type.

For instance, subclause 6.5 of this specification provides a normative Logical Model for MPEG-4 video resources under the namespace `urn:mpeg:mpeg21:2006:01-LM-MP4-NS`. When a URI addresses an MPEG-4 video, if no namespace is specified for a Logical Unit in the pointer data, say `track` in the following example, the default namespace `urn:mpeg:mpeg21:2006:01-LM-MP4-NS` applies, that is:

```
myFile.mp4#mp(/track)
```

is equivalent to:

```
myFile.mp4#xmlns(mp4=urn:mpeg:mpeg21:2006:01-LM-MP4-NS)mp(/mp4:track)
```

4 MPEG URI Fragment Identifier Framework

4.1 Introduction

This clause defines a set of principles for addressing fragments of MPEG media resources. This set of principles is referred to as the MPEG URI Fragment Identifier Framework.

The framework is based on the XPointer framework, as explained in subclause 4.2. In particular, the xmlns() pointer scheme is used to define namespace bindings that apply to the pointer part (subclause 4.3). Furthermore, the framework provides a way to address fragments in a hierarchical manner (subclause 4.4). The generic syntax and semantics of the framework are defined in subclauses 4.5 and 4.7 respectively.

4.2 W3C XPointer Framework

The W3C XPointer Framework recommendation specifies a framework for XML addressing. It is intended to be used as a basis for URI fragment identifiers for XML-based resources. It defines two possible pointer formats:

- Shorthand pointers;
- Scheme-based pointers.

The shorthand pointer, formerly known as a *barename*, consists of a non-colonized name, which is a schema-determined or DTD-determined ID. Scheme-based pointers consist of a scheme name and scheme data in the form of `schemeName(schemeData)`.

4.3 Use of namespaces and of the xmlns() pointer scheme

The framework defined in this specification is intended to be extensible, in that it can be used with proprietary pointer schemes. To ensure unique scheme names, pointer schemes can be qualified with a namespace. In addition, some schemes may require the use of qualified names in the scheme data. For this purpose, this framework uses the `xmlns()` pointer scheme to define namespace bindings.

The use of the W3C `xmlns()` pointer is hereby extended in the MPEG Fragment Identifier Framework to allow the use of namespace prefixes in fragment identifiers addressing non-XML resources. A prefix defined using an `xmlns()` pointer part can be used for qualifying Logical Units or attributes that appear within the scheme data or for qualifying a scheme name.

As in the W3C `xmlns()` specification, a pointer part with the `xmlns()` scheme declares a namespace prefix to be associated with a namespace name. Each pointer part that uses the `xmlns()` scheme contributes a new entry to the namespace binding context. If a pointer part defines a binding for a namespace prefix that already has an entry in the namespace binding context, the new entry overrides the old one.

A pointer part that uses the `xmlns()` scheme never identifies a resource fragment and thus the MPEG URI Fragment Identifier processor evaluation always continues with the next pointer part.

In the following example, the first `xmlns()` pointer part allows the `mp()` part to the right to refer to the title (as `d:title`), the chapter (as `d:chapter`) and the audio (as `d:audio`) of a DVD; Then the second `xmlns()` pointer part allows the `mp()` part to refer to channels (as `au:channel`) of an mp3 file `urn:example:mp3` on the DVD:

```
#xmlns(d=http://example.org/dvd)xmlns(au=urn:example:mp3)mp(/d:DVD/d:title/d:chapter/d:aud
io/au:channel[1])
```

4.4 Hierarchical combination of pointer parts

In some cases, a resource consists of a collection of sub-resources, possibly organized in a hierarchical way. In such a container resource, a fragment identifier may be used to identify a sub-resource. Furthermore, it may be required to identify a fragment of a sub-resource. There is therefore a need for identifying resource fragments in a hierarchical way.

This specification provides a “*” operator for this purpose.

When two consecutive pointer parts are separated by the “*” operator, the fragments located by the first pointer part (to the left of the “*” operator) are used as a context for evaluating the second pointer part (to the right of the “*” operator).

For example, the following URI identifies a time point in a bitstream contained in the file `myFile.mp4`. This bitstream is first located using its `item_name`:

```
http://a.com/d/myFile.mp4#ffp(item_name=myBitstream)*mp(~time('npt','50'))
```

4.5 Syntax

This subclause describes the syntax of MPEG Fragment Identifiers.

The syntax of MPEG Fragment Identifiers is based on the syntax of *scheme-based* pointers of the W3C XPointer Framework. As defined below using EBNF, the syntax of XPointer Framework *scheme-based* pointers is extended to support the “*” operator:

```
Pointer          ::= SchemeBased
SchemeBased      ::= PointerPart ( ( S | '*' )? PointerPart)*
PointerPart      ::= SchemeName '(' SchemeData ')'
SchemeName       ::= QName
SchemeData       ::= EscapedData*
EscapedData      ::= NormalChar | '^(' | '^)' | '^^' | '(' SchemeData ')'
NormalChar       ::= UnicodeChar - [()^]
UnicodeChar      ::= [#x0-#x10FFFF]
```

The symbol `S` is defined in XML1.0. The symbol `QName` refers to qualified names as specified in the W3C recommendation for Namespaces in XML. Each pointer part has a scheme name and contains data enclosed within parentheses.

NOTE If either a left or a right parenthesis occurs in scheme data without being balanced by its counterpart, it must be escaped with a circumflex (^) character preceding it.

4.6 Character escaping

The parameter `SchemeData` supports Unicode characters. However, MPEG Fragment Identifiers are designed to be used in the context of URI references (RFC 3986), which require encoding and escaping of certain characters. MPEG Fragment Identifiers can also appear in XML documents, which impose some escaping requirements of their own. Other contexts might require additional escaping to be applied to MPEG Fragment Identifiers.

4.7 Processing

This subclause describes the behaviour of an MPEG Fragment Identifier processor.

An MPEG Fragment Identifier processor takes as input:

- A string to be used for locating a resource (e.g. a URL), and
- A URI fragment identifier that addresses a portion of the resource.

An MPEG Fragment Identifier processor attempts to evaluate the fragment identifier against the resource, and typically produces as output a set of parameters that can be used by an application for locating the fragments.

If multiple pointer parts are provided, an MPEG Fragment Identifier processor must evaluate them in a left-to-right order.

If two consecutive pointer parts are separated by a "*" operator, the fragments located by the first pointer part are used as a context for evaluating the second pointer part as specified in subclause 4.4.

If several consecutive pointer parts are not separated by a "*" operator, the result of the first pointer part whose evaluation identifies one or more resource fragments is returned by the processor and evaluation stops, as specified in XPointer Framework.

For example, in the following fragment identifier, if the first `mp()` pointer part is not understood by the processor or fails to identify any fragment, the second `ffp()` pointer part is evaluated. If the first `mp()` pointer part identifies one or more fragments, the second `ffp()` pointer part is not evaluated.

```
mp(/CD/track[1])ffp(track_ID=1)
```

When an MPEG Fragment Identifier processor encounters a URI fragment compliant to the URL form for meta boxes as defined by ISO/IEC 14496-12, it shall extract the fragment part before the "*" operator and wrap it into an `ffp()` pointer.

For example, the MPEG Fragment Identifier processor shall convert the following URI:

```
http://a.com/d/myFile.mp4#item_name=myBitstream*mp(~time('npt','50'))
```

To:

```
http://a.com/d/myFile.mp4#ffp(item_name=myBitstream)*mp(~time('npt','50'))
```

The processing of "*"-separated pointer parts is compatible with the processing of ISO/IEC 14496-12 URL forms.

Note that in the case of the above example, an URL form-aware application will first fetch `myFile.mp4` from a.com using HTTP. It then inspects the top-level meta box in `myFile.mp4` and adds the items in it, logically, to its cache of the directory “d” of a.com. It then re-forms the URL as:

```
http://a.com/d/myBitstream#mp(~time('npt','50'))
```

And then locates the 50th second in Normal Play Time of the content of myBitstream.

Note that the `item_name` in the first pointer part has been elevated to a full file name, and the first “*” has been transformed back into a “#”.

5 Pointer schemes

5.1 Introduction

This clause specifies a set of pointer schemes. Each pointer scheme applies to a particular set of Internet Media types:

- The `ffp()` pointer scheme provides a simple mechanism for locating an item or track within an ISO base media file format.

- The `offset()` pointer scheme locates a range of bytes in any bitstream.

- The `mp()` pointer scheme applies to audiovisual multimedia resources and locates a set of spatial, temporal or spatiotemporal regions and/or Logical Units according to a Logical Model of a resource.

- The `mask()` pointer scheme allows addressing of a binary (1-bit deep) mask defined in a resource.

5.2 The `ffp()` pointer scheme

5.2.1 Introduction

The `ffp()` pointer scheme applies to file formats conforming to ISO/IEC 14496-12 and allows the identification of an *item* or a *track* as defined in these formats.

The subclause 8.44.7 of ISO/IEC 14496-12 defines a so-called URL form for meta boxes. The subclauses below describe how URL forms relate to this specification.

5.2.2 Syntax

The syntax of an `ffp` pointer is defined by the following EBNF notation:

```
FileFormatPointer ::= FileFormatPointerSchemeName "(" FileFormatPointerSchemeData ")"
FileFormatPointerSchemeName ::= "ffp"
FileFormatPointerSchemeData ::= TrackId | ItemId | ItemName
TrackId     ::= "track_ID=" Integer
ItemId      ::= "item_ID=" Integer
ItemName    ::= "item_name=" Name
Name        ::= EscapedData
Integer     ::= DIGIT+
```

5.2.3 Semantics

The `ffp()` pointer scheme identifies two types of fragments: *items* and *tracks* of the ISO Base Media File Format.

When identifying an *item*, the scheme makes use of the fields `item_ID` of the `iloc` box or the field `item_name` of the `iinf` box of the ISO Base Media File Format. `item_ID` is an unsigned integer while `item_name` is a string of characters. This string of characters has to comply with the syntax of `EscapedData` as specified in subclause 4.5. The target fragment consists physically of one or several data chunks, which can be located by their offset and length specified in the `iloc` box.

When identifying a *track*, the scheme makes use of the field `track_ID` of the `tkhd` box. `track_ID` is an unsigned integer. The target fragment consists physically of one or several data chunks, which can be located by their offset and length specified in the `stbl` boxes and their sub-boxes.

5.2.4 Relation with URL forms defined in ISO/IEC 14496-12

Subclause 8.44.7 of ISO/IEC 14496-12 specifies a syntax for addressing an *item* in an ISO base file by means of its `item_name` or `item_ID` in the following way:

a) `item_ID=<n>`, identifying the item by its ID (the ID may be 0 for the whole resource);

b) `item_name=<item_name>`, when the item information box is used.

This specification extends the syntax of ISO/IEC 14496-12 in two ways: firstly, it wraps the original fragment identifier as pointer scheme data as in `ffp(item_ID=<n>)`. Secondly, it allows a track to be identified by means of its `track_ID`.

Furthermore, ISO/IEC 14496-12 introduces the use of a "`*`" operator for the hierarchical combination of two pointer parts. The functionality of the "*" operator has been generalized in this specification to allow the concatenation of multiple pointer parts (see subclause 4.4).

5.2.5 Resolution mechanism

For resolving `item_ID`, a `ffp()` processor parses the `iloc` boxes contained in the meta boxes of the file, and scans the list of item entries until it finds an item with the specified `item_ID`. The resulting fragment consists of the concatenation of the item's data chunks located by the offsets and lengths (the `extent_offset` and `extent_length` fields) of each data chunk.

For resolving `item_name`, a `ffp()` processor parses the `iinf` boxes contained in the meta boxes of the file, and scans the list of item entries (`infe` sub-boxes) until it finds an item with the specified `item_name`. The fragment is then located by the corresponding `item_ID`, as described above.

For resolving `track_ID` a `ffp()` processor parses the `tkhd` boxes of the file until it finds the specified `track_ID`. The resulting fragment then consists of the concatenation of the data chunks of the corresponding `stbl` box and sub-boxes.

NOTE In both cases (that is items and tracks), the actual data chunks are located via their offsets and lengths, irrespective of the containing box (which consists of one or several `mdat` boxes).

For example, the URI below locates an item in the `meta` box through its name.

```
http://www.example.com/myfile.mp4#ffp(item_name=file.mp4)
```

In this second example, the URI locates an item in the `meta` box through its ID.

```
http://www.example.com/myfile.mp4#ffp(item_ID=1)
```

In this third example, the URI locates a track in an MP4 file through its ID.

```
http://www.example.com/myfile.mp4#ffp(track_ID=101)
```

5.3 The `offset()` pointer scheme

5.3.1 Introduction

The `offset()` pointer scheme applies to any digital resource and identifies a range of bytes in a data stream.

5.3.2 Syntax

The syntax of an `offset` pointer is defined by the following EBNF notation:

```
OffsetPointer ::= OffsetPointerSchemeName "(" OffsetPointerSchemeData ")"
OffsetPointerSchemeName ::= "offset"
OffsetPointerSchemeData ::= position ("," length)?
position    ::= Integer
length      ::= Integer
Integer     ::= DIGIT+
```

5.3.3 Semantics

The `offset()` pointer scheme addresses a contiguous segment of data in any digital resource by indicating the `position` of the first byte of the segment and its `length`. When the `length` is not provided, the segment is supposed to extend to the end of the resource. The first byte of the resource has position 0, and both `position` and `length` are in bytes.

For example, the following URI locates 100 bytes starting at the 11th byte of the file `myImage.jpg`

```
http://www.example.com/myImage.jpg#offset(10,100)
```

5.4 The `mp()` pointer scheme

5.4.1 Introduction

The `mp()` pointer scheme defines an addressing scheme for multimedia resources. In particular, it can be used for addressing fragments of resources whose Internet media type (or MIME type) is one of:

- audio/mpeg	[RFC3003];
- video/mpeg	[RFC2045,RFC2046];
- video/mp4	[RFC4337];
- audio/mp4	[RFC4337];
- application/mp4	[RFC4337].

The addressing scheme provides two complementary mechanisms for identifying fragments in a multimedia resource.

Firstly, the `mp()` pointer scheme defines a set of so-called axes, which allow the identification of temporal, spatial or spatiotemporal fragments in a multimedia resource, e.g. an audio, an image or a video, independent of the coding format.

Secondly, the `mp()` pointer scheme allows the addressing of fragments of a multimedia resource via a given hierarchical Logical Model of the resource, e.g. a track in an audio CD. Such Logical Models may be standardized or proprietary. This specification defines a syntax based on XPath's location paths for locating Logical Units in such a hierarchical Logical Model. Examples of Logical Models and `mp()` pointers are provided in Annex B.

Together, these two mechanisms allow, for example, the addressing of a time fragment in an audio CD track.

The following EBNF notation defines the `mp()` pointer scheme:

```
MediaPointer              ::= MediaPointerSchemeName "(" MediaPointerSchemeData ")"
MediaPointerSchemeName    ::= "mp"
```

`MediaPointerSchemeData` is defined in the following subclauses as an extension of the W3C XPath Abbreviated Syntax.

5.4.2 Evaluation context

The `mp` fragment identifier is evaluated to a set of media locations. A processor shall initialize the evaluation context to include the following information before evaluating an expression:

- A location (the context location), initialized to the root Logical Unit of the Logical Model;
- A non-zero context position, initialized to 1;
- A non-zero context size, initialized to 1;
- An empty set of variable bindings. Use of a variable reference in an expression results in failure of the pointer part, that is, variables are not supported;
- A library of functions which minimally consists of XPath Core functions library;
- A namespace binding context consisting of namespace bindings made by any `xmlns()` pointer to the left of the current `mp()` pointer.

The evaluation depends on the logical model of the content. Given a published model, the `mp` fragment identifier enables one to navigate through the logical structure of a resource by means of XPath-like expressions.

5.4.3 Extensions to the XPath data model

An XPath location path consists of a succession of '/'-separated location steps. Each location step contains an axis, a node-test and an optional list of predicates:

- An axis specifies the tree relationship between the nodes selected by the location step and the context node;
- A node-test specifies the type of the selected nodes;
- Predicates can use any named function from the XPath Core Function Library (defined in clause 4 of the XPath Recommendation) to further refine the set of selected nodes;

Therefore, the type of nodes selected by an XPath expression is determined by the axis and node-test. The predicates plus the functions they use are for filtering out unwanted nodes selected by the axis and node-test.

The `mp()` scheme extends XPath by:

- Generalising the XPath concept of node to the concept of location;
- Generalising the XPath concept of node types to the concepts of location types;
- Generalising the XPath concept of node-sets to the concepts location-sets, which are sets of locations in the same way that XPath node sets are sets of nodes;

- Generalising each of the XPath axes (e.g. child, descendant, attribute, etc) as one dimension of the current context to allow a node-test to select a set of locations that bear a certain hierarchical relation in that dimension;

- Generalising the XPath concept of node-test to the concept of location-test;

- Adding rules for establishing the evaluation context;

- Adding the set of new dimensions/axes and node-tests listed in Table 1 below:

Table 1 — New axes and node-tests

Axes (dimensions)		**Node-tests**
Syntax	**Abbreviated syntax**	
`temporal`	~	`TimeNodeTest`
`spatial`	~	`RegionNodeTest` `VolumeNodeTest`
`spatiotemporal`	~	`MovingRegionNodeTest`

The same abbreviated syntax is defined for all spatiotemporal axes as all the node-tests that apply to these axes have unique names.

For example, the following fragment identifiers:

```
mp(temporal::time('npt','30'))
mp(spatial::region(rect(20,20,40,40)))
mp(spatiotemporal::moving-region(rect(0,0,5,5),pt(10,10,t(5)),pt(20,20)))
```

Are equivalent to:

```
mp(~time('npt','30'))
mp(~region(rect(20,20,40,40)))
mp(~moving-region(rect(0,0,5,5),pt(10,10,t(5)),pt(20,20)))
```

5.4.3.1 Definition of a location

A location of type time is defined in terms of two data items:

- A reference to a node, and

- One or two time indexes, which represent temporal offsets to the start of the timeline of the associated node. Two temporal offsets define a time interval.

A location of type region is defined by two data items:

- A reference to a node, and

- A set of geometrical parameters defining a 2D or 3D region with respect to the spatial extent of the associated node.

A location of type temporal-region is defined by three data items:

- A reference to a node, and

- Two time indexes which represent a time interval on the timeline of the associated node, and

- a set of time-dependent geometrical parameters defining a sequence of 2D or 3D regions (as defined in subclauses 5.4.5.6 and 5.4.5.7) in time with respect to the spatial extent of the associated node within the time interval.

Given these definitions, two time locations are identical if they are associated with the same node and have the same index(es); two region locations are identical if they are associated with the same node and have the same geometrical parameters; two temporal-region locations are identical if they are associated with the same node and have the same time indexes and geometrical parameters.

The temporal, spatial and spatiotemporal axes of a location are defined as follows:

- The child, descendant, descendant-or-self, preceding-sibling, following-sibling, preceding, following, attribute, and namespace axes are empty;

- The self axis contains the location itself;

- The parent axis contains the node referred to by the location;

- The ancestor axis contains the node referred to by the location and its ancestors;

- The ancestor-or-self axis contains the location itself, the node referred to by the location, and its ancestors.

The string-value of time, region and temporal-region locations are empty. In addition, these locations do not have an expanded-name.

5.4.3.2 Logical Model

The evaluation requires a Logical Model of the content. Given a Logical Model, the `mp()` pointer enables one to navigate through the logical structure of a resource by means of expressions based on the XPath syntax.

An `mp` processor evaluates the `mp()` scheme data against the extended XPath data model of the resource as defined in subclause 5.4.3.

The Logical Model can be specified by the first Logical Unit in the Location Path of the fragment identifier. Otherwise, if no Logical Model is specified, the content is assumed to have spatio-temporal characteristics and is treated as having an implicit Logical Model with a single Logical Unit, the root node, which consists of the entire resource.

In this specification, an XML description derived from a Logical Model Schema is used for representing a resource. Such representation is referred to as a Logical Model Description. This specification, however, does not mandate any particular method of instantiation.

Consider for example a Logical Model of an audio CD that defines the CD as a list of `track` Logical Units (Figure 2). When an `mp` processor evaluates the expression `/CD/track/~time('npt','30')`, it successively considers the following Location Steps:

- The first Logical Unit, `CD`, of the resource, which also identifies the CD's Logical Model;

- The `track` Logical Units;

- Time points located 30 seconds from the start of each track.

The resulting fragments are a set of Locations that reference a time point located 30 seconds from the start of each track.

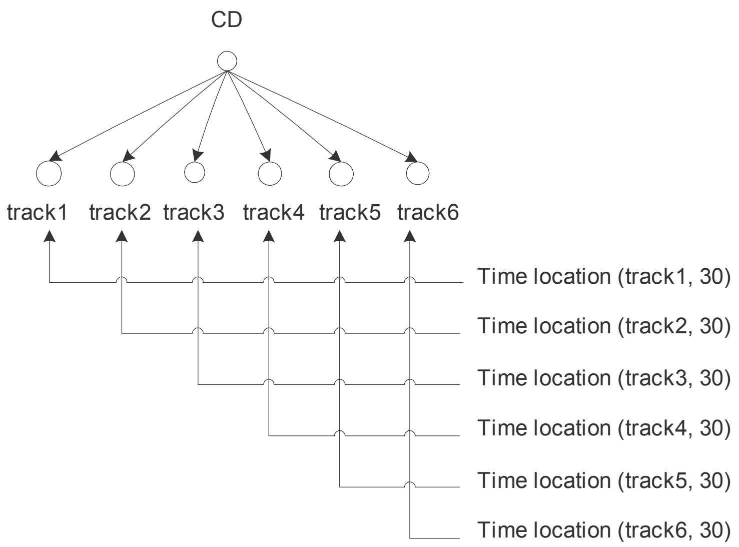
CD
track1
track2
track3
track4
track5
track6
Time location (track1, 30)
Time location (track2, 30)
Time location (track3, 30)
Time location (track4, 30)
Time location (track5, 30)
Time location (track6, 30)

Figure 2 — Example Logical Model instantiation

5.4.4 Evaluation model

The spatio-temporal characteristics of a Location are that of its associated node offset by the Location's own temporal and/or spatial parameters. A node that represents a time-based fragment has a timeline while a node that represents a fragment with spatial dimensions has spatial extent. A Location is said to have a timeline if its associated node represents a time-based fragment. A Location is said to have spatial extent if its associated node represents a fragment with spatial dimensions.

The spatio-temporal characteristics of a node are the union of those of its descendant nodes.

If a node has descendants, its timeline is formed by concatenating the timelines of all of its time-based descendants in a depth-first order, while its spatial extent is the union of the spatial extents of its descendants as illustrated in Figure 3.

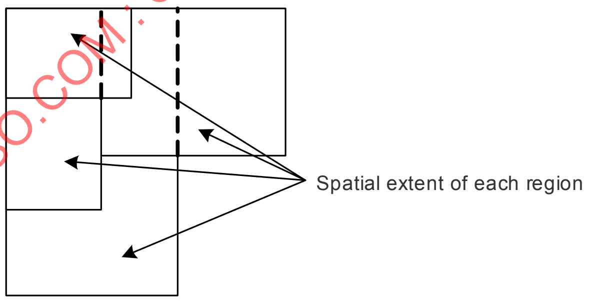

Figure 3 — Union of the spatial extents of the descendants of a node

5.4.4.1 Evaluation of a Location along a temporal axis

When evaluating a Location against a location step along the temporal axis, if the Location has a timeline, the location step is evaluated against the Location's timeline. A Location that has no timeline evaluates to an empty Location-set along the temporal axis.

5.4.4.2 Evaluation of a Location along a spatial axis

When evaluating a Location against a location step along the spatial axis, if the Location has a spatial extent, the location step is evaluated against the spatial extent of the Location. A Location that has no spatial extent evaluates to an empty Location-set along the spatial axis.

5.4.4.3 Evaluation of a Location along a spatiotemporal axis

When evaluating a Location against a location step along the spatiotemporal axis, if the Location has a timeline and a spatial extent, the location step is evaluated against the timeline and spatial extent of the Location. A Location with no timeline and/or no spatial extent evaluates to an empty Location-set along the spatiotemporal axis.

In the following example, the location step `/view` will output a Location-set of views:

```
mp(/view/~time('npt','30','40'))
```

Assuming that the Logical Unit `view` contains a sequence of time-based `frame` Logical Units, `view` is also considered as having a timeline. The location step `/~time('npt','30','40')` is evaluated against the timeline of each `view` in the output Location-set of the previous step. The timeline of each `view` is formed by concatenating the timeline of all its child `frame`. The `time` Node-test selects a 10 second interval of each `view` starting from 30 second into the `view`. In another word, each `view` in the final Location-set consists of 10 seconds (or less) of frames.

5.4.5 Scheme data syntax

5.4.5.1 Temporal axis

On the `temporal` axis, the context location is interpreted as a set of time points or ranges.

5.4.5.2 Time node-test

The `TimeNodeTest` selects a time point or a time range in a resource. The syntax of `TimeNodeTest` is specified below in EBNF notation:

```
TimeNodeTest   ::=    "time(" TimeNotation   ")"
TimeNotation   ::=    time_scheme "," start_time  ("," end_time )?
start_time     ::=    "'" time_spec  "'"
end_time       ::=    "'" time_spec  "'"
```

The `TimeNodeTest` accepts up to three parameters: a `time_scheme`, an inclusive `start_time` and an exclusive `end_time` that comply with the specified time scheme (defined in 5.4.5.3). The third parameter `end_time`, if specified, allows addressing a time range of a resource.

Both `start_time` and `end_time` must be greater than or equal to zero. A `start_time` of zero refers to the beginning of the parent Logical Unit. The parameter `end_time`, if specified, must be greater than `start_time`. If `end_time` is beyond the length of the parent Logical Unit, a time range up to the end of the parent Logical Unit will be selected.

In the following example, the `mp` fragment identifier will select the first sixty seconds - or the entire track if it is less than sixty seconds - of all the tracks:

```
mp(/CD/track/~time('npt','0','60'))
```

5.4.5.3 Time schemes

The time_scheme can be any of the following:

```
time_scheme    ::= "'" npt-type | smpte-type | mp7-type | utc-type | fractional | other-
type "'"
npt-type    ::= "npt"
smpte-type ::= "smpte-24" | "smpte-24-drop" | "smpte-25" | "smpte-30" | "smpte-30-drop"
| "smpte-50" | "smpte-60" | "smpte-60-drop"
mp7-type    ::= "mp7t"
utc-type    ::= "clock"
fractional ::= "frac"
```

npt-type refers to Normal Play Time as defined in ISO/IEC 13818-6 (DSM-CC). smpte-type refers to the Society of Motion Picture and Television Engineers time codes as specified by the SMPTE time and control code standard. The mp7-type refers to MPEG-7 MediaTimePointType as defined in ISO/IEC 15938-5 (MDS). utc_type is the Universal Time Code which gives wall-clock time as specified by the ISO 8601 standard.

The time scheme fractional for fractional notations is defined in subclause 5.4.5.4 of this specification.

The syntax also allows the use of proprietary or non-normative time schemes (other-type). However, an application that does not support the proprietary or non-normative time scheme will stop the evaluation. The parameter other_type shall be specified using the syntax of EscapedData as specified in subclause 4.5:

```
other-type::= EscapedData
```

5.4.5.4 Time specification

The value of the parameter time_spec depends on the time_scheme:

```
time_spec      ::= npt_time_spec | spmte_time_spec | mp7t_time_spec |
                   utc_time_spec | frac_time_spec | other_time_spec
```

npt_time_spec, spmte_time_spec, mp7t_time_spec and utc_time_spec are compliant to standard time schemes reproduced in Annex C.

The parameter frac_time_spec is defined by the following syntax:

```
frac_time_spec ::= a "," b
a              ::= DIGIT+
b              ::= DIGIT+
```

The parameter a specifies the time as an integer count of a clock that ticks b times per second.

Time data for other_time_spec shall be specified using the syntax of EscapedData as specified in subclause 4.5:

```
other_time_spec ::= EscapedData
```

In the following example, the fragment identifier specifies a time point of an audio resource using MPEG-7 MediaTimePointType:

```
mp(~time('mp7t','T17:30:45:2F10'))
```

5.4.5.5 Spatial axis

With the `spatial` axis, the context location is interpreted as pixels of a 2D image or voxels of a 3D image.

5.4.5.6 Region node-test

The `RegionNodeTest` selects a 2D region bounded by the curve specified by the parameter `shape`. The syntax of `RegionNodeTest` is specified below in EBNF notation:

```
RegionNodeTest ::= "region(" shape ")"

shape ::= BoundingCurve ("," Resolution )?
BoundingCurve  ::= Ellipse |  Polygon  | Rectangle

Ellipse        ::= "ellipse("  AbsCoords "," AbsCoords "," AbsCoords ")"
Polygon        ::= "polygon("  AbsCoords ","  RelCoords (","  RelCoords )+  ")"
Rectangle      ::= "rect("AbsCoords "," AbsCoords ("," AbsCoords )? ")"

AbsCoords      ::= Integer "," Integer
RelCoords      ::= Integer "," Integer
Resolution     ::= "range(" Integer "," Integer ")"
Integer        ::= DIGIT+
```

The origin is set at the top-left corner of the image with the *x*- and *y*- axis coordinates increasing to the right and down. The two `Resolution` parameters give the *x*- and *y*- resolutions of the source from which the bounding curve is determined. It allows the possibility of using a spatial fragment identifier to locate an equivalent region of the content encoded in different resolutions.

Absolute and relative coordinates are specified using two integer values. The first value is relative to the *x*-axis and the second value is relative to the *y*-axis.

In the case of an `Ellipse`, the absolute coordinates of three vertices of its circumscribing rectangle are specified.

In the case of a `Polygon`, only the first pair of coordinates uses absolute *x* and *y* coordinates values. All the subsequent coordinates are specified by the relative values `RelCoords`, which are the differences from the *x* and *y* coordinates of the previous vertex.

In the case of a `Rectangle`, if two pairs of coordinate values are specified, the edges of the `Rectangle` are assumed to be parallel to the *x* and *y* axes. When three pairs of coordinate values are specified, the fourth vertex can easily be computed.

In the following example, the fragment identifier locates a 20x20 square region of an image:

```
mp(~region(rect(20,20,40,40)))
```

5.4.5.7 Volume Node-test

The `VolumeNodeTest` selects a 3D volume bounded by a bounding surface specified by the BoundingSurface parameter. The syntax of `VolumeNodeTest` is specified below in EBNF notation:

```
VolumeNodeTest ::= "volume("  BoundingSurface  ("," Resolution)?  ")"
BoundingSurface ::= Ellipsoid | 3D_Object | Box

Ellipsoid  ::= "ellipsoid("  AbsCoords "," AbsCoords "," AbsCoords "," AbsCoords ")"
3D_Object  ::= "object3d(" AbsCoords "," RelCoords ","
                                 RelCoords (","  RelCoords )+ ")"
Box        ::= "box("         AbsCoords "," AbsCoords ","
                                 AbsCoords ("," AbsCoords )?  ")"

AbsCoords  ::= Integer "," Integer "," Integer
RelCoords  ::= Integer "," Integer "," Integer

Resolution ::= "range("  Integer   ","  Integer  ","  Integer  ")"

Integer    ::= DIGIT+
```

The origin is set at the top-left-front corner of a frame with the *x*-, *y*- and *z*-axis coordinates increasing to the right, down and inward. Coordinates are specified in (integral) voxel values. The three `Resolution` parameters give the *x*- *y*- *and z*- resolutions of the source from which the bounding curve is determined. It allows the possibility of using a spatial fragment identifier to locate an equivalent region of the content encoded in different resolutions.

Absolute and relative coordinates are specified using three integer values. The first value is relative to the *x*-axis, the second value is relative to the *y*-axis and the third value is relative to the *z*-axis.

In the case of an `Ellipsoid`, the absolute coordinates of four non-coplanar vertices of its circumscribing box are specified.

The type of surface generated for a `3D_Object` is a 3D polygon defined by at least four non-coplanar vertices. Only the first pair of coordinates uses absolute *x* and *y* coordinates values. All the subsequent coordinates are specified by the relative values `RelCoords`, which are the differences from the *x, y* and *z* coordinates of the previous vertex.

In the case of a `Box`, if three pairs of coordinate values are specified, the edges of the `Box` are assumed to be parallel to the *x, y* and *z* axes. When four non-coplanar pairs of coordinate values are specified, the other vertices can easily be computed.

5.4.5.8 Spatiotemporal axis

With the `Spatiotemporal` axis, the context location is interpreted as sets of 2D regions in time.

5.4.5.9 Moving-region node-test

The `MovingRegionNodeTest` supports:

- Various shapes of regions;
- Movement with constant and non-constant velocity;
- Movement along non-linear paths;
- Non-rigid regions, i.e. regions to which a 2D transform can be applied during the movement.

The syntax of `MovingRegionNodeTest` is specified below in EBNF notation:

```
MovingRegionNodeTest ::= "moving-region(" shape "," path ")"
```

Each sub-part of this syntax is detailed in the following subclauses.

5.4.5.9.1 Specification of the `shape`

The `spatiotemporal` axis selects, within the current video context, a sequence of 2D regions that are bounded by the specified bounding shape (Figure 4). The origin corresponds to the top-left corner of a frame with the *x*- and *y*- axis coordinates increasing to the right, and down. Coordinates are specified in (integral) pixel values. The bounding shape is identical to the parameter `shape` defined for the `RegionNodeTest` in subclause 5.4.5.6.

5.4.5.9.2 Specification of the `path`

The moving-region is updated along a `path`. The `path` specifies the trajectory of a particular point on the bounding shape. This particular point, called the *attachment point*, is different for different shapes.

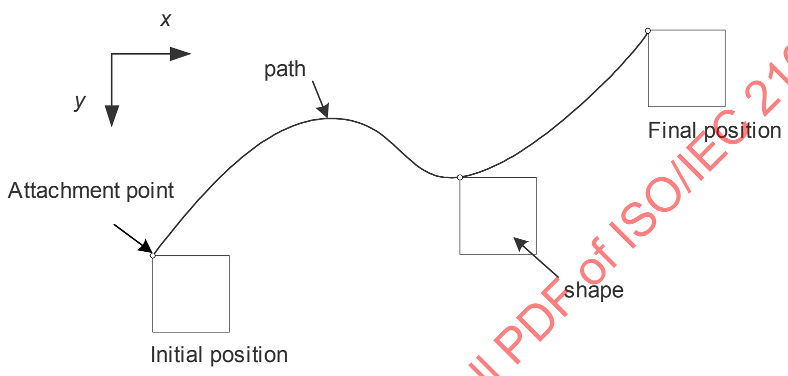

Figure 4 — Moving region model

The *attachment point*s for the different shapes are defined in Table 2 below:

Table 2 — Attachment points

shape	Attachment point
Ellipse	Centre
Polygon	First point
Rectangle	Top-left corner

The specification of the `path` supports:

- The specification of one or more key points;
- The description of a type of interpolation (subclause 5.4.5.9.3);
- The description of an affine transformation (subclause 5.4.5.9.4);
- The description of key time points (subclause 5.4.5.9.5).

The syntax of the `path` is defined below in EBNF notation:

```
path ::= KeyPoint+
KeyPoint ::= "pt(" x "," y ("," ax "," ay )?
                           ("," timepoint )? ("," transformation )? ")"
x   ::= Integer
y   ::= Integer
ax  ::= Float
ay  ::= Float
timepoint  ::= "t(" TimeNotation ")"

Integer    ::= DIGIT+
Float      ::= ( "-" )? DIGIT+ ( "." DIGIT+ )?
```

The syntax of `TimeNotation` is defined in 5.4.5.2. The other components of this syntax are detailed in the following subclauses.

5.4.5.9.3 Specification of the interpolation

The model of interpolation allows the description of the trajectory of a point moving between two successive key points at time t_0 and t_1 respectively at constant speed or constant acceleration, over the time interval. The syntax supports the specification of an interpolation of first or second order between the two key points. The parameters `ax` and `ay` in the `path` specify the type of interpolation:

$$\begin{cases} x(t) = \frac{1}{2}a_x(t-t_0)^2 + v_x(t-t_0) + x_0 \\ y(t) = \frac{1}{2}a_y(t-t_0)^2 + v_y(t-t_0) + y_0 \end{cases}$$

Where (x_0, y_0) are the coordinates of the first key point at t_0, and v_x and v_y are the velocities in the x and y directions at t_0, and a_x and a_y denote the accelerations in the x and y directions. Note that, the velocities v_x and v_y can be calculated from the other parameters, as:

$$\begin{cases} v_x = \frac{x_1 - x_0}{t_1 - t_0} - \frac{1}{2}a_x(t_1 - t_0) \\ v_y = \frac{y_1 - y_0}{t_1 - t_0} - \frac{1}{2}a_y(t_1 - t_0) \end{cases}$$

In addition, if the interpolation is linear, parameters a_x and a_y are both zero and can be omitted.

For example, in the following expression, the type of interpolation is linear by default:

```
pt(10,10) pt(250,300)
```

Different interpolations can be specified for different time intervals along the path. Whenever an interpolation is defined, it is applied to the portion of path delimited by the previous key point and the current key point (Figure 5).

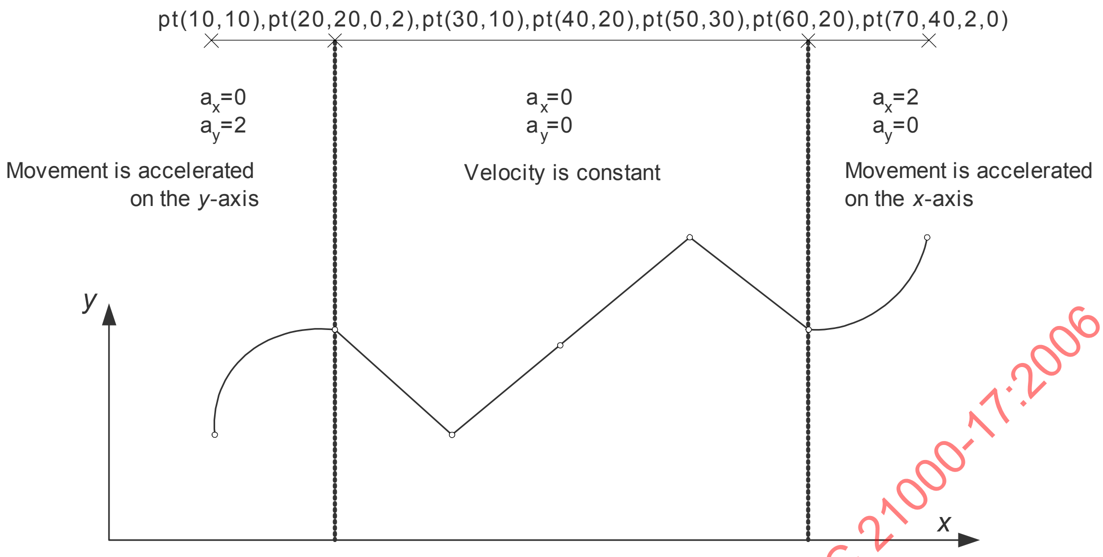
pt(10,10),pt(20,20,0,2),pt(30,10),pt(40,20),pt(50,30),pt(60,20),pt(70,40,2,0)
a_x=0
a_y=2
Movement is accelerated on the y-axis
a_x=0
a_y=0
Velocity is constant
a_x=2
a_y=0
Movement is accelerated on the x-axis
y
x

Figure 5 — Successive types of interpolations along a path

5.4.5.9.4 Specification of the transformation

The term `transformation` specifies an affine 2D transformation applied to the bounding shape of the moving region. The transformation can be a rotation, a uniform scaling, a non-uniform scaling, a skewing or a combination of these:

```
transformation ::= "tr(" rotation | scaling | scaling_rotation |
                         scaling_skew_rotation  | ""  ")"
rotation    ::= Angle
scaling     ::= Float "," Float
scaling_rotation      ::= Angle "," ScaleX "," ScaleY
scaling_skew_rotation ::= Angle "," ScaleX "," ScaleY "," SkewCoeff

Angle          ::= DIGIT+
ScaleX         ::= UnsignedFloat
ScaleY         ::= UnsignedFloat
SkewCoeff      ::= Float

Float     ::= ( "-" )? DIGIT+  ( "." DIGIT+ )?
UnsignedFloat ::= DIGIT+ ("." DIGIT+ )?
```

The models of transformations are the following:

Skew: $\begin{pmatrix} 1 & \texttt{SkewCoeff} \\ 0 & 1 \end{pmatrix}$

Rotation: $\begin{pmatrix} \cos(\texttt{Angle}) & \sin(\texttt{Angle}) \\ -\sin(\texttt{Angle}) & \cos(\texttt{Angle}) \end{pmatrix}$

Scaling: $\begin{pmatrix} \texttt{ScaleX} & 0 \\ 0 & \texttt{ScaleY} \end{pmatrix}$

The parameter `Angle` is measured anti-clockwise and specifies a rotation in degrees of the shape around its *attachment point*.

The `ScaleX` parameter is a scaling coefficient to be applied to the *x* coordinate and `ScaleY` is a scaling coefficient to be applied to the *y* coordinate. The `SkewCoeff` parameter is a skew coefficient.

In a `scaling_rotation` transformation, the scaling is performed prior to the rotation. In a `scaling_skew_rotation` transformation, the scaling is performed before the skewing that is performed prior to the rotation.

Multiple transformations can be specified along the path. A transformation specifies the final transformation to the bounding shape when it reaches the `KeyPoint` that specifies the transformation.

The transformation proceeds uniformly with time between two key points, which are not necessarily successive. For example, a region will be rotated by half the amount and scaled by half the amount in each spatial dimension when the time is half way between the start and end time of the transformation.

The special form `tr()` refers to an idempotent transformation (or *Identity*), i.e no transform is applied.

If no transformation is specified at the last `KeyPoint`, then an idempotent transformation applies.

For example, in the path illustrated in Figure 6, a rotation is applied between point `(10,10)` and point `(20,20)`, no transform is applied between point `(20,20)` and point `(50,30)` and a scaling is applied between point `(50,30)` and point `(70,40)`:

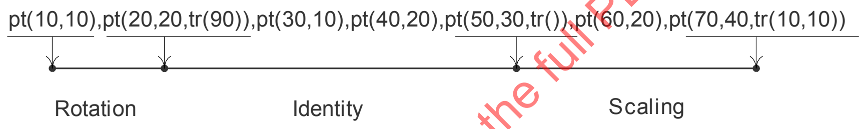
pt(10,10),pt(20,20,tr(90)),pt(30,10),pt(40,20),pt(50,30,tr()),pt(60,20),pt(70,40,tr(10,10))
Rotation
Identity
Scaling

Figure 6 — Successive transforms of the shape

5.4.5.9.5 Specification of key time points

The time interval of the moving-region can be specified by a preceding location step that uses the `temporal` axis defined in 5.4.5.1. Additionally, a `timepoint` for each `KeyPoint` on the path of the moving-region can be specified.

For example, in the following fragment identifier, the context for the moving region is restricted to the time interval between 10 and 30 seconds NPT:

```
mp(/~time('npt','10','30')/~moving-region(rect(0,0,5,5),pt(10,10,t('npt','5')),pt(20,20)))
```

The time point `t(5)` is relative to this context and so the moving-region is a square whose attachment point moves from point (0,0):

- To point `(10,10)` at time 15s NPT;
- To point `(20,20)` at time 30s NPT.

In the next example, as there is no temporal axis to refine the current context, the time point `t(5)` refers to the 5th second from beginning of the video:

```
mp(/~moving-region(rect(0,0,5,5),pt(10,10,t('npt','5')),pt(20,20)))
```

Finally, if the moving-region contains several `KeyPoints` but does not specify any `timepoint`, the points are assumed to be evenly spread in time within the time interval of the current context.

For example, in the following fragment identifier:

```
mp(/~time('npt','0','30')/~moving-region(rect(0,0,5,5),pt(10,10),pt(20,20),pt(30,30)))
```

- Point `(0,0)` is reached at time 0s;
- Point `(10,10)` is reached at time 10s;
- Point `(20,20)` is reached at time 20s;
- Point `(30,30)` is reached at time 30s.

5.5 The `mask()` pointer scheme

5.5.1 Introduction

The `mask()` pointer scheme defines an addressing scheme for multimedia resources whose Internet media type (or MIME type) is one of:

- video/mp4	[RFC4337];
- video/mpeg	[RFC2045, RFC2046].

It allows addressing of a binary (1-bit deep) mask defined in a resource. This mask is meant to be applied to a video resource. The video resource may itself be the resource that contains the mask. The scheme supports multiple formats for specifying a mask.

5.5.2 Syntax

The syntax of a `mask` pointer is defined by the following EBNF notation:

```
MaskPointer ::= MaskPointerSchemeName "(" MaskPointerSchemeData ")"
MaskPointerSchemeName::= "mask"
MaskPointerSchemeData::= Uri ( "," Handler )?
Handler ::= "mpeg"
```

5.5.3 Semantics

The parameter `Uri` specifies the location of the mask. `Uri` may be relative to the video resource or absolute and may contain a fragment identifier. Note that all URI characters that are not supported by the URI fragment character set shall be escaped.

The parameter `Handler` optionally specifies how the mask shall be interpreted. By default, its value is `mpeg` which means that white areas in the mask should be selected regions of the video resource, and conversely black areas, and areas outside the mask, should be unselected regions of the video resource.

If the overall dimensions of the mask are not the same as the overall dimensions of the video, the selected region is obtained by aligning the top-left corners of the mask and the video and then applying the mask onto the video. If the duration of the mask is shorter than the duration of the video, only a temporal portion of the video that equals to the duration of the mask is selected. Conversely, if the duration of the mask is beyond the duration of the video, the selection is only based on the temporal portion of the mask that equals to the duration of the video.

In the following example, the fragment identifier selects a portion of an MPEG-4 video, `myVideo.mp4,` using a mask defined in the first `track` of the same MPEG-4 video:

```
myVideo.mp4#mask(%23ffp(track_ID=1),mpeg)
```

Note that the second # character needs to be escaped using the notation %23.

In the next example, the fragment identifier selects a portion of an MPEG-4 video, myVideo.mp4, using a mask defined in the track which track_ID equals one of another MPEG-4 video located at the URL http://www.example.com/secondVideo.mp4:

```
myVideo.mp4#mask(http://www.example.com/secondVideo.mp4%23ffp(track_ID=1),mpeg)
```

6 Logical Model representation

6.1 Introduction

This clause defines tools for representing Logical Models for audiovisual media to enable the addressing of fragments of media content. A Logical Model of an audiovisual resource provides an abstraction of the resource that is meaningful to the User of the resource. For instance, a Logical Model for DVDs might specify that a DVD contains Titles, Menus, Chapters and so forth. Note that the logical structure is not required to map directly to any physical structure.

The tools consist of a collection of standardized schema components specified using XML Schema. These schema components are to be used for defining the Logical Units of a Logical Model, their spatiotemporal properties and their hierarchical structure.

The following subclauses describe the schema components for Logical Models. Examples of uses of these tools are given in Annex B.

6.2 Schema for Logical Model representation

This subclause specifies the schema tools that facilitate the description of Logical Models. The tools consist of a hierarchy of base types defined in the following XML schema:

```
<schema
targetNamespace="urn:mpeg:mpeg21:2006:01-LM-NS"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:lm="urn:mpeg:mpeg21:2006:01-LM-NS"
elementFormDefault="qualified" attributeFormDefault="unqualified"
version="ISO/IEC 21000-17"
id="lmd.xsd">

    <!-- ######################################## -->
    <!--        Base LogicalUnitType              -->
    <!-- ######################################## -->
    <complexType name="LogicalUnitBaseType" abstract="true"/>

    <!-- ######################################## -->
    <!--        LogicalUnitType                    -->
    <!-- ######################################## -->
    <complexType name="LogicalUnitType">
        <complexContent>
            <extension base="lm:LogicalUnitBaseType"/>
        </complexContent>
    </complexType>

    <!-- ######################################## -->
    <!--        spatialLogicalUnitType            -->
    <!-- ######################################## -->
    <complexType name="SpatialLogicalUnitType">
        <complexContent>
            <extension base="lm:LogicalUnitType">
                <attribute name="spatial" type="boolean" fixed="true"
                 form="qualified"/>
            </extension>
        </complexContent>
    </complexType>

    <!-- ######################################### -->
    <!--       temporalLogicalUnitType            -->
    <!-- ######################################### -->
    <complexType name="TemporalLogicalUnitType">
        <complexContent>
            <extension base="lm:LogicalUnitType">
                <attribute name="temporal" type="boolean" fixed="true"
                 form="qualified"/>
            </extension>
        </complexContent>
    </complexType>

    <!-- ######################################### -->
    <!--       spatiotemporalLogicalUnitType        -->
    <!-- ######################################### -->
    <complexType name="SpatiotemporalLogicalUnitType">
        <complexContent>
            <extension base="lm:LogicalUnitType">
                <attribute name="spatiotemporal"
                 type="boolean" fixed="true" form="qualified"/>
                <attribute name="spatial" type="boolean" fixed="true"
                 form="qualified"/>
                <attribute name="temporal" type="boolean" fixed="true"
                 form="qualified"/>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Note that a schema document has a `version` attribute, the value of which is “ISO/IEC 21000-17”. Furthermore, an informative identifier is given as the value of the `id` attribute of the schema component. The value of this identifier is non-normative and is a convention used by MPEG-21 specifications to allow other documents to reference this schema.

6.3 Semantics

The grammar supplies a top-level abstract type `LogicalUnitBaseType`, which is extended into four types, described in Table 3 below:

Table 3 — Logical Model types

Name	*Definition*
`LogicalUnitType`	A type for declaring a Logical Unit with no specific processing attribute. Its purpose may be for instance to contain other Logical Units.
`SpatialLogicalUnitType`	A type for declaring a Logical Unit that has a spatial extent.
`TemporalLogicalUnitType`	A type for declaring a Logical Unit that has a temporal extent.
`SpatioTemporalLogicalUnitType`	A type for declaring a Logical Unit that has both a spatial and a temporal extent.

NOTE The behaviour of the processor is undefined for a Logical Model Schema which contains a type that is neither one of the four types defined above nor derived from one of them.

6.4 Schema for schemas

This clause specifies a mechanism for specifying the root element of a Logical Model. This is done by means of an extension to XML Schema in the following schema for schemas. The extension consists in an attribute `rootElement` that can be used as an attribute of the `schema` element of an XML Schema for specifying which global element is the root Logical Unit.

```
<schema
   targetNamespace="urn:mpeg:mpeg21:2006:01-LM-NS"
   xmlns:lm="urn:mpeg:mpeg21:2006:01-LM-NS"
   xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified"
   version="ISO/IEC 21000-17">

   <attribute name="rootELement" type="QName"/>

</schema>
```

The use of the `rootElement` attribute is illustrated in subclause 6.5, which also provides the normative Logical Model for MPEG-4.

6.5 Logical Model for MPEG-4 files

This clause provides a Logical Model for files whose Internet Media type is `video/mp4`, that is MPEG-4 files. The Logical Model for MPEG-4 files defines a root Logical Unit `mp4` which contains zero or more `tracks` and at least one `scene`. Both `scene` and `track` can have attached meta-data. Logical Units `track` can be of various types (video, audio, hint, scene, OD, mpeg-7, etc), which are defined by the attribute `hType`, and hold a `track_ID` attribute that corresponds to the field `track_ID` of the `tkhd` box as defined in ISO/IEC 14496-12.

Logical Units `track` and `scene` hold spatiotemporal processing attributes. Logical Units `metadata` do not have spatio/temporal processing attributes.

```
<!-- ISO/IEC 21000-17 Logical Model for MPEG-4 files -->
<xs:schema targetNamespace="urn:mpeg:mpeg21:2006:01-LM-MP4-NS"
   xmlns:lm="urn:mpeg:mpeg21:2006:01-LM-NS"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns="urn:mpeg:mpeg21:2006:01-LM-MP4-NS"
   elementFormDefault="qualified" attributeFormDefault="unqualified"
   version="ISO/IEC 21000-17"
```

```
  id="mp4.xsd"
  lm:rootElement="mp4">

 <!-- import schema for Logical Model descriptions -->
 <xs:import namespace="urn:mpeg:mpeg21:2006:01-LM-NS" schemaLocation="lm.xsd"/>

 <!-- ######################################### -->
 <!--       Logical Unit metadata (global)      -->
 <!-- ######################################### -->
 <xs:element name="metadata" type="lm:LogicalUnitType"/>

 <!-- ######################################### -->
 <!--       root Logical Unit mp4           -->
 <!-- ######################################### -->
 <xs:element name="mp4">
     <xs:complexType>
         <xs:complexContent>
             <xs:restriction base="lm:LogicalUnitType">
                 <xs:sequence>
                     <xs:element name="scene" maxOccurs="unbounded">
                         <xs:complexType>
                             <xs:complexContent>
                                 <xs:extension
                                  base="lm:SpatiotemporalLogicalUnitType">
                                     <xs:sequence>
                                         <xs:element ref="metadata" minOccurs="0"/>
                                     </xs:sequence>
                                 </xs:extension>
                             </xs:complexContent>
                         </xs:complexType>
                     </xs:element>
                     <xs:element name="track" minOccurs="0" maxOccurs="unbounded">
                         <xs:complexType>
                             <xs:complexContent>
                                 <xs:restriction
                                  base="lm:SpatiotemporalLogicalUnitType">
                                     <xs:sequence>
                                         <xs:element ref="metadata" minOccurs="0"/>
                                     </xs:sequence>
                                     <xs:attribute name="track_ID" type="xs:int"/>
                                     <xs:attribute name="htype" type="handlerType"/>
                                 </xs:restriction>
                             </xs:complexContent>
                         </xs:complexType>
                     </xs:element>
                 </xs:sequence>
             </xs:restriction>
         </xs:complexContent>
     </xs:complexType>
 </xs:element>

 <!-- ######################################### -->
 <!--    Handler type                        -->
 <!-- describes the nature of the media in       -->
 <!--    a stream                          -->
 <!-- ######################################### -->
 <xs:simpleType name="handlerType">
     <xs:restriction base="xs:string">
         <xs:enumeration value="odsm"/>
```

```
            <xs:enumeration value="crsm"/>
            <xs:enumeration value="sdsm"/>
            <xs:enumeration value="vide"/>
            <xs:enumeration value="soun"/>
            <xs:enumeration value="m7sm"/>
            <xs:enumeration value="ocsm"/>
            <xs:enumeration value="ipsm"/>
            <xs:enumeration value="mjsm"/>
            <xs:enumeration value="hint"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```