INTERNATIONAL STANDARD

ISO/IEC 21000-7

First edition
2004-10-15

AMENDMENT 2
2007-07-01

Information technology — Multimedia framework (MPEG-21) —

Part 7: Digital Item Adaptation

AMENDMENT 2: Dynamic and Distributed Adaptation

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

Partie 7: Adaptation d'article numérique

AMENDEMENT 2: Adaptations dynamique et distribuée

Reference number
ISO/IEC 21000-7:2004/Amd.2:2007(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

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

Published in Switzerland

Foreword

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

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

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

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

Amendment 2 to ISO/IEC 21000-7:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Multimedia framework (MPEG-21) —

Part 7: **Digital Item Adaptation**

AMENDMENT 2: Dynamic and Distributed Adaptation

In subclause 1.4, replace the fourth list item with the following:

— *Bitstream Syntax Description* tools comprise the third major category of Digital Item Adaptation tools. A BSD describes the syntax – in most cases, the high level structure – of a binary media resource. Using such a description, a Digital Item resource adaptation engine can transform the bitstream and the corresponding description using editing-style operations such as data truncation and simple modifications. Streaming instructions enhance the BSD by defining a set of properties and attributes which describe the fragmentation, timing and random access point indication for the BSD and its described resource. They are used for streamed processing and transport, e.g., in dynamic and distributed adaptation scenarios.

In Clause 2, add the following text:

ISO/IEC 14977:1996, *Information technology — Syntactic metalanguage — Extended BNF*

ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding*

ISO/IEC 16262, *Information technology — ECMAScript language specification*

Document Object Model (DOM) Level 3 Core Specification, Version 1.0, W3C Recommendation 07 April 2004

Insert a new subclause 3.1.4 as follows:

3.1.4 Dynamic and Distributed Adaptation-specific terms and definitions

3.1.4.1
dynamic adaptation
adaptation of **Digital Item**s according to dynamically changing **usage environment**s

EXAMPLE The available bandwidth may drop during a streaming session and the Digital Item is consequently adapted to this new usage environment.

3.1.4.2
distributed adaptation
multiple adaptation steps successively performed on different ISO/IEC 21000 **peers**

EXAMPLE A same resource may be successively adapted on a server, network node and/or terminal.

3.1.4.3
process unit
well-formed fragment of a BSD or other XML metadata used for adaptation purposes, that can be consumed as such by the ISO/IEC 21000 **peer**, and to which a time information may be attached, indicating the point in time when it becomes available to the ISO/IEC 21000 **peer** for consumption

NOTE A process unit is a processing-oriented concept rather than a delivery-oriented concept. It does not depend on any encoding method used for delivering it.

3.1.4.4
XML fragmentation
authoring process by which an XML document is split into **process unit**s meaningful for consumption purposes

NOTE This process may attach time information to the output **process unit**s indicating the point in time when they become available to the ISO/IEC 21000 **peer** for consumption.

3.1.4.5
processing time
point in time when a **process unit** is available to the ISO/IEC 21000 **peer** for consumption

3.1.4.6
access unit
smallest segment of data that is atomic in time

EXAMPLE ISO/IEC 14496-1 access unit or ISO/IEC 15938-1 access unit (cf. TeM or BiM).

3.1.4.7
encoding
process by which a **process unit** is transformed into another representation for storage and/or delivery purposes

NOTE 1 The output of encoding is a sequence of one or more **access units**.

NOTE 2 Encoding does not necessarily imply compression (see for example TeM for XML).

3.1.4.8
random access point
point which indicates whether an **access unit** carries information that can be consumed by a ISO/IEC 21000 **peer** independent from any other **access units**

In subclause 3.2, append the following text at the end of this sublcause and order the resulting list alphabetically:

PU:	Process Unit
AU:	Access Unit
RAP:	Random Access Point
CTS:	Composition Time Stamp
DTS:	Decoding Time Stamp
PTS:	Processing Time Stamp

In subclause 3.4, append the following new rows to Table 1:

msi	urn:mpeg:mpeg21:2003:01-DIA-MSI-NS
si	urn:mpeg:mpeg21:2003:01-DIA-XSI-NS
ps	urn:mpeg:mpeg21:2003:01-DIA-PSS-NS
bs1x	urn:mpeg:mpeg21:2003:01-DIA-BSDL1x-NS
bs2x	urn:mpeg:mpeg21:2003:01-DIA-BSDL2x-NS

Insert a new subclause 4.4.4 as follows:

4.4.4 Classification scheme resolution mechanism

ISO/IEC 15938-5 specifies a descriptor named `mpeg7:ClassificationSchemeAliasType` for assigning an alias to a namespace. Additionally, this part of ISO/IEC 21000 specifies a more compact syntax for the same purpose by making use of the XML namespace binding context. The XML Namespaces recommendation [13] specified how to bind a prefix to a namespace. This namespace binding context is then inherited by the child elements of the element where the namespace is declared. This part of 21000 uses this namespace binding context to resolve the aliases used in classification schemes.

In case of conflicts between the namespace binding context and the binding specified by `mpeg7:ClassificationSchemeAliasType`, the namespace bound by the latter is used.

EXAMPLE 1 When a classification scheme term such as `:myAlias:myCSTerm`, the alias `myAlias` alias is resolved against the namespace binding context, i.e. the XML namespaces declaration.

EXAMPLE 2 Classification Scheme resolution mechanism using the XML namespace declaration. The alias `MV4` is resolved against the namespace binding context, i.e., the XML namespace declaration within the Description element which is inherited by its child elements (`xmlns:MV4="urn:mpeg:mpeg4:video:cs:syntacticalLabels"`).

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
<Description xsi:type="gbsd:gBSDType"
    xmlns:gbsd="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
    bs1:bitstreamURI="akiyo.mpg4"
    xmlns:MV4="urn:mpeg:mpeg4:video:cs:syntacticalLabels">

  <gBSDUnit syntacticalLabel=":MV4:VO" start="0" length="18" />
  <!-- ... and so on ... -->

 </Description>
</ DIA>
```

Remove subclause 5.2.3, Unsigned integer datatypes semantics.

In subclause 5.5.2, append the following text to the schema of this subclause:

```
<!-- ################################################ -->
<!-- Definition of ExternalVectorRefType                -->
<!-- ################################################ -->

<complexType name="ExternalVectorRefType" abstract="true">
   <complexContent>
      <extension base="dia:VectorDataType">
         <attribute name="uri" type="anyURI" use="required"/>
      </extension>
   </complexContent>
</complexType>

<!-- ############################################### -->
<!-- Definition of ExternalNMTokenVectorRefType        -->
<!-- ############################################### -->

<complexType name="ExternalNMTokenVectorRefType">
   <complexContent>
      <extension base="dia:ExternalVectorRefType"/>
   </complexContent>
</complexType>
```

In subclause 5.5.3, append the following text after the semantics of the `SemanticalDataRefType`*:*

Semantics of the `ExternalVectorRefType`:

`ExternalVectorRefType`	`ExternalVectorRefType` extends `VectorDataType` and provides a base abstract type for referencing external data by use of the XPointer framework.
`uri`	Describes the URI fragment identifier to be evaluated.

Semantics of the `ExternalNMTokenVectorRefType`:

`ExternalNMTokenVectorRefType`	Describes a vector of `xsd:NMTOKEN` values which must be returned by the URI fragment identifier evaluation.

In subclause 6.5.4.2, replace the text of the whole subclause with the following text:

```
<!-- ############################################### -->
<!--  Definition of CodecCapabilities                  -->
<!-- ############################################### -->

<complexType name="CodecCapabilitiesType">
  <complexContent>
    <extension base="dia:TerminalCapabilityBaseType">
      <sequence>
        <element name="Decoding" type="dia:CodecCapabilityBaseType"
           minOccurs="0" maxOccurs="unbounded"/>
        <element name="Encoding" type="dia:CodecCapabilityBaseType"
           minOccurs="0" maxOccurs="unbounded"/>
        <element name="DecodingEncoding" type="dia:CodecCapabilityBaseType"
           minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

In subclause 6.5.4.3, append a new row after the definition of the `Encoding` element:

`DecodingEncoding`	Describes the decoding and encoding capabilities of the terminal.

In subclause 6.5.12.2, add a new attribute to the `AudioOutputCapabilitiesType`:

```
    <attribute name="silenceSuppression" type="boolean" use="optional"/>
```

In subclause 6.5.12.3, append a new row after the definition of the `numChannels` attribute:

`silenceSuppression`	Describes whether silence suppression shall be applied for the audio. NOTE This feature is usually applied to stop the audio transfer through the network during periods when the participants in an audio conference are currently not talking. Furthermore, this feature is applied to decrease the used bandwidth in moments of silence.

Insert the following as a new subclause 6.5.29 and increment subsequent clauses accordingly:

6.5.29 RELCapabilities

6.5.29.1 Introduction

This subclause specifies the rights expression languages a terminal supports. A terminal can support none, one, or multiple rights expression languages (RELs). In the case of supporting multiple RELs, the terminal can also specify the preference level for each one of them.

6.5.29.2 RELCapabilities syntax

```
<!-- ################################################ -->
<!--  Definition of RELCapabilities                   -->
<!-- ################################################ -->

<complexType name="RELCapabilitiesType">
   <complexContent>
      <extension base="dia:TerminalCapabilityBaseType">
         <sequence>
            <element name="RightsLanguage" type="dia:RELCapabilityType"
               minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

6.5.29.3 RELCapabilities semantics

Semantics of the `RELCapabilitiesType`:

Name	*Definition*
`RELCapabilitiesType`	Tool for describing the rights expression language capabilities a terminal supports.
`RightsLanguage`	Describes a rights expression language supported by a terminal.

Insert the following as a new subclause 6.5.30 and increment subsequent clauses accordingly:

6.5.30 RELCapability

6.5.30.1 Introduction

This subclause specifies a rights language that a terminal supports.

6.5.30.2 RELCapability syntax

```
<!-- ################################################ -->
<!--  Definition of RELCapability                      -->
<!-- ################################################ -->

<complexType name="RELCapabilityType">
   <complexContent>
      <extension base="dia:DIABaseType">
         <sequence>
            <element name="Name" type="mpeg7:ControlledTermUseType"/>
         </sequence>
         <attribute name="preferenceLevel" type="positiveInteger"
            use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

6.5.30.3 RELCapability semantics

Semantics of the `RELCapabilityType`:

Name	*Definition*
`RELCapabilityType`	Tool for describing a rights expression language a terminal supports.
`Name`	Describes the name of a rights language. A classification scheme that may be used for this purpose is the `RightsLanguagesCS` defined in Annex A.2.16.
`preferenceLevel`	Describes the preference level of a rights language among the supported ones by a terminal. The value of one indicates the top preference level.

In subclause 6.5.29 (will be 6.5.31 after incrementing) append the following text:

EXAMPLE 5 The following example provides an instantiation of the RELCapabilities of the terminal. In this example, the terminal is able to interpret the following languages in decreased level preference: ISO/IEC 21000-5 Base profile and ISO/IEC 21000-5.

```
<DIA>
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironmentProperty xsi:type="TerminalsType">
      <Terminal>
        <TerminalCapability xsi:type="RELCapabilitiesType">
          <RightsLanguage xsi:type="RELCapabilityType" PreferenceLevel="1">
            <Name
               href="urn:mpeg:mpeg21:2003:01-DIA-RightsLanguagesCS-
NS:2003:RELProfile:Base">
              <mpeg7:Name xml:lang="en">MPEG-21 REL Base Profile</mpeg7:Name>
            </Name>
          </RightsLanguage>
          <RightsLanguage xsi:type="RELCapabilityType" PreferenceLevel="2">
            <Name
               href="urn:mpeg:mpeg7:2003:01-DIA-RightsLanguagesCS-NS:2003:REL">
              <mpeg7:Name xml:lang="en">MPEG-21 REL</mpeg7:Name>
            </Name>
          </RightsLanguage>
        </TerminalCapability>
      </Terminal>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

In subclause 6.6.4.2, add a new attribute to the `NetworkCapabilityType`:

```
    <attribute name="maxPacketSize" type="positiveInteger" use="optional"/>
```

In subclause 6.6.4.3, append a new row after the definition of the `errorCorrection` attribute:

maxPacketSize	Describes the maximum size of a packet in bit. NOTE This attribute is considered for technologies that support packet transport (e.g. IP networks).

In subclause 7.2.2, append the following text to the schema:

```
    <!-- ####################################################### -->
    <!-- Definition of the StreamingRefType                      -->
    <!-- ####################################################### -->

    <complexType name="StreamingRefType">
        <complexContent>
            <extension base="dia:ReferenceType">
                <attribute name="minPUSize" type="positiveInteger"
                    use="optional"/>
                <attribute name="avgPUSize" type="positiveInteger"
                    use="optional"/>
                <attribute name="maxPUSize" type="positiveInteger"
                    use="optional"/>
            <attribute name="minBitstreamPUSize" type="positiveInteger"
               use="optional"/>
            <attribute name="avgBitstreamPUSize" type="positiveInteger"
               use="optional"/>
            <attribute name="maxBitstreamPUSize" type="positiveInteger"
               use="optional"/>
         </extension>
      </complexContent>
   </complexType>
```

In subclause 7.2.3, append the following text at the end of this subclause:

Semantics of the `StreamingRefType`:

Name	*Definition*
`StreamingRefType`	Describes a reference to the (generic) Bitstream Syntax Description or AdaptationQoS description which can be used by a streaming processor. The target of this reference shall be of `bs1:BSDType` or `dia:AdaptationQoSType`. NOTE In case it is used by the `SteeringDescriptionRef` element the reference shall be of `dia:AdaptationQoSType`. In case it is used by the `BSDRef` element the reference shall be of `bs1:BSDType`.
`minPUSize`	Describes the minimum size of the process unit in bytes.
`avgPUSize`	Describes the average size of the process unit in bytes.
`maxPUSize`	Describes the maximum size of the process unit in bytes.
`minBitstreamPUSize`	Describes the minimum size of the bitstream fragment in bytes which is described by a single process unit. NOTE 1 This attribute shall appear only once, e.g., within a single BSDLink description.
`avgBitstreamPUSize`	Describes the average size of the bitstream fragment in bytes which is described by a single process unit. NOTE 2 This attribute shall appear only once, e.g., within a single BSDLink description.
`maxBitstreamPUSize`	Describes the maximum size of the bitstream fragment in bytes which is described by a single process unit. NOTE 3 This attribute shall appear only once, e.g., within a single BSDLink description.

NOTE 4 The information within the `StreamingRefType` may be used by the buffer of an adaptation engine in a streaming scenario – similar to a decoding buffer of a media resource player – which is related to the processing part of the delivery and not the delivery itself.

In subclause 7.2.4, append the following text at the end of this subclause:

EXAMPLE This example demonstrates the usage of the `dia:StreamingRefType` within a BSDLink description for an ISO/IEC 14496-3 bit sliced arithmetic coding (BSAC) bitstream.

```
<DIA>
  <Description xsi:type="BSDLinkType">
    <SteeringDescriptionRef xsi:type="StreamingRefType"
       uri="bsac_AQoS.xml"
       maxPUSize="52"/>
    <BSDRef xsi:type="StreamingRefType"
       uri="bsac_gBSD.xml"
       maxPUSize="132"
       minBitstreamPUSize="98"
       maxBitstreamPUSize="515"
       avgBitstreamPUSize="228"/>
    <BitstreamRef uri="bsac.raw"/>
    <BSDTransformationRef uri="bsac.xslt"
       type="http://www.w3.org/1999/XSL/Transform"/>
    <Parameter xsi:type="IOPinRefType" name="nlayers">
      <Value>LAYERS_OF_SCALABLE_AUDIO</Value>
    </Parameter>
  </Description>
</DIA>
```

In subclause 8.1.6, append a new paragraph at the end of this subclause:

Subclauses 8.5 and 8.6 specify media resource and XML streaming instructions required for adapting resources in streaming scenarios, in particular for dynamic and distributed adaptation scenarios. The media resource and XML streaming instructions define a set of properties and attributes which describe the fragmentation, timing, and additional information of a BSD and its corresponding bitstream. Furthermore, subclause 8.7 specifies a so-called properties stylesheet enabling to set these properties without modifying the XML document.

In subclause 8.2.1, append a new paragraph at the end of this subclause:

Optional features

This part of ISO/IEC 21000 specifies a set of optional features in Annex L organized in modules. A conformant BSDtoBin or BintoBSD parser is not required to implement these features. Alternatively, when it implements, it shall conform to their specification in this part of ISO/IEC 21000. These features are organized in three modules:

— Use of ISO/IEC 16262 (ECMAScript) for user-defined datatypes.

— Use of ISO/IEC 16262 (ECMAScript) for user-defined XPath functions.

— Support for array of variables.

NOTE A set of non-normative features of BSDL used for extensibility and for debugging is defined in Annex M.

Information about the BS Schema

A `bs1:schemaInformation` element contained at the root of the schema document wraps information about the BS Schema. This information is provided as attributes in BSDL-1 or BSDL-2 namespaces. BSDL-1 attributes are relevant to both BSDtoBin and BintoBSD processors whereas BSDL-2 attributes are only relevant to BintoBSD.

When a BSDL-1 attribute is provided both in the BSD and the bs1:schemaInformation element of the BS Schema with conflicting values, the value used in the BSD supersedes the one used in the BS Schema.

In subclause 8.3.1.1, replace Table 4 with the following table:

See Section	Datatype name	Encoding length
3.2.1	xsd:string	Indefinite
3.3.1	xsd:normalizedString	Indefinite
3.2.4	xsd:float	4
3.2.5	xsd:double	8
3.2.15	xsd:hexBinary	Indefinite
3.2.16	xsd:base64Binary	Indefinite
3.3.16	xsd:long	8
3.3.17	xsd:int	4
3.3.18	xsd:short	2
3.3.19	xsd:byte	1
3.3.21	xsd:unsignedLong	8
3.3.22	xsd:unsignedInt	4
3.3.23	xsd:unsignedShort	2
3.3.24	xsd:unsignedByte	1

In subclause 8.3.1.1, replace the following paragraph:

All integer types (xsd:long, xsd:int, xsd:short, xsd:byte) and their unsigned derivatives (xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte) are encoded in big endian. BSDL does not provide equivalent types for little endian, but the BS Schema author still has the possibility to specify a little endian value by decomposing it into several ordered bytes.

with:

All integer types (xsd:long, xsd:int, xsd:short, xsd:byte) and their unsigned derivatives (xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte) are encoded in big endian. BSDL provides equivalent types for little endian in subclause 8.3.1.4.

In subclause 8.3.1.1, replace the following paragraph:

xsd:normalizedString is coded as US-ASCII. Note that the xsd:string type is not supported by BSDL since it allows carriage return and line feed characters whose processing is platform-dependent.

with:

xsd:string and xsd:normalizedString are coded as US-ASCII. Note that care should be taken with the xsd:string type since it allows carriage return and line feed characters whose processing is platform-dependent. In case such control characters with platform-dependent encoding are present in the stream it is recommended to escape these characters in the BSD with a character reference as specified in XML such as "&#x0D;" or "&#13;" instead of the Carriage Return character.

In subclause 8.3.1.1, replace the following paragraph:

BSDL built-in datatypes

In addition to the XML Schema built-in datatypes listed above, BSDL provides two built-in datatypes named `bs1:byteRange` and `bs1:bitstreamSegment`. Their syntax is defined in the *Schema for BSDL-1 extensions*. They carry specific semantics in the context of bitstream generation explained in subclause 8.3.1.4 and have an indefinite length similarly to `xsd:hexBinary` and `xsd:base64Binary`.

with:

BSDL built-in datatypes

In addition to the XML Schema built-in datatypes listed above, BSDL provides two built-in datatypes pointing to segments of data within the described bitstream, a set of built-in integer datatypes supporting little-endian encoding, two built-in datatypes corresponding to the exponentional Golomb variable-length coding scheme, and string datatypes supporting UTF-8 and UTF-16 encoding, as well as strings terminated with a null character. Their syntax is defined in the *Schema for BSDL-1 extensions*. They carry specific semantics in the context of bitstream generation explained in subclause 8.3.1.4.

Additionaly, BSDL provides datatypes for unsigned integer values encoded from 1 to 32 bits. Their syntax is defined in a separate schema document with no target namespace provided in subclause 8.3.1.2 and their semantics is defined in subclause 8.3.1.4. This schema document is included by the *Schema for BSDL-1 extensions* and the datatype consequently inherit the BSDL-1 namespace.

Extension datatypes

In addition to XML Schema and BSDL built-in datatypes, BSDL provides the possibility to specify proprietary extension datatypes and use them in a BS Schema. An extension datatype is identified by a URI comprising a base part followed by a fragment identifier. The `bs1:codec` attribute allows a schema author to override a datatype defined in the schema. For this, the simple type needs to be derived by extension by declaring a `bs1:codec` attribute with a fixed or default value equal to the URI identifying the extension datatype. It is then possible to extend the implementation of BSDtoBin and BintoBSD processors to implement the decoding (i.e., reading the syntactical element from the bitstream and instantiating its lexical representation) and encoding (i.e., binary encoding the lexical value and writing it to the bitstream) of these datatypes. This extension mechanism is further specified in Annex K.

Properties in BS Description

For processing purpose, BSDL defines a set of abstract element properties that enrich the infoset of BS Description elements and are considered by the BSDtoBin processor. Each has an associated attribute. These properties are ineritable from the parent element. Properties may be specified by the associated attribute provided by the BS Description or the BS Schema (via its *default* or *fixed* value) or inherited from the parent element in the following way:

For each BS Decsription element in document order:

- if the associated attribute is present in the BS Description element, then the property is obtained from the value specified by the attribute; the way the property is computed from the attribute value is specific to the property,
- otherwise, if the BS Schema declares a default or fixed value for the associated attribute, then the property is obtained from this value,
- otherwise, if the element has a parent element, the property is inherited from the property of the parent element,
- otherwise (root element), it takes the default value for the document.