
**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)

AVA Documents
Document Sharing Hub
© ISO/IEC 2007

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.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2007

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

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

Published in Switzerland

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 Items** according to dynamically changing **usage environments**

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 units** meaningful for consumption purposes

NOTE This process may attach time information to the output **process units** 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 subclause 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` 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"/>
    </extension>
  </complexContent>
</complexType>
```

```

        <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
<code>StreamingRefType</code>	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 <code>bs1:BSDType</code> or <code>dia:AdaptationQoSType</code>. NOTE In case it is used by the <code>SteeringDescriptionRef</code> element the reference shall be of <code>dia:AdaptationQoSType</code>. In case it is used by the <code>BSDRef</code> element the reference shall be of <code>bs1:BSDType</code>.
<code>minPUSize</code>	Describes the minimum size of the process unit in bytes.
<code>avgPUSize</code>	Describes the average size of the process unit in bytes.
<code>maxPUSize</code>	Describes the maximum size of the process unit in bytes.
<code>minBitstreamPUSize</code>	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 <code>BSDLink</code> description.
<code>avgBitstreamPUSize</code>	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 <code>BSDLink</code> description.
<code>maxBitstreamPUSize</code>	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 <code>BSDLink</code> 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 `bsl: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	<code>xsd:string</code>	Indefinite
3.3.1	<code>xsd:normalizedString</code>	Indefinite
3.2.4	<code>xsd:float</code>	4
3.2.5	<code>xsd:double</code>	8
3.2.15	<code>xsd:hexBinary</code>	Indefinite
3.2.16	<code>xsd:base64Binary</code>	Indefinite
3.3.16	<code>xsd:long</code>	8
3.3.17	<code>xsd:int</code>	4
3.3.18	<code>xsd:short</code>	2
3.3.19	<code>xsd:byte</code>	1
3.3.21	<code>xsd:unsignedLong</code>	8
3.3.22	<code>xsd:unsignedInt</code>	4
3.3.23	<code>xsd:unsignedShort</code>	2
3.3.24	<code>xsd:unsignedByte</code>	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 "" or "" 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 exponential 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.

Additionally, 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 inheritable 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 Description 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.

The properties are the following:

- The *addressUnit* property is associated with the the `bs1:addressUnit` attribute. It indicates the address unit used by the `bs1:byteRange` and `bs1:bitstreamSegment` datatypes. Its value is *bit* or *byte*, the default value for the document is *byte*.
- The *ignore* property is associated with the the `bs1:ignore` attribute. Its value is a *true* or *false*. If set to *true*, BSDtoBin shall ignore the current element. Otherwise, process the element normally. The default value for the document is *false*.
- The *bitstreamURI* property indicates the absolute URI of the described bitstream and is associated to the `bs1:bitstreamURI` attribute. If the `bs1:bitstreamURI` attribute indicates a relative URI, then the *bitstreamURI* property of the element is obtained by resolving the `bs1:bitstreamURI` attribute value against the *bitstreamURI* property of the parent element. Otherwise (absolute value), the *bitstreamURI* property is set to the value of the `bs1:bitstreamURI` attribute. The default value for the document is the absolute URI of the BSD document. If a `bs1:bitstreamURI` attribute is specified for the BSD root element (either explicitly in the BSD or by default in the BS Schema) with a relative value, the resulting *bitstreamURI* property is therefore obtained by resolving the relative URI against the absolute BSD URI.
- The *insertEmPrevByte* property is associated with the `bs1:insertEmPrevByte` attribute. Its value is an even number of hexadecimal strings or *none*. If the `bs1:insertEmPrevByte` attribute is empty, then the property value is *none*. The default value for the document is *none*.

In subclause 8.3.1.2, replace the text of the whole subclause with the following text:

```
<?xml version="1.0"?>
<!-- Digital Item Adaptation ISO/IEC 21000-7/Amd.2 -->
<!-- Schema for BSDL-1 extensions -->
<schema
  version="ISO/IEC 21000-7:2004/Amd.2"
  id="BSDL-1-AMD2.xsd"
  targetNamespace="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS">

  <annotation>
    <documentation>
      Schema for BSDL-1 extensions
    </documentation>
  </annotation>

  <import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS" schemaLocation="DIA.xsd"/>
  <include schemaLocation="UnsignedIntegers-AMD2.xsd"/>

  <!-- ##### -->
  <!-- BitstreamSyntaxDescriptionType -->
  <!-- ##### -->

  <complexType name="BSDType" abstract="true">
    <complexContent>
      <extension base="dia:DIADescriptionType"/>
    </complexContent>
  </complexType>
```

```

<element name="schemaInformation">
  <complexType>
    <complexContent>
      <restriction base="anyType">
        <anyAttribute processContents="lax"/>
      </restriction>
    </complexContent>
  </complexType>
</element>

<!-- ##### -->
<!--          BSDL Attributes          -->
<!-- ##### -->

<attribute name="ignore" type="boolean"/>
<attribute name="bitstreamURI" type="anyURI"/>

<attribute name="addressUnit" type="bs1:unitsType"/>

<attribute name="codec" type="anyURI"/>
<attribute name="requiredExtensions" type="bs1:anyURIList"/>

<!-- Attribute specifying the insertion of emulation prevention bytes -->
<attribute name="insertEmPrevByte" type="bs1:hexBinaryStrings"/>

<!-- Address unit definition -->
<simpleType name="unitsType">
  <restriction base="NMTOKEN">
    <enumeration value="bit"/>
    <enumeration value="byte"/>
  </restriction>
</simpleType>

<!-- A list of hexBinary strings -->
<simpleType name="hexBinaryStrings">
  <list itemType="hexBinary"/>
</simpleType>

<!-- A list of anyURI -->
<simpleType name="anyURIList">
  <list itemType="anyURI"/>
</simpleType>

<!-- ##### -->
<!--          BSDL Built-in Datatype          -->
<!-- ##### -->

<simpleType name="byteRange">
  <restriction>
    <simpleType>
      <list itemType="nonNegativeInteger"/>
    </simpleType>
    <length value="2" fixed="true"/>
  </restriction>
</simpleType>

<complexType name="bitstreamSegment">
  <complexContent>
    <restriction base="anyType">
      <attribute name="start" type="nonNegativeInteger" use="optional"/>
      <attribute name="length" type="nonNegativeInteger" use="optional"/>
    </restriction>
  </complexContent>
</complexType>

```

```

<simpleType name="stringUTF16NT">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF16BENT">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF16LENT">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF8NT">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF16">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF16BE">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF16LE">
  <restriction base="string"/>
</simpleType>
<simpleType name="stringUTF8">
  <restriction base="string"/>
</simpleType>
<simpleType name="longLE">
  <restriction base="long"/>
</simpleType>
<simpleType name="intLE">
  <restriction base="int"/>
</simpleType>
<simpleType name="shortLE">
  <restriction base="short"/>
</simpleType>
<simpleType name="unsignedLongLE">
  <restriction base="unsignedLong"/>
</simpleType>
<simpleType name="unsignedIntLE">
  <restriction base="unsignedInt"/>
</simpleType>
<simpleType name="unsignedShortLE">
  <restriction base="unsignedShort"/>
</simpleType>
<simpleType name="align32">
  <restriction base="hexBinary">
    <length value="4"/>
  </restriction>
</simpleType>
<simpleType name="align16">
  <restriction base="hexBinary">
    <length value="2"/>
  </restriction>
</simpleType>
<simpleType name="align8">
  <restriction base="hexBinary">
    <length value="1"/>
  </restriction>
</simpleType>
<simpleType name="unsignedExpGolomb">
  <restriction base="unsignedInt"/>
</simpleType>

```

```

<simpleType name="signedExpGolomb">
  <restriction base="int"/>
</simpleType>

</schema>

```

The following schema document specifies a library of unsigned integer datatypes encoded on 1 to 32 bits. It is defined with no target namespace and is included by the Schema for BSD-1 extensions. Consequently, all the datatypes inherit the BSD-1 namespace.

NOTE Since this schema document has no target namespace and is included by the Schema for BSD-1 extensions, the base types must be qualified with a prefix bound to the XML Schema namespace in order to avoid namespace ambiguity.

```

<?xml version="1.0"?>
<!-- Digital Item Adaptation ISO/IEC 21000-7:2004/Amd.2 -->
<!-- Unsigned Integer Datatypes Schema -->
<schema
  version="ISO/IEC 21000-7:2004/Amd.2"
  id="UnsignedIntegers-AMD2.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <simpleType name="b32">
    <restriction base="xsd:unsignedInt"/>
  </simpleType>
  <simpleType name="b31">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="2147483648"/>
    </restriction>
  </simpleType>
  <simpleType name="b30">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="1073741824"/>
    </restriction>
  </simpleType>
  <simpleType name="b29">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="536870912"/>
    </restriction>
  </simpleType>
  <simpleType name="b28">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="268435456"/>
    </restriction>
  </simpleType>
  <simpleType name="b27">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="134217728"/>
    </restriction>
  </simpleType>
  <simpleType name="b26">
    <restriction base="xsd:unsignedInt">
      <maxExclusive value="67108864"/>
    </restriction>
  </simpleType>

```

```

<simpleType name="b25">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="33554432"/>
  </restriction>
</simpleType>
<simpleType name="b24">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="16777216"/>
  </restriction>
</simpleType>
<simpleType name="b23">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="8388608"/>
  </restriction>
</simpleType>
<simpleType name="b22">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="4194304"/>
  </restriction>
</simpleType>
<simpleType name="b21">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="2097152"/>
  </restriction>
</simpleType>
<simpleType name="b20">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="1048576"/>
  </restriction>
</simpleType>
<simpleType name="b19">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="524288"/>
  </restriction>
</simpleType>
<simpleType name="b18">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="262144"/>
  </restriction>
</simpleType>
<simpleType name="b17">
  <restriction base="xsd:unsignedInt">
    <maxExclusive value="131072"/>
  </restriction>
</simpleType>
<simpleType name="b16">
  <restriction base="xsd:unsignedShort"/>
</simpleType>
<simpleType name="b15">
  <restriction base="xsd:unsignedShort">
    <maxExclusive value="32768"/>
  </restriction>
</simpleType>
<simpleType name="b14">
  <restriction base="xsd:unsignedShort">
    <maxExclusive value="16384"/>
  </restriction>
</simpleType>
<simpleType name="b13">
  <restriction base="xsd:unsignedShort">

```

```

        <maxExclusive value="8192"/>
    </restriction>
</simpleType>
<simpleType name="b12">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="4096"/>
    </restriction>
</simpleType>
<simpleType name="b11">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="2048"/>
    </restriction>
</simpleType>
<simpleType name="b10">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="1024"/>
    </restriction>
</simpleType>
<simpleType name="b9">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="512"/>
    </restriction>
</simpleType>
<simpleType name="b8">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="256"/>
    </restriction>
</simpleType>
<simpleType name="b7">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="128"/>
    </restriction>
</simpleType>
<simpleType name="b6">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="64"/>
    </restriction>
</simpleType>
<simpleType name="b5">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="32"/>
    </restriction>
</simpleType>
<simpleType name="b4">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="16"/>
    </restriction>
</simpleType>
<simpleType name="b3">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="8"/>
    </restriction>
</simpleType>
<simpleType name="b2">
    <restriction base="xsd:unsignedShort">
        <maxExclusive value="4"/>
    </restriction>
</simpleType>

```

```

<simpleType name="b1">
  <restriction base="xsd:unsignedShort">
    <maxExclusive value="2"/>
  </restriction>
</simpleType>
</schema>

```

In subclause 8.3.1.3, append the following new rows:

schemaInformation	Element to be included in <code>xsd:schema</code> component via the <code>xsd:annotation/xsd:appinfo</code> combination. It is meant to contain attributes providing information on the BS Schema.
addressUnit	Specifies the addressing unit (bit or byte) used by <code>bs1:byteRange</code> and <code>bs1:bitstreamSegment</code> datatypes.
codec	Specifies the extension datatype. This attribute may be used either in the BS Description or declared in the BS Schema with a default or fixed value. Its value is the URI identifying the extension datatype.
insertEmPrevByte	Specifies the insertion of emulation prevention bytes. It contains an even number of hexadecimal strings. For each pair of hexadecimal strings, the first string specifies the byte sequence that requires the insertion of emulation prevention bytes and the second string specifies the byte sequence that should be written to the output bitstream instead. When the attribute is empty, no emulation prevention byte is inserted.
requiredExtensions	The <code>bs1:requiredExtensions</code> attribute indicates the user-defined datatypes required by this schema document. Its value is a list of URIs. Each URI indicates either a single datatype or a library of datatypes. In the first case, it contains a fragment identifier pointing to datatype name itself. In the second case, it identifies the library. This attribute shall be used in the BSD or in the BS Schema within the <code>bs1:schemaInformation</code> element. When provided both in the BSD and the <code>bs1:schemaInformation</code> element of the BS Schema with conflicting values, the value used in the BSD supersedes the one used in the BS Schema.

In subclause 8.3.1.4, replace the semantics of `byteRange` and `bitstreamSegment` by the following new text:

Name	Definition
byteRange	BSDL built-in datatype indicating the byte range of the resource identified by the bitstream URI of the current element. It consists in a list of two non-negative integers. The first integer parameter indicates the offset of the relevant range of data and the second parameter indicates its length. These values are specified in bits or bytes according to the value of the <code>addressUnit</code> property. The offset is relative to the start of the bitstream.

Name	Definition
bitstreamSegment	BSDL built-in complex type indicating a bitstream segment. The <code>start</code> and <code>length</code> attributes respectively indicate the offset of the segment and its length. These values are specified in bits or bytes according to the value of the <code>addressUnit</code> property. The offset is relative to the start of the bitstream. The BSDtoBin parser copies the relevant data segment only if the current element has no child elements. This built-in datatype is provided for compatibility with gBS Schema (see subclause 8.3.2) and shall not be used for BS Description generation (BSDL-2).

In subclause 8.3.1.4, append the following new rows:

longLE, intLE, shortLE, unsignedLongLE, unsignedIntLE, unsignedShortLE,	Integer datatypes with the same lexical value as their XML Schema counterpart (respectively <code>xsd:long</code>, <code>xsd:int</code>, <code>xsd:short</code>, <code>xsd:unsignedLong</code>, <code>xsd:unsignedInt</code> and <code>xsd:unsignedShort</code>), but with little-endian encoding.
stringUTF8, stringUTF16BE, stringUTF16LE, stringUTF16	String datatypes with UTF-8 and UTF-16 character encoding. UTF8 refers to the eight-bit UCS Transformation Format, UTF16BE refers to the sixteen-bit UCS Transformation Format, big-endian byte order, UTF16LE refers to the sixteen-bit UCS Transformation Format, little-endian byte order, and UTF16 refers to the sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark.
stringUTF8NT, stringUTF16BENT, stringUTF16LENT, stringUTF16NT	The same semantics apply as for <code>bs1:stringUTF8</code>, <code>bs1:stringUTF16BE</code>, <code>bs1:stringUTF16LE</code>, and <code>bs1:stringUTF16</code>, but with a terminating null character. Note that the terminating null character is not allowed in XML and hence does not appear in the lexical value. However BintoBSD must read it from the input bitstream, and BSDtoBin and gBSDtoBin must generate it into the output bitstream.
align8, align16, align32	Datatypes enabling reading a value from the input bitstream and writing padding bits to the output bitstream until it is aligned on a 1 byte (for <code>align8</code>), 2 byte (for <code>align16</code>) or 4 byte (for <code>align32</code>) word. BintoBSD reads bits from the input bitstream until it is aligned on respectively a 1 byte, 2 byte or 4 byte word and instantiates an empty element. When the bitstream is already correctly aligned, BintoBSD does not read any bit but still instantiates an empty element. BSDtoBin considers the element value in the PSVI, namely the lexical value indicated in the BS Description element if present, the fixed default specified in the BS Schema otherwise. If the BS Description element is empty and no fixed or default value is specified by the BS Schema, then BSDtoBin uses 0 as default value. BSDtoBin then encodes the obtained value, starting from the most significant bit until the output bitstream is adequately aligned. If the bitstream is already correctly aligned, BSDtoBin does not write any bit.

unsignedExpGolomb	BSDL built-in datatype corresponding to the unsigned exp-golomb encoding specified in ISO/IEC 14496-10.
signedExpGolomb	BSDL built-in datatype corresponding to the signed exp-golomb encoding specified in ISO/IEC 14496-10.
b1	A 1-bit unsigned integer value.
b2	A 2-bit unsigned integer value.
b3	A 3-bit unsigned integer value.
b4	A 4-bit unsigned integer value.
b5	A 5-bit unsigned integer value.
b6	A 6-bit unsigned integer value.
b7	A 7-bit unsigned integer value.
b8	A 8-bit unsigned integer value.
b9	A 9-bit unsigned integer value.
b10	A 10-bit unsigned integer value.
b11	A 11-bit unsigned integer value.
b12	A 12-bit unsigned integer value.
b13	A 13-bit unsigned integer value.
b14	A 14-bit unsigned integer value.
b15	A 15-bit unsigned integer value.
b16	A 16-bit unsigned integer value.
b17	A 17-bit unsigned integer value.
b18	A 18-bit unsigned integer value.
b19	A 19-bit unsigned integer value.
b20	A 20-bit unsigned integer value.
b21	A 21-bit unsigned integer value.
b22	A 22-bit unsigned integer value.
b23	A 23-bit unsigned integer value.
b24	A 24-bit unsigned integer value.
b25	A 25-bit unsigned integer value.

b26	A 26-bit unsigned integer value.
b27	A 27-bit unsigned integer value.
b28	A 28-bit unsigned integer value.
b29	A 29-bit unsigned integer value.
b30	A 30-bit unsigned integer value.
b31	A 31-bit unsigned integer value.
b32	A 32-bit unsigned integer value.

In subclause 8.3.1.6, append the following lines to the first paragraph:

Furthermore, when an element of the description is empty and its declaration in the schema specifies a value constraint (fixed or default attribute), the BSDtoBin processes this default value.

In subclause 8.3.2.2, replace the text of the whole subclause with following text.

```
<?xml version="1.0"?>
<!-- Digital Item Adaptation ISO/IEC 21000-7/Amd.2 -->
<!-- generic Bitstream Syntax Schema (gBSSchema) -->
<schema
  version="ISO/IEC 21000-7:2004/Amd.2"
  id="gBSSchema-AMD2.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:gbsd="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
  targetNamespace="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <annotation>
    <documentation>
      Definition of generic Bitstream Syntax Schema (gBSSchema)
    </documentation>
  </annotation>

  <import namespace="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
    schemaLocation="BSDL-1-AMD2.xsd"/>
  <include schemaLocation="UnsignedIntegers.xsd"/>

  <element name="gBSDUnit" type="gbsd:gBSDUnitType"/>

  <!-- ##### -->
  <!-- gBSDType definition -->
  <!-- ##### -->

  <complexType name="gBSDType">
    <complexContent>
      <extension base="bs1:BSDType">
        <sequence>
          <element ref="gbsd:gBSDUnit" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

```

        <attributeGroup ref="gbsd:addressAttributes"/>
        <anyAttribute namespace="##other" processContents="lax"/>
    </extension>
</complexContent>
</complexType>

<!-- ##### -->
<!-- Attribute group definition -->
<!-- ##### -->

<attributeGroup name="addressAttributes">
    <attribute name="addressMode" type="gbsd:addressModeType" use="optional"/>
    <attribute name="addressUnit" type="gbsd:unitsType" use="optional"/>
    <attribute ref="bs1:bitstreamURI" use="optional"/>
</attributeGroup>

<!-- ##### -->
<!-- gBSDUnit definition -->
<!-- ##### -->

<complexType name="gBSDUnitType">
    <complexContent>
        <extension base="bs1:bitstreamSegment">
            <choice minOccurs="0" maxOccurs="unbounded">
                <element name="Parameter" type="gbsd:paramType"/>
                <element ref="gbsd:gBSDUnit"/>
            </choice>
            <attribute name="syntacticalLabel" type="gbsd:labelType" use="optional"/>
            <attribute name="marker" type="gbsd:markerType" use="optional"/>
            <attributeGroup ref="gbsd:addressAttributes"/>
            <anyAttribute namespace="##other" processContents="lax"/>
        </extension>
    </complexContent>
</complexType>

<!-- Resource address unit definition -->
<simpleType name="unitsType">
    <restriction base="NMTOKEN">
        <enumeration value="bit"/>
        <enumeration value="byte"/>
    </restriction>
</simpleType>

<!-- Address mode definition -->
<simpleType name="addressModeType">
    <restriction base="NMTOKEN">
        <enumeration value="Absolute"/>
        <enumeration value="Consecutive"/>
        <enumeration value="Offset"/>
    </restriction>
</simpleType>

<!-- ***** Marker definition ***** -->
<simpleType name="markerType">
    <list itemType="NMTOKEN"/>
</simpleType>

```

```

<!-- ***** Label definition ***** -->
<simpleType name="labelType">
  <union>
    <simpleType>
      <restriction base="NMTOKEN">
        <whiteSpace value="collapse"/>
        <pattern value=":[^:]+:[^:]+"/>
      </restriction>
    </simpleType>
    <simpleType>
      <restriction base="anyURI"/>
    </simpleType>
  </union>
</simpleType>

<!-- ##### -->
<!--      Parameter definition      -->
<!-- ##### -->

<complexType name="paramType">
  <complexContent>
    <extension base="bs1:bitstreamSegment">
      <sequence>
        <element name="Value" type="anySimpleType"/>
      </sequence>
      <attribute name="name" type="gbsd:labelType" use="optional"/>
      <attribute name="marker" type="gbsd:markerType" use="optional"/>
      <attributeGroup ref="gbsd:addressAttributes"/>
      <anyAttribute namespace="##other" processContents="lax"/>
    </extension>
  </complexContent>
</complexType>

</schema>

```

In subclause 8.3.2.4, bullet 3) i), replace bullet II) with following text:

II) Consecutive: The length attribute is mandatory whereas the start attribute is optional. If the start attribute is not present, the segment represented by this gBSDUnit starts at the next unit (bit or byte) after the end of the preceding-sibling element or at the same location than its parent element in case the element is the first child. The length of the segment is given by the value of the length attribute. If a start attribute is present, its value (n) provides the offset from the preceding-sibling or parent element as follows: the beginning of the data represented by this gBSDUnit element can be located n “units” (bit or byte) after the end of the previous preceding-sibling element or after the start of the parent element in case it is the first child element.

In subclause 8.3.2.4, bullet 3) ii), replace

In the case when the length attribute of the Parameter element specifies a length (L1) longer than the length (L2) implied by the datatype specified by the xsi:type attribute of the Value element,

with:

In the case when the length attribute of the Parameter element is provided and specifies a length (L1) longer than the length (L2) implied by the datatype specified by the xsi:type attribute of the Value element,

In subclause 8.3.2.5, append the following text:

EXAMPLE 5 The following example illustrates the usage of the `bs1:datatype` attribute. It shows an excerpt of EXAMPLE 2 with the only difference that the datatype is defined by URI referring to the respective datatype as defined within the XML Schema definition. Thus, any datatype can be used as long as it can be referenced by URI, e.g., by using the namespace and XPointer's shorthand pointer.

```
<!--... and so on ...-->
<Parameter name="J2K:LMarker" length="2">
  <Value bs1:codec="http://www.w3.org/2001/XMLSchema#unsignedShort">47</Value>
</Parameter>
<Parameter name="J2K:Rsiz" length="2">
  <Value bs1:codec="http://www.w3.org/2001/XMLSchema#unsignedShort">0</Value>
</Parameter>
<Parameter name="J2K:Xsiz" length="4">
  <Value bs1:codec="http://www.w3.org/2001/XMLSchema#unsignedInt">768</Value>
</Parameter>
<Parameter name="J2K:Ysiz" length="4">
  <Value bs1:codec="http://www.w3.org/2001/XMLSchema#unsignedInt">512</Value>
</Parameter>
<!--... and so on ...-->
```

In subclause 8.4.1, replace the following text:

XPath expressions are evaluated relatively to a reference node, called *context node*. BSDL uses variables in the form of XPath expression for three cases:

- Number of occurrences of a particle (`bs2:nOccurs` attribute). In this case, the evaluation of the XPath expression should yield a *number* casted as an integer and the context node is the parent element of the current particle.
- Conditional occurrence of a particle (`bs2:if` attribute). In this case, the evaluation of the XPath expression should yield a *boolean* and the context node is the parent element of the current particle.
- Value of the `bs2:length` facet. In this case, the evaluation of the XPath expression should yield a *number* casted as an integer and the context node is the element, the type of which is constrained by the current facet.

with:

XPath expressions are evaluated relatively to a reference node, called *context node*. BSDL uses variables in the form of XPath expression for several cases:

- Number of occurrences of a particle (`bs2:nOccurs` attribute). In this case, the evaluation of the XPath expression should yield a *number* casted as an integer and the context node is the parent element of the current particle.
- Conditional occurrence of a particle (`bs2:if` attribute). In this case, the evaluation of the XPath expression should yield a *boolean* and the context node is the parent element of the current particle.
- Value of the `bs2:length` and `bs2:bitLength` facets. In this case, the evaluation of the XPath expression should yield a *number* casted as an integer and the context node is the element, the type of which is constrained by the current facet.
- Value of the `bs2:layerLength` attribute. In this case, the evaluation of the XPath expression should yield a *number* casted as an integer and the context node is the current element, i.e. the element whose `complexType` definition is constrained by the `bs2:layerLength` attribute.

- Value of `bs2:startContext`, `bs2:redefineMarker`, `bs2:stopContext` attributes. In this case, the evaluation of the XPath expression should yield a string. The context node is the current element.

In subclause 8.4.1, append the following text at the end of the section entitled Variables and before the section entitled Facets:

BSDL specifies the `bs2:log2()` function in addition to core XPath function library as follows: this function returns the logarithm in base 2 of the input argument. The input argument and output value are XPath *numbers*. This function is defined in the BSDL-2 namespace and must be used with a prefix bound to the BSDL-2 namespace. The namespace binding context used to resolve the prefixes used in an XPath expressions is the context of the element where this expression is used in the BS Schema.

XPath variables

BSDL supports the use of XPath variables within an XPath expression. Therefore, this part of ISO/IEC 21000 specifies three ways of assigning a value to an XPath variable.

- Firstly, the `bs2:parameter` element allows declaring a constant value for a variable.
- Secondly, the `bs2:assignPost` attribute assigns the value of a simple content element to a variable.
- Thirdly, the `bs2:assignPre` attribute assigns one or several values read directly upstream to variables.

In subclause 8.4.1, replace the following text:

Facets

BSDL-2 introduces a set of new facets to specify constraints on BSDL and XML Schema datatypes. Since XML Schema does not allow a user to add his/her own facets, they are declared in the BSDL-2 namespace and added to the `xsd:restriction` component via the annotation mechanism, i.e., the `xsd:annotation/xsd:appinfo` combination.

The list of BSDL-2 facets consists of `bs2:length`, `bs2:startCode` and `bs2:endCode`. They are used to constrain `xsd:hexBinary`, `xsd:base64Binary` and `bs1:byteRange` built-in datatypes. When applied to `bs1:byteRange`, note that the `xsd:length` facet does not indicate the number of bytes to read from the bitstream, but the number of integers in the datatype, namely always two, corresponding to the start and length information. This is why the *Schema for BSDL-1 extensions* forbids the use of `xsd:length` for further restricting `bs1:byteRange`. Alternatively, the `bs2:length` facet, which is ignored by XML Schema, can and shall be used with `bs1:byteRange` to indicate the length in bytes of the data segment to read.

Note that if a datatype with indefinite length (`xsd:hexBinary`, `xsd:base64Binary` and `bs1:byteRange`) is not constrained by either an `xsd:length`, `bs2:length`, `bs2:startCode` or `bs2:endCode` facet, then the bitstream is parsed until the end of file.

with:

BSDL-2 introduces a set of new facets to specify constraints on BSDL and XML Schema datatypes. Since XML Schema does not allow a user to add his/her own facets, they are declared in the BSDL-2 namespace and added to the `xsd:restriction` component via the annotation mechanism, i.e., the `xsd:annotation/xsd:appinfo` combination.

The list of BSDL-2 facets consists of `bs2:length`, `bs2:startCode`, `bs2:endCode`, `bs2:bitLength`, `bs2:escape` and `bs2:cdata`.