INTERNATIONAL STANDARD

ISO/IEC 15938-1

First edition
2002-07-01

AMENDMENT 1
2005-02-15

Information technology — Multimedia content description interface —

Part 1: **Systems**

AMENDMENT 1: Systems extensions

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

Partie 1: Systèmes

AMENDEMENT 1: Extensions des systèmes

Reference number
ISO/IEC 15938-1:2002/Amd.1:2005(E)

Documents
Document Sharing Hub

© ISO/IEC 2005

Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

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/IEC 2005

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

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

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 1 to ISO/IEC 15938-1:2002 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 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

Introduction

This document specifies the first Amendment to the Systems (MDS) part of ISO/IEC 15938.

Note: this document preserves the sectioning of ISO/IEC 15938-1 Systems. The text and figures given below are additions and/or modifications to those corresponding sections in ISO/IEC 15938-1. All figures and tables shall be renumbered due to the addition of several figures and tables.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

Information technology — Multimedia content description interface —

Part 1:
Systems

AMENDMENT 1: Systems extensions

Add the following reference in Clause 2:

- *RFC 1950, ZLIB Compressed Data Format Specification version 3.3.*

Add the following definition at the end of subclause 3.1.2.2.5:

ReservedBitsZero: a binary syntax element whose length is indicated in the syntax table. The value of each bit of this element shall be “0”. These bits may be used in the future for ISO/IEC defined extensions.

Add the following definitions to subclause 3.2 (keep alphabetical order):

initial schema
The schema that is known by the decoder before the decoding process starts.

additional schema
A schema that can be updated after the start of the decoding process.

schema update unit
Information in an access unit, conveying a schema or a portion thereof. Schema update units provide the means to modify the current decoder schema knowledge.

description fragment reference
A reference to a description fragment.

Note - For instance, a fragment reference can be a URI which serves to locate the fragment on the world wide web.

fragment reference
short term for description fragment reference.

fragment reference resolver
An entity that is capable of resolving the fragment reference provided in the fragment update payload.

fragment reference marker
A specific information used to describe a deferred fragment reference, which is present within the current description tree. It consists of a fragment reference, the name and type of the top most element of the referenced fragment.

fragment reference format
An encoding format of fragment references.

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

deferred fragment reference
A fragment reference that can be resolved at any time by the application using the terminal.

non-deferred fragment reference
A fragment reference that shall be resolved by the terminal at the composition time of the access unit containing the fragment reference.

optimised decoder
A decoder associated to a type and dedicated to certain encoding methods better suited than the generic ones.

type codec
Synonym to optimised decoder.

fixed optimised decoder
An optimised decoder used to decode either a complex type or a simple type. Fixed optimised decoders are set up at decoder initialisation phase and their mapping to types can't be modified during binary description stream lifetime.

advanced optimised decoder
An optimised decoder used to decode a simple type. Advanced optimised decoders parameters and their mappings to types can be modified during binary description stream lifetime.

advanced optimised decoder instance
An advanced optimised decoder initialised and ready to be used for the decoding of some data types.

Note - There can be several instances of the same advanced optimised decoder with different or identical parameters.

advanced optimised decoder type
The type, identified by a URI, of an advanced optimised decoder.

advanced optimised decoder instances table
A table of all the advanced optimised decoders available at a certain instant in time.

contextual optimised decoder
An optimised decoder which behavior is dependent on the current context of the decoding.

Note - For instance, the ZLib optimised decoder (see Clause 9) is a contextual optimised decoder.

Note - Upon certain events, the context must be reset. Upon a certain command or events they are flushed to release their contents. Only contextual optimised decoders are flushable.

advanced optimised decoder parameters
The parameters of an advanced optimised decoder.

contextual optimised decoder reset
An operation that resets the optimised decoder to put it in a defined initial state. All contextual information is discarded.

skippable subtree
A subtree of an XML document that the decoder is permitted not to decode.

optimised decoder mapping
An association between a type and a set of optimised decoders.

Renumber all definitions in subclause 3.2.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

Add the following abbreviations to the table in subclause 4.1:

MSB	Most Significant Bit
SU	Schema Update
SUU	Schema Update Unit

Add the following mnemonic to subclause 4.3:

Name	*Definition*
vlurmsbf5	Variable length code unsigned rational number, most significant bit first. The first n bits (Ext) which are '1' except of the nth bit which is '0', indicate that the rational number R in the interval 0≤R<1 is encoded by n times 4 bits. The ith bit of the n times 4 bits representing the rational number corresponds to a value of 2^-i. Thus the (n+1)st bit of the vlurmsbf5 code word (which corresponds to the MSB of the rational number) represents a value of ½, the (n+2)nd bit of the vlurmsbf5 code word represents a value of ¼., and so forth. An example for this type is shown in Figure AMD1-1. Note - Comparing two rational numbers A and B represented by a vlurmsbf5 code word can be done by comparing bit by bit the rational numbers starting from their respective MSBs. Then the rational number A is bigger if there is a '1' bit at a position at which there is a '0' for B. A is also bigger if there is a '1' bit at a position which is not present for B and when A is longer than B.

In subclause 4.3, add the following figure after Figure 2:

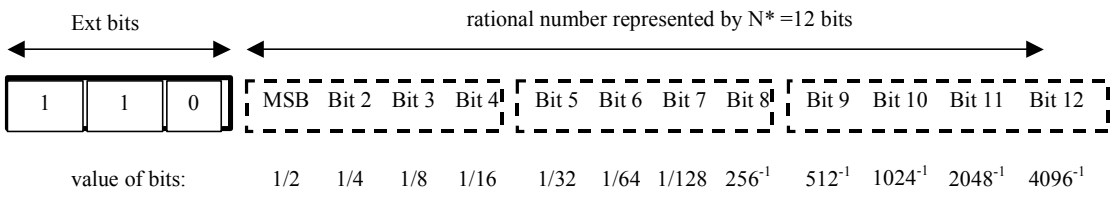


Figure AMD1-1 - Informative example for the vlurmsbf5 data type

Remove the '(informative)' in the title of subclause 5.2.3.

In subclause 5.2.3, replace this paragraph:

The BiM, in order to gain its compression efficiency, relies on a schema analysis phase. During this phase, internal tables are computed to associate binary code to XML elements, types and attributes.This principle mandates the full knowledge of the same schema by the decoder and the encoder for maximum interoperability.

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

by

The BiM, in order to gain its compression efficiency, relies on a schema analysis phase. During this phase, internal tables are computed to associate binary code to schema components (XML elements, types and attributes). BiM defines two methods to address schema components.

The first method allows the decoder to resolve a schema, possibly including schema components originating from several namespaces, at initialization phase. This set of schema components form the *initial schema.* In this schema, all type and substitution codes are merged together no matter the namespace they belong to. This results in shorter codes in the binary description stream. The initial schema can't be updated and is considered fixed for the binary description stream lifetime. It contains by default and at minimal the type codes of the xml schema types: anyType, anySimpleType, and all xml schema simple types. In this specification, anySimpleType is considered as a subtype of anyType.

The second method allows the decoder both to resolve a schema at initialization phase (the *initial schema*) and to receive updated schema information called *additional schemas*. Additional schemas differ from the initial schema as the codes of their schema components defined in different namespaces are defined in different code spaces. This results in larger code size but has the required flexibility for late updating. For full flexibility it is also possible to receive exclusively additional schemas and thus to operate without initial schema.

Both initial schemas and additional schemas are part of a unique table in which each entry identifies a specific schema. The first entries identify schemas that are part of the initial schema. The following ones identify additional schemas.

To further improve compression, BiM allows the association of specific codecs to specific data types instead of using the generic mechanisms defined in Clause 8. These encoding schemes can be optimised with the full knowledge of the properties of that data type.

In subclause 5.2.3, replace this paragraph:

As with the textual decoder, the resulting current description tree may be topologically equivalent to the original description if desired by the encoder, but it may also exhibit dynamic characteristics such that certain parts of the description are present at the decoder only at chosen times, are never present at all, can be acquired on application demand, or appear in a different part of the tree.

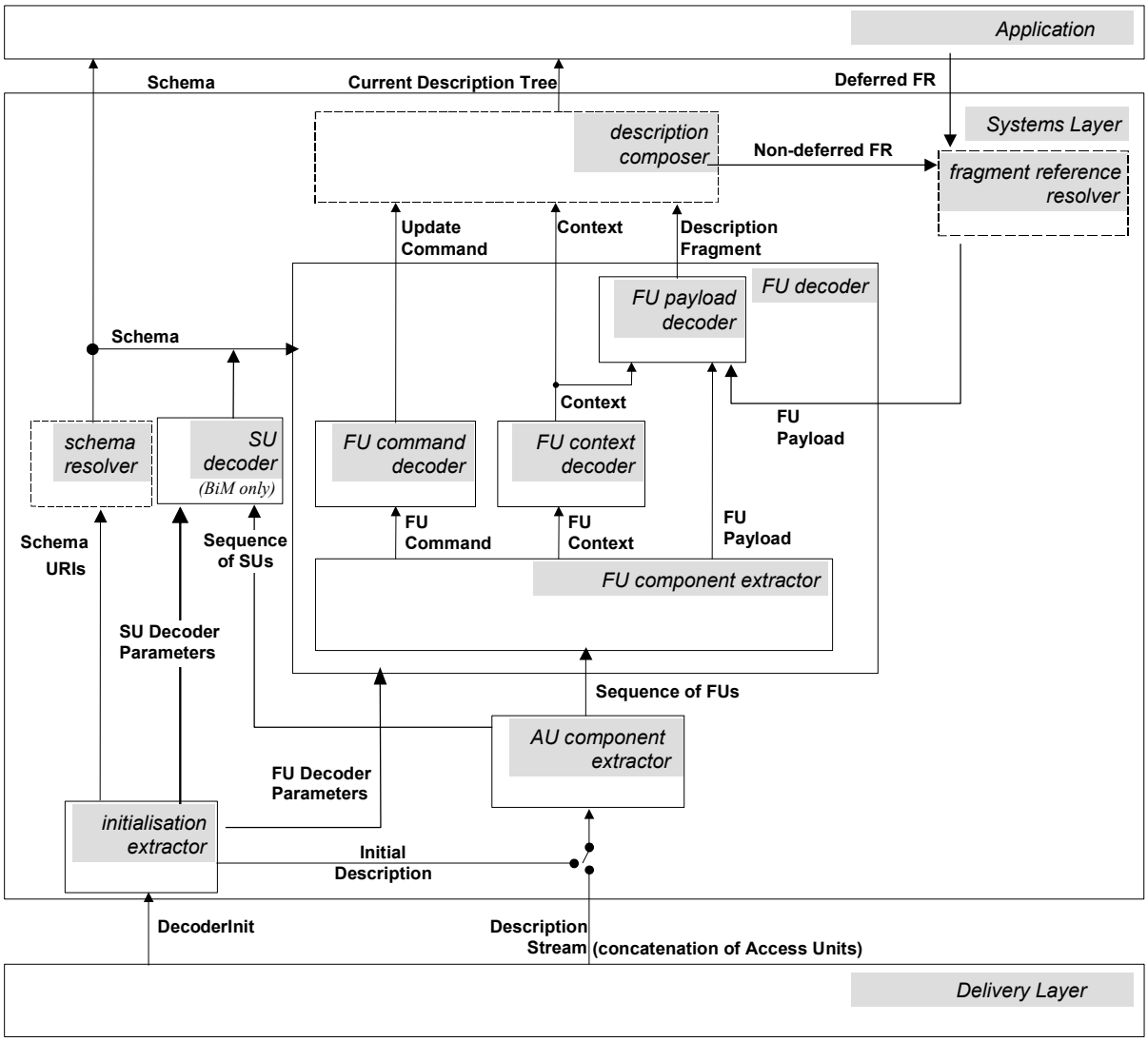
by

As with the textual decoder, the resulting current description tree may be topologically equivalent to the original description if desired by the encoder, but it may also exhibit dynamic characteristics such that certain parts of the description are present at the decoder only at chosen times, are never present at all, can be acquired on application demand, or appear in a different part of the tree.

Note - The schema update capabilities provided by the BiM framework defined in this specification aims at upgrading BiM decoder. It is not a mean to transmit an XML schema as is. To do so, one should use the W3C schema of schema to encode its schema.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

In subclause 5.3, replace the "Figure 4 - Terminal Architecture" by the following figure:



In subclause 5.3, second paragraph, replace:

…(FU Decoder Parameters, in Figure …

by

…(FU Decoder Parameters and SU Decoder Parameters, in Figure …

In subclause 5.4, add the following text after the first paragraph:

In the case of BiM, an access unit is composed of any number of schema update units followed by any number of fragment update units which are extracted by the access unit component extractor.

A schema update unit carries parts of an additional schema and is composed of

— a namespace identifier,

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

— a set of code tables to represent global elements, global types and global attributes,

— a binary encoded schema carrying the schema components definitions.

The full schema is not always necessary for the decoding of a particular binary description stream. To avoid unnecessary transmission, a schema update unit may contain only the definitions that are required for the decoding of the bitstream. In this case the code tables can also be sent partially.

Some further constraints are applied to the acquisition of schema update units, notably to ensure that a decoder will not break in case of a missed schema update unit. A specific schema update unit, the so-called first schema update unit, contains initialization information and shall be acquired by the decoder before any use of a received definition. The decoder behavior in case of such missed schema update units is not normative. A transmitted schema definition shall not change during binary description stream lifetime and there shall not be two schema identifiers associated to the same namespace. Finally, all the optimised decoders associated to existing types are immediately applied to all types they derive from in accordance to the rules defined for the optimized decoders in Clause 9.

Once received by the decoder, a schema update unit immediately updates schema information managed by the decoder. It becomes available for the fragment update unit carried in the same access unit as well as future access units.

In subclause 5.4, replace this paragraph:

An access unit is composed of any number of fragment update units, each of which is extracted in sequence by the fragment update component extractor. Each fragment update unit consists of:

by

In case of TeM, an access unit is composed of any number of fragment update units.

In both TeM and BiM, fragment update units are extracted in sequence by the fragment update component extractor. Each fragment update unit consists of:

In subclause 5.4, replace the following paragraph:

— a fragment update payload conveying the coded description fragment to be added or replaced.

by

— a fragment update payload conveying either the coded description fragment (extracted out of the original description) to be added or replaced, or a reference to it.

In subclause 5.4, replace the following text:

The corresponding update command and context are processed by the non-normative description composer, which either places the description fragment received from the fragment update payload decoder at the appropriate node of the current description tree at composition time, or sends a reconstruction event containing this information to the application. The actual reconstruction of the current description tree by the description composer is implementation-specific, i.e., the application may direct the description composer to prune or ignore unwanted elements as desired. There is no requirement on the format of this current description tree, e.g. it may remain a binary representation.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

by

The corresponding update command and context are processed by the non-normative description composer, which either places the description fragment extracted out of the original description or a reference to it received from the fragment update payload decoder at the appropriate node of the current description tree, or sends a reconstruction event containing this information to the application. If the payload consists of a fragment reference, depending on its nature, the referenced fragment is either immediately acquired (non-deferred fragment reference) or its acquisition is left to the application (deferred fragment references). In case of a deferred fragment reference, a fragment reference marker is available to the application to help further acquisition. This marker consists of the fragment reference itself, the name and type of the top most element of the referenced fragment. The fragment reference marker is added to the current description tree at the location defined by the fragment update context.

The actual reconstruction of the current description tree by the description composer is implementation-specific, i.e., the application may direct the description composer to prune or ignore unwanted elements as desired. There is no requirement on the format of this current description tree, e.g. it may remain a binary representation.

Change the title of subclause 5.5.2 by:

Deferred nodes, fragment references and their use

In subclause 5.5.2, add the following sentence at the end of the first paragraph:

Some deferred nodes are marked with a fragment reference marker that specifies where the fragment can be acquired. It is then left to the application to decide when to acquire it.

In subclause 5.5.2, replace the following sentence:

… The deferred nodes may then be replaced in any subsequent access unit without changing the tree topology maintained internally in the decoder. …

by

… The deferred nodes may then be replaced in any subsequent access unit or on application demand without changing the tree topology maintained internally in the decoder. …

In subclause 5.5.3, remove "ISO/IEC 15938" of the second sentence of the first paragraph.

In subclause 5.5.3, add the following note at the end of the subclause:

Note – Forward compatibility can also be used to generate bitstreams that can be decoded even in case of a schema update unit has not been received (for example because an error occurred) or because the decoder is not able to accept schema update units.

In subclause 5.6.3, replace the following text:

In the TeM, the commands are AddNode, ReplaceNode, and DeleteNode. The AddNode is effectively an "append" command, adding an element of the target node. Insertion between two already-received,

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

consecutive children of a node is not possible. One must replace a previously deferred node. By performing a DeleteNode on a node on the current description tree, the addressable indices of its siblings change appropriately.

by

In the TeM, the commands are AddNode, ReplaceNode, and DeleteNode. The AddNode is by default an “append” command, adding an element after the last child of the target node. The position of the added element can also be explicitely defined allowing an element for instance to be inserted before the first element or between two other elements. By performing a DeleteNode on a node on the current description tree, the addressable indices of its siblings change appropriately.

In subclause 5.6.3, replace the following text:

In the BiM, the commands are AddContent, ReplaceContent and DeleteContent. The AddContent conveys the node data for a node whose path within the description tree is predetermined from the schema evaluation as described in . Hence, internally to the BiM decoder, the paths to (or addresses of) non-empty sibling nodes may be non-contiguous, e.g., the second and fourth occurrence of an element may be present. The “hole” in the numbering is not visible in the current description tree generated by the description composer. Hence, if the third occurrence of said element is added (using AddContent) in a subsequent access unit, it appears to any further processing steps as an “inserted” element in the current description tree, while it simply fills the existing “hole” with respect to the internal numbering of the BiM decoder.

by

In the BiM, the commands are AddContent, ReplaceContent and DeleteContent. The AddContent conveys the node data for a node whose path within the description tree is predetermined from the schema evaluation as described in 5.6.2. The insertion point for the node data is indicated using so called position codes. BiM supports 2 mechanisms for representing these position codes: integer or rational numbers. The first method (integer numbers) requires that "holes" must be left to enable the insertion of new node data. Thus documents to be sent must be known in advance or some contingency holes must be allocated. Hence, if the third occurrence of said element is added (using AddContent) in a subsequent access unit, it appears to any further processing steps as an “inserted” element in the current description tree, while it simply fills the existing “hole” with respect to the internal numbering of the BiM decoder. The “hole” in the numbering is not visible in the current description tree generated by the description composer. However, it is not always possible to know in advance the number of nodes to be added and where within the description tree they are to be added. The second method (rational numbers) removes this limitation by enabling the insertion of new node data at any location within the current binary description tree. A single method shall be used for representing position codes within a given stream. This is signalled within the decoderInit.

In subclause 5.6.4, replace the following text:

Wildcards and mixed content models (defined in ISO/IEC 15938-2) are not supported at all by the BiM. Therefore a schema that uses these mechanisms cannot be supported by the binary format.

by

Wildcards (defined in ISO/IEC 15938-2) are supported by the BiM, only in the case the schema of the elements or attributes validated by the wildcard is known by the decoder (i.e. it is in the initial schema or in one of the additional schemas acquired by the decoder).

The BiM encodes only the canonical form of XML documents. Therefore comments as well as namespace prefixes are lost in the encoding process.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

Add the following subclause (subclause 5.8):

5.8 Decoding of Fragment References

5.8.1 Decoding of Non-Deferred Fragment References

The result of decoding a non-deferred fragment reference shall be, passed to a mechanism (fragment reference resolver) which returns fragment update payload data to the FU payload decoder. This fragment update payload data may be in one of two forms:

— a TeM fragment update payload containing the description fragment data in case of a TeM bitstream;

— a BiM fragment update payload containing the description fragment data in case of a BiM bitstream.

Note - Examples of possible fragment resolver are:
- An HTTP communication session to a WEB server
- A DSM-CC Object carousel

5.8.2 Decoding of Deferred Fragment References

The result of decoding a deferred fragment reference shall be a fragment reference marker which consists of a fragment reference, the name and type of its top most element.

Note - This fragment reference marker is signalled to the application and can be used to acquire the fragment through the fragment reference resolver at any instant of the description stream.

In subclause 6.1, replace the following text:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:mpeg7s="urn:mpeg:mpeg7:systems:2001"
       targetNamespace="urn:mpeg:mpeg7:systems:2001"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">

       <!-- here clause 6 schema definition -->

</schema>
```

by

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"

       xmlns:mpeg7s="urn:mpeg:mpeg7:systems:amd1:2004"
       targetNamespace="urn:mpeg:mpeg7:systems:amd1:2004"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">

       <!-- here clause 6 schema definition -->

</schema>
```

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

In subclause 6.5.2, replace the add command definition by:

addNode	Adds the node conveyed within the `FUPayload` to the context node or the parent node of the context node according to the value of the position attribute of the FUContext. If the position is set to the value “lastChild”, the node conveyed within the FUPayload shall be added to the context node as the last child of the context node, or if the position attribute is set to the value “prevSibling”, the node conveyed within the FUPayload shall be added to the parent node of the context node as the previous sibling of the context node.

In subclause 6.6.1, rename the “`FUContextType`” in the DDL definition to “`FragmentUpdateContextTypeBase`”.

In subclause 6.6.1, add the following DDL definition:

```
<complexType name="FragmentUpdateContextType">
   <simpleContent>
      <extension base="mpeg7s:FragmentUpdateContextTypeBase">
         <attribute name="position" default="lastChild">
            <simpleType>
               <restriction base="string">
                  <enumeration value="lastChild"/>
                  <enumeration value="prevSibling"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </simpleContent>
</complexType>
```

In subclause 6.6.2, add the following semantics:

position	This attribute shall be only present when the FUCommand takes the value “addNode”, and indicates the position of the node added by the “addNode” command against the context node specified by the navigation path. This attribute can have the following values: — “lastChild” : the last child of the context node. — “prevSibling” : the previous sibling of the context node.

In subclause 6.7.1, replace the following text:

```
<complexType name="FragmentUpdatePayloadType">
     <sequence>
       <any processContents="skip" minOccurs="0"/>
     </sequence>

     <attribute name="hasDeferredNodes" type="boolean"
                use="required" default="false"/>
     <anyAttribute namespace="##other" processContents="skip" use="optional"/>
</complexType>
```


© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

by

```
<complexType name="FragmentUpdatePayloadType">
     <sequence>
       <any namespace="##other" processContents="skip" minOccurs="0"/>
       <element name="FragmentReference" type="mpeg7s:FragmentReferenceType"
                   minOccurs="0" />
     </sequence>

     <attribute name="hasDeferredNodes" type="boolean"
                use="required" default="false"/>
     <anyAttribute namespace="##other" processContents="skip" use="optional"/>
</complexType>
```

In subclause 6.7.2, add the following semantics:

FragmentReference	Defines that the current fragment update payload carries a fragment reference instead of a complete fragment. This element shall not be present if fragment payload actually contains a fragment. The element name and type of the top most element of the fragment being referenced shall be carried before the fragment reference itself.

Add the following subclause (subclause 6.8):

6.8 Textual Fragment Reference

6.8.1 Syntax

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

<complexType name="FragmentReferenceType" abstract="true">
   <attribute name="isDeferred" type="boolean" use="optional" default="false"/>
</complexType>

<complexType name="URIFragmentReferenceType" >
   <complexContent>
      <extension base="mpeg7s:FragmentReferenceType">
         <attribute name="href" type="anyURI" use="required" />
      </extension>
   <complexContent>
</complexType>
```

6.8.2 Semantics

The `FragmentReferenceType` is an abstract complex type which serves as a base type for specific implementation of a fragment reference. The `URIFragmentReferenceType` is a concrete complexType which defines fragment references as a URI.

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

Name	*Definition*
isDeferred	Defines the deferred nature of the fragment reference: — If the `isDeferred` attribute has a value of true, the fragment reference is deferred and shall be resolved as defined in subclause 5.8.2. — If the `isDeferred` attribute has value of false, the fragment reference is non-deferred and shall be resolved as defined in subclause 5.8.1.
href	Defines the URI of a fragment refererence of type `URIFragmentReferenceType`.

6.8.3 Examples

In the following, example of the instances of the FUPayload datatype using the fragment reference is shown:

```
<FUPayload>
   <mpeg7:VisualDescriptor xsi:type="mpeg7:ScalableColorType"/>
   <FragmentReference xsi:type="mpeg7s:URIFragmentReferenceType"
      isDeferred="true"
      href="http://aaa.bbb/ccc.xml"/>
</FUPayload>
```

In subclause 7.1, add the following text and figure at the end of the subclause:

Identifying schema components in the BiM framework

As described in Clause 5, BiM relies upon schema knowledge. In this specification, schema components (elements, types and attributes) are identified by both a *schema identifier* and a *component identifier*.

The decoder manages both a unique initial schema and several additional schemas. From the decoder point of view, both initial schemas and additional schemas are indentified through a unique table in which each entry identifies a specific schema: the first '`NumberOfSchemas`' entries identify schemas that are part of the initial schema. The following ones identify additional schemas (starting at the '`NumberOfSchemas`' entry and ending at the '`NumberOfSchemas + NumberOfAdditionalSchemas - 1`').

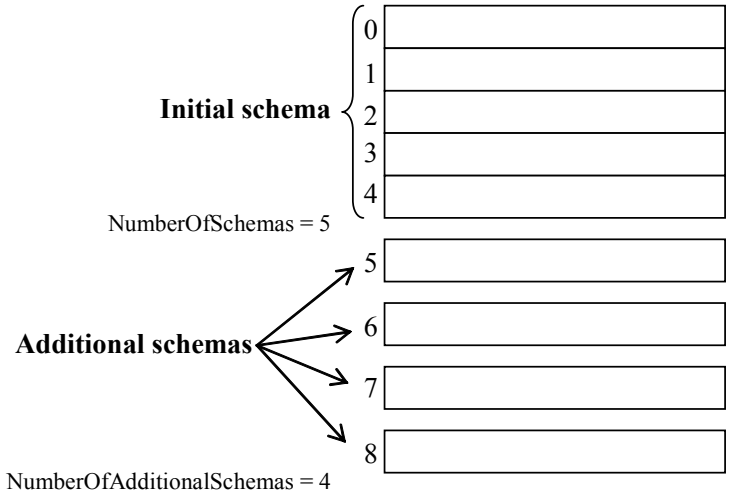


Figure AMD1-2 - Addressing the initial schema and the additional schemas

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

The schema component codes (type codes, element codes or attribute codes) are accessible through all these schemas. However codes are constructed differently depending on which schema they are defined. The initial schema aggregates all schema components possibly coming from different namespaces in a single code space. On the contrary, additional schemas contains only schema components which are defined in their namespace.

In subclause 7.2.1, replace the following text:

An optimised type codec specifies an encoding for a data type not using the generic mechanism specified in Clause 8 but optimised with full knowledge of the properties of that data type. Some optimised type codecs are specified in Part 3 of this specification.

by

An optimised type codec specifies an encoding for a data type not using the generic mechanism specified in Clause 8 but optimised with full knowledge of the properties of that data type. There are two kinds of optimised type codec (or optimised decoders). A fixed optimised decoder associates a specific encoding scheme to a type of the schema (complex as well as simple) and this association is fixed for the entire stream. An advanced optimised decoder associates a specific encoding scheme to any simple type and this association can be changed during the transmission of the bitstream. Moreover, several advanced optimised decoders can be associated to a single type and can accept parameters. Some fixed optimised type codecs are specified in Part 3 of the specification. Some advanced optimised decoders are defined in Clause 9.

In subclause 7.2.1, add the following text at the end of the subclause:

Several other coding modes are initialised in the `DecoderInit` related to the features used by the binary description stream: the insertion of elements, the transmission of schema information and references to fragments.

Transmission of additional schema is specified for two different use cases: The retrieval of schema information in binary format from a location indicated by a URI, the transmission of schema information in a binary description stream jointly or not with the transmission of a description. In the latter case there is a requirement that all schema information needed for the decoding of a fragment of the transmitted description must have been received before such fragment arrives.

Replace subclause 7.2.2 content by:

DecoderInit () {	**Number of bits**	**Mnemonic**
SystemsProfileLevelIndication	8+	vluimsbf8
UnitSizeCode	3	bslbf
NoAdvancedFeatures	1	bslbf
ReservedBits	4	bslbf
If (! NoAdvancedFeatures) {		
AdvancedFeatureFlags_Length	8+	vluimsbf8
/** FeatureFlags **/		
InsertFlag	1	bslbf
AdvancedOptimisedDecodersFlag	1	bslbf
AdditionalSchemaFlag	1	bslbf

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

AdditionalSchemaUpdatesOnlyFlag	1	bslbf
FragmentReferenceFlag	1	bslbf
MPCOnlyFlag	1	bslbf
HierarchyBasedSubstitutionCodingFlag	1	bslbf
ReservedBitsZero	FeatureFlags_Length*8-6	bslbf
/** FeatureFlags end **/		
}		
*/** Start FUUConfig **/*		
If(! AdditionalSchemaUpdatesOnlyFlag) {		
NumberOfSchemas	8+	vluimsbf8
for (k=0; k< NumberOfSchemas; k++) {		
SchemaURI_Length[k]	8+	vluimsbf8
SchemaURI[k]	8* SchemaURI_Length[k]	bslbf
LocationHint_Length[k]	8+	vluimsbf8
LocationHint[k]	8* LocationHint_Length[k]	bslbf
NumberOfTypeCodecs[k]	8+	vluimsbf8
for (i=0; i< NumberOfTypeCodecs[k]; i++) {		
TypeCodecURI_Length[k][i]	8+	vluimsbf8
TypeCodecURI[k][i]	8* TypeCodecURI_Length[k][i]	bslbf
NumberOfTypes[k][i]	8+	vluimsbf8
for (j=0; j< NumberOfTypes[k][i]; j++) {		
TypeIdentificationCode[k][i][j]	8+	vluimsbf8
}		
}		
}		
*/** FUUConfig - Advanced optimised decoder framework **/*		
If (AdvancedOptimisedDecodersFlag) {		
NumOfAdvancedOptimisedDecoderTypes	8+	vluimsbf8
for (i=0; i< NumOfAdvancedOptimisedDecoderTypes; i++) {		
AdvancedOptimisedDecoderTypeURI_Length[i]	8+	vluimsbf8
AdvancedOptimisedDecoderTypeURI[i]	8* AdvancedOptimisedDecoderTypeURI_Length[i]	bslbf
}		
AdvancedOptimisedDecodersConfig ()		
}		

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

/** FUUConfig - Fragment reference framework **/		
If (FragmentReferenceFlag) {		
NumOfSupportedFragmentReferenceFormat	8	uimsbf
for (i=0;i< NumOfSupportedFragmentReferenceFormat;i++) {		
SupportedFragmentReferenceFormat[i]	8	blsbf
}		
}		
}		
/** end FUUConfig **/		
If (AdditionalSchemaFlag) {		
AdditionalSchemaConfig ()		
}		
/** Initial description **/		
If (!AdditionalSchemaUpdateOnlyFlag) {		
InitialDescription_Length	8+	vluimsbf8
InitialDescription()		
}		
}		

In subclause 7.2.3, add the following semantics after the sematics of the `UnitSizeCode` *syntax element:*

NoAdvancedFeatures	Signals that none of the following advanced features is used in the binary stream: — dynamic insertions of child elements in the binary current description tree; — advanced optimised decoders; — schema transmission; — fragment references; — position codes only based on MPC.
AdvancedFeatureFlags_Length	Defines the number of bytes used for the indication of the advanced features. Note – This length provides a simple framework for future extensions.
InsertFlag	Signals that the insertion of child elements in the binary description tree at specific positions is performed by the use of rationale position code as described in subclause 7.6.5.5.
AdvancedOptimisedDecodersFlag	Signals that advanced optimised decoders are supported as described in Clause 9.
AdditionalSchemaFlag	Signals that additional schemas are supported.

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

AdditionalSchemaUpdatesOnlyFlag	Signals that the description stream contains only additional schema updates i.e. no fragment update units. The `AdditionalSchemaFlag` shall be set to true when this flag is set to true.
FragmentReferenceFlag	Signals that fragment references are supported.
MPCOnlyFlag	Signals that position codes in the fragment update context are encoded in MPC mode only.
HierarchyBasedSubstitutionCodingFlag	Signals that element substitution codes are computed taking into account their substitution hierachy. If additional schemas are supported (i.e. `AdditionalSchemaFlag`==true) this flag shall be set to true.

In subclause 7.2.3, replace the "NumberOfSchemas" semantics by:

NumberOfSchemas	Indicates the number of schemas on which the description stream is based. These schemas compose the initial schema. A zero-value is forbidden.

In subclause 7.2.3, add the following semantics after the sematics of the `TypeIdentificationCode` syntax element

NumOfAdvancedOptimisedDecoderTypes	Defines the number of advanced optimised decoder types that are necessary to properly decode the binary description stream.
AdvancedOptimisedDecoderTypeURI_Length[i]	Indicates the size in bytes of the `AdvancedOptimisedDecoderTypeURI[i]` syntax element.
AdvancedOptimisedDecoderTypeURI[i]	Defines the UTF-8 representation of the URI referencing the advanced optimised decoder type with index i.
AdvancedOptimisedDecodersConfig()	See subclause 9.2.
NumOfSupportedFragmentReferenceFormat	Specifies the number of fragment reference format that shall be supported by the decoder.
SupportedFragmentReferenceFormat[i]	Specifies the ith fragment reference format, according to Table AMD1-1, that shall be supported by the decoder. The `SupportedFragmentReferenceFormat[0]` indicates the default fragment reference format.
AdditionalSchemaConfig()	See subclause 7.2.4.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

In subclause 7.2.3, add the following table:

Table AMD1-1 - Fragment Reference Formats

Fragment Reference Type	***Fragment Reference Format***	***Description***
0		ISO reserved
1	URIFragmentReference	This fragment reference format should be used where the reference can be expressed as a URI.
2 - 224		ISO reserved
225 – 255		Private Use

Add the following subclause (subclause 7.2.4):

7.2.4 Syntax of AdditionalSchemaConfig

AdditionalSchemaConfig () {	**Number of bits**	**Mnemonic**
NumberOfAdditionalSchemas	8+	vluimsbf8
NumberOfKnownAdditionalSchemas	8+	vluimsbf8
for (int t=0;t<NumberOfKnownAdditionalSchemas;t++){		
KnownAdditionalSchemaID	8+	
AdditionalSchemaURI_Length[KnownAdditionalSchemaID]	8+	vluimsbf8
AdditionalSchemaURI[KnownAdditionalSchemaID]	8* AdditionalSchemaURI_Length [KnownAdditionalSchemaID]	bslbf
BinaryLocationHint_Length[KnownAdditionalSchemaID]	8+	vluimsbf8
BinaryLocationHint[KnownAdditionalSchemaID]	8*BinaryLocationHint_Length[KnownAdditionalSchemaID]	bslbf
NumberOfTypeCodecs[KnownAdditionalSchemaID]	8+	vluimsbf8
for (i=0; i< NumberOfTypeCodecs[KnownAdditionalSchemaID]; i++) {		
TypeCodecURI_Length[KnownAdditionalSchemaID][i]	8+	vluimsbf8
TypeCodecURI[KnownAdditionalSchemaID][i]	8* TypeCodecURI_Length[KnownAdditionalSchemaID][i]	bslbf
NumberOfTypes[KnownAdditionalSchemaID][i]	8+	vluimsbf8
for (j=0; j< NumberOfTypes[KnownAdditionalSchemaID][i]; j++) {		
TypeIdentificationCode[KnownAdditionalSchemaID][i][j]	8+	vluimsbf8
}		
}		

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

}		
SchemaEncodingMethod	8	blsbf
ExternallyCastableTypeTable(InitialSchema)		
ExternallySubstitutableElementTable(InitialSchema)		
ReservedBitsZero	7	blsbf
}		

Add the following subclause (subclause 7.2.5):

7.2.5 Semantics of AdditionalSchemaConfig

Name	*Definition*
NumberOfAdditionalSchemas	Indicates the number of schemas that can be transmitted and that are not declared in the list of `schemaURI`. If additional schemas are not supported, this value is set to zero.
NumberOfKnownAdditionalSchemas	Indicates the number of additional schemas that are known to be updated in the bitstream.
KnownAdditionalSchemaID	Identifies a schema known to be updated in the bistream. This identifier shall only address an additional schema i.e. its value shall be superior to '`NumberOfSchemas-1`'
AdditionalSchemaURI_Length[KnownAdditionalSchemaID]	Indicates the size in bytes of the `AdditionalSchemaURI[KnownAdditionalSchemaID]` length. A value of zero is forbidden.
AdditionalSchemaURI[KnownAdditionalSchemaID]	Indicates the UTF-8 representation of the URI of the additional schema identified by `KnownAdditionalSchemaID`. Note – This field allows to identify some of the additional schemas that are expected to be updated. This information allows one decoder not to monitor the schema updates for which it already knows the schema.
BinaryLocationHint_Length[KnownAdditionalSchemaID]	Indicates the size in bytes of the `BinaryLocationHint_Length[KnownAdditionalSchemaID]`. A value of zero indicates that for the schema that is referenced by the index `KnownAdditionalSchemaID` there is no binary encoded schema available.
BinaryLocationHint[KnownAdditionalSchemaID]	This is the UTF-8 representation of the URI to unambiguously reference the location of the binary encoded schema that is referenced by the index `KnownAdditionalSchemaID`. The schema can be fetched by the schema resolver and is then received as a description stream composed only of schema update units i.e. for which the `SchemaOnlyFlag` is set to true.
NumberOfTypeCodecs[KnownAdditionalSchemaID]	see `NumberOfTypeCodecs[k]` in 7.2.3.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

TypeCodecURI_Length[KnownAdditionalSchemaID]	see `TypeCodecURI_Length[k]` in 7.2.3.
TypeCodecURI[KnownAdditionalSchemaID][i]	see `TypeCodecURI[k][i]` in 7.2.3.
NumberOfTypes[KnownAdditionalSchemaID][i]	see `NumberOfTypes[k][i]` in 7.2.3.
TypeIdentificationCode[KnownAdditionalSchemaID][i][j]	see `TypeIdentificationCode[k][i][j]` in 7.2.3.
SchemaEncodingMethod	Indicates the encoding method of the schema update units.
ExternalCastableTypeTable	Defines the types of the initial schema that are externally castable as defined in subclause 7.7.5.4 and 7.7.5.5.
ExternalSubstitutableElementTable	Defines the elements of the initial schema that are substitutable as defined in subclause 7.7.6.4 and 7.7.6.5.

Table AMD1-2 - Schema encoding method

SchemaEncodingMethod	***definition***
0	ISO reserved
1	BiM encoded schema as described in subclause 7.7.8
2-224	ISO reserved
225-255	Private use

In subclause 7.3.2, replace the `AccessUnit` *syntax by:*

AccessUnit () {	**Number of bits**	**Mnemonic**
If (AdditionalSchemaFlag) {		
NumberOfSUU	8+	vluimsbf8
for (i=0; i< NumberOfSUU ; i++) {		
SchemaUpdateUnit()		
}		
}		
If(! AdditionalSchemaUpdateOnlyFlag) {		
NumberOfFUU	8+	vluimsbf8
for (i=0; i< NumberOfFUU ; i++) {		
FragmentUpdateUnit()		
}		
}		
}		

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

In subclause 7.3.3, replace the semantics table by:

Name	*Definition*
NumberOfSUU	Indicates the number of schema update units in this access unit. Value ‘0’ signifies that no schema update unit is carried.
NumberOfFUU	Indicates the number of fragment update units in this access unit. Value '0' signifies that no fragment update unit is carried.

In subclause 7.4.2, insert the following syntax elements, between the `FUU_Length` *and the* `FragmentUpdateCommand` *syntax elements:*

If (AdvancedOptimisedDecodersFlag){		
OptimisedDecoderReparameterization	2	bslbf
if (OptimisedDecoderReparameterization == '00') {		
AdvancedOptimisedDecodersConfig ()		
}		
}		

In subclause 7.4.3, insert the following semantics, between the `FUU_Length` *and the* `FragmentUpdateCommand` *semantics:*

OptimisedDecodersReparameterization	This 2-bit flag signals if the parameters of the optimised decoders shall be updated. It can take the following values: — ‘00' – the optimised decoder instance table and mappings shall be redefined; — '01' – the optimised decoder instance table and mappings shall not be redefined; — '10' – the optimised decoder instance table and mappings are reset to the default table and mappings defined in the `DecoderInit`; — '11' – reserved.
AdvancedOptimisedDecodersConfig()	See subclause 9.2.

In subclause 7.6.1, replace the following text:

There are two different TBC tables associated to each complexType: The ContextTBC table contains only references to the child elements of complexType and additionally one code word to signal the termination of the path (Path Termination Code). The ContextTBC table contains also one TBC to refer to the parent node. It allows relative navigation within the binary format description tree and move upwards to the parent node. The OperandTBC table additionally contains also the references to the attributes and either to the elements of simpleType or to a simple content, but does not contain the Path Termination Code nor the reference to the parent node. Furthermore, in the OperandTBC table one TBC is reserved for User Data Extension as defined in section 7.6.5.2. Example TBC tables are shown in Table 4 and Table 5.

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

by

There are two different TBC tables associated to each complexType: The ContextTBC table contains only references to the child elements of complexType and additionally one code word to signal the termination of the path (Path Termination Code). The ContextTBC table contains also one TBC to refer to the parent node. It allows relative navigation within the binary format description tree and move upwards to the parent node. The OperandTBC table additionally contains also the references to the attributes and either to the elements of simpleType or to a simple content, but does not contain the Path Termination Code nor the reference to the parent node. Furthermore, in the OperandTBC table one TBC is reserved for User Data Extension as defined in subclause 7.6.5.2. In case of a mixed content model the OperandTBC table also contains a reference to the character data that may appear between the elements. Example TBC tables are shown in Table 4 and Table 5.

Add the following sentence before the Table "Example of a Context TBC Table":

In this example the content model of the complex type definition is not 'mixed'.

In subclause 7.6.2, replace the `SchemaID` *syntax element in the* `FragmentUpdateContext` *syntax table:*

SchemaID	ceil(log2(`NumberOfSchemas` + `NumberOfAdditionalSchemas`))	uimsbf

In subclause 7.6.2, replace the `ContextPath` *syntax table by:*

ContextPath () {	**Number of bits**	**Mnemonic**
TBC_Counter = 0		
NumberOfFragmentPayloads = 1		
do {		
if ((ContextModeCode == '001' || ContextModeCode == '011') && TBC_Counter ==0) { /* *absolute addressing mode and first TBC of the context path* */		
If (AdditionalSchemaFlag) {		
SchemaIDOfSBC_Context_Selector	ceil(log2(`NumberOfSchemas` + `NumberOfAdditionalSchemas`))	uimsbf
Extended_SBC_Context_Selector	ceil(log2(number_of_global elements in `SchemaIDOfSBC_Context_Selector`))	bslbf
} else {		
SBC_Context_Selector	ceil(log2(number of global elements +1))	bslbf
}		
PathTypeCode()		

© ISO/IEC 2005 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005

}		
else {		
SBC_Context	ceil(log2(number of child elements of complexType + 2))	bslbf
If (SBC_Context == “SBC_any”) {		
AnyElementDecoding ()		
} else {		
SubstitutionCode()		
}		
PathTypeCode()		
}		
TBC_Counter ++		
} while ((SBC_Context_Selector != “Path Termination Code”) && (SBC_Context != “Path Termination Code”))		
if (SBC_Context_Selector == “Path Termination Code”)){		
If (AdditionalSchemaFlag) {		
SchemaIDOfSBC_Operand_Selector	ceil(log2(`NumberOfSchemas` + `NumberOfAdditionalSchemas`))	uimsbf
Extended_SBC_Operand_Selector	ceil(log2(number_of_global elements in `SchemaIDOfSBC_Operand_Selector`))	bslbf
} else {		
SBC_Operand_Selector	ceil(log2(number of global elements))	bslbf
}		
PathTypeCode()		
}		
else {		
SBC_Operand	ceil(log2(number of child elements + number of attributes + has_simpleContent + 1))	bslbf
if (SBC_Operand == “SBC_anyAttribute”) {		
SingleAnyAttributeDecoding()		
}		
if (SBC_Operand == “SBC_any”) {		
AnyElementDecoding()		
}		
SubstitutionCode()		
PathTypeCode()		
}		

© ISO/IEC 2005 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 15938-1_2002_Amd 1_2005