INTERNATIONAL STANDARD

ISO/IEC 23001-1

First edition
2006-04-01

AMENDMENT 2
2008-02-15

Information technology — MPEG systems technologies —

Part 1: Binary MPEG format for XML

AMENDMENT 2: Conservation of prefixes and extensions on encoding of wild cards

Technologies de l'information — Technologies des systèmes MPEG —

Partie 1: Format binaire de MPEG pour XML

AMENDEMENT 2: Conservation des préfixes et des extensions pour l'encodage des caractères de remplacement

Reference number
ISO/IEC 23001-1:2006/Amd.2:2008(E)

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 23001-1:2006 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 — MPEG systems technologies —

Part 1:
Binary MPEG format for XML

AMENDMENT 2: Conservation of prefixes and extensions on encoding of wild cards

In 6.2.2, replace:

DecoderInit () {	Number of bits	Mnemonic
SystemsProfileLevelIndication	8+	vluimsbf8
…		

with:

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
AdditionalSchemaUpdatesOnlyFlag	1	bslbf
FragmentReferenceFlag	1	bslbf
MPCOnlyFlag	1	bslbf
HierarchyBasedSubstitutionCodingFlag	1	bslbf
ContextPathTableFlag	1	bslbf
PrefixTableFlag	1	bslbf
ReservedBitsZero /for additional Flags **/**	24	bslbf
If (**PrefixTableFlag**) {		
PrefixTable ()		
}		
ReservedBitsZero	AdvancedFeatureFlags_Lent	

	h*8-33-PrefixTable_length*8	
}/** 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
}		
}		
}		
If (ContextPathTableFlag) {		
ContextPathTable()		
}		
/** FUUConfig - Advanced optimised decoder framework **/		
If (AdvancedOptimisedDecodersFlag) {		
NumOfAdvancedOptimisedDecoderTypes	8+	vluimsbf8
for (i=0; i< NumOfAdvancedOptimisedDecoderTypes; i++) {		
AdvancedOptimisedDecoderTypeURI_Length[i]	8+	vluimsbf8
AdvancedOptimisedDecoderTypeURI[i]	8* AdvancedOptimisedDecoder TypeURI_Length[i]	bslbf
}		
AdvancedOptimisedDecodersConfig ()		
}		
/** 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 document **/		
If (!AdditionalSchemaUpdateOnlyFlag) {		
InitialDocument_Length	8+	vluimsbf8
InitialDocument()		
}		
}		

Add the following paragraph:

PrefixTable {		
NumberofPrefixes	8+	vluimsbf8
for (k=0; k< NumberOfPrefixes; k++) {		
SchemaURI_Length[k]	8+	vluimsbf8
SchemaURI[k]	8* SchemaURI_Length[k]	bslbf
Prefix_Length[k]	8+	vluimsbf8
Prefix[k]	8* [Prefix_Length[k]]	bslbf
}		
}		

Add the following paragraph:

Semantics

Name	*Definition*
PrefixTable_length	Indicates length in Bytes of PrefixTable.
PrefixTableFlag	Indicates whether a prefix table is given in the `decoderInit`.
NumberofPrefixes	Specifies the number of prefixes defined in the PrefixTable.
Prefix_Length[k]	Indicates the size in bytes of the Prefix[k]. A value of zero is forbidden.
Prefix[k]	This is the UTF-8 representation of the prefix associated to SchemaURI[k] in the `PrefixTable`. The decoder SHOULD bind the namespace names in the current document to the prefixes given in the `PrefixTable` if present. The default namespace, used to construct unprefixed elements, will be declared as a null string “”. If different prefixes are associated to a same namespace name in the

`PrefixTable`, or a given namespace name is associated to different prefixes, the decoder should only use the first defined prefix/namespace binding in the output document.

NOTE A document author wishing to use prefix conservation in order to maintain, after decoding, the validity of QNames/XPath expressions in the content of an element should only use one global prefix/namespace name mapping for those prefix and namespace names that are used in the QNames/XPath expressions. However, in order to ensure in all cases the validity of those expressions after decoding, the optimized decoder for QName expressions specified in subclause 8.8 should be used.

In 6.2.2, replace:

Table 1 — Index Table for SystemsProfileLevelIndication

Index	***Systems Profile and Level***
0	no profile specified
1 – 127	Reserved for ISO Use

with:

Table 1 — Index Table for SystemsProfileLevelIndication

Index	***Systems Profile and Level***
0	no profile specified
1 – 124	Reserved for ISO Use
125	ETSI TS 102 323/102 529 (DVB GBS TV-A and DVB IPTV BCG)
126	ETSI TS 102 034 (DVB IPTV SD&S)
127	ETSI TS 102 471 (DVB IPDC ESG)
128-	User specific

In 7.5.2.4.5.2 (AnyElementDecoding Syntax), replace:

AnyElementDecoding() {	**Number of bits**	**Mnemonic**
GlobalElementSchemaID	ceil(log2(`NumberOfSchemas` + `NumberOfAdditionalSchemas`))	uimsbf
AnyElement_Length	5+	vluimsbf
Any_SBC_Operand_Selector	5+	vluimsbf
If (inPayloadDecoding()) {		
Element(ChildrenSchemaMode, theAnyType)		
}		
}		

with:

AnyElementDecoding() {	**Number of bits**	**Mnemonic**
GlobalElementSchemaID	ceil(log2(`NumberOfSchemas + NumberOfAdditionalSchemas`))	uimsbf
AnyElement_Length	5+	vluimsbf
if (GlobalElementSchemaID == LAXSchemaID) {		
//lax encoding		
AnyElementLaxDecoding();		
} else {		
Any_SBC_Operand_Selector	5+	vluimsbf
If (inPayloadDecoding()) {		
Element(ChildrenSchemaMode, theAnyType)		
}		
}		
}		

In 7.5.2.4.5.3 (AnyElementDecoding Semantics), replace:

Name	*Definition*
GlobalElementSchemaID	The schema in which the global element is defined. Its value is the index of the URI in the `SchemaURI` array defined in 6.2 (optionally extended with the list of additional schemas).
AnyElement_Length	Indicates the length in bits of the remainder of this `AnyElementDecoding`.
Any_SBC_Operand_Selector	Selects one global element of the schema referenced by `GlobalElementSchemaID` using the `OperandTBC` table for Extended_SBC_Operand_Selector as specified in 6.6.5.2.3. Therefore, Any_SBC_Operand_Selector is equivalent to the Extended_SBC_Operand_Selector but with a bit representation in vluimsbf5.
inPayloadDecoding()	Returns true if the AnyElementDecoding procedure has been triggered from a payload decoding procedure.
Element()	See 7.4.1.
theAnyType	The type of the element identified by the `SBC_GlobalElement_SelectorCode` as defined in the schema identified by the `GlobalElementSchemaID`.

with:

Name	*Definition*
GlobalElementSchemaID	The schema in which the global element is defined. Its value is the index of the URI in the `SchemaURI` array defined in 6.2 (optionally extended with the list of additional schemas). If GlobalElementSchemaID is equal to the ID of the virtual lax schema `urn:mpeg:mpegb:bim:laxencoding:2007` specified in the decoderInit (i.e., GlobalElementSchemaID == LAXSchemaID), the lax decoding procedure AnyElementLaxDecoding() is called.
AnyElement_Length	Indicates the length in bits of the remainder of this `AnyElementDecoding`.
Any_SBC_Operand_Selector	Selects one global element of the schema referenced by `GlobalElementSchemaID` using the `OperandTBC` table for Extended_SBC_Operand_Selector as specified in 6.6.5.2.3. Therefore, Any_SBC_Operand_Selector is equivalent to the Extended_SBC_Operand_Selector but with a bit representation in vluimsbf5.
inPayloadDecoding()	Returns true if the AnyElementDecoding procedure has been triggered from a payload decoding procedure.
Element()	See 7.4.1.
theAnyType	The type of the element identified by the `SBC_GlobalElement_SelectorCode` as defined in the schema identified by the `GlobalElementSchemaID`.
AnyElementLaxDecoding()	See 7.5.2.4.5.4.

In 7.5.2.4.5 (Wildcard transition behavior), add:

7.5.2.4.5.4 AnyElementLaxDecoding procedure

7.5.2.4.5.4.1 AnyElementLaxDecoding

AnyElementLaxDecoding() {	**Number of bits**	**Mnemonic**
ReservedBits	4	bmsbf
AnyElementLaxCompressionScheme	4	bmsbf
AnyElementLaxDecodingContent()		
}		

Name	*Definition*
ReservedBits	Reserved bits for future use.
AnyElementLaxCompressionScheme	Indicates the compression scheme used in the AnyElementLaxDecoding() procedure. Table AMD2.1 gives the possible values.
AnyElementLaxDecodingContent	See 7.5.2.4.5.4.2.

Table AMD2.1 — Compression methods

AnyElementLaxCompressionScheme	*Definition*
0	No compression
1	Compression method described in 7.5.2.4.5.4.2
2-10	ISO reserved
11-16	Private use

7.5.2.4.5.4.2 AnyElementLaxDecodingContent

7.5.2.4.5.4.2.1 Overview

The `AnyElementLaxDecodingContent` function contains the decoding of lax format syntax. This is a byte-aligned SAX tokenized events stream, which is decoded by a large switch statement.

Three dynamic dictionaries are used to store the XML structural items names: namespaces, elements names and attributes names. These strings dictionaries are dynamic and can grow during the encoding process, with the help of the ADD_NS, ADD_ENAME and ADD_ANAME special events, discussed below. By default, these dictionaries are initialized as empty ones.

Currently understood events are described in Table AMD2.2. All SAX events are defined by a corresponding SAX event callback and three special events are used to dynamically add a namespace, an element name or an attribute name in the corresponding dictionary. The first event with zero UID triggers decoding according to the general BiM syntax.

The UID is a fixed numerical ID able to unambiguously define an event, but this is not the value used to encode an event, as explained in 7.5.2.4.5.4.3. An event can carry zero, one or several parameters, which can be strings or numerical IDs, which point to the corresponding dynamic strings dictionary. Strings are encoded in UTF-8 format, with a terminating zero.

