INTERNATIONAL STANDARD ISO/IEC 15938-1:2002/Amd.1:2005
TECHNICAL CORRIGENDUM 1

Published 2005-11-15

Information technology — Multimedia content description interface —

Part 1: **Systems**

AMENDMENT 1: Systems extensions

TECHNICAL CORRIGENDUM 1

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

Partie 1: Systèmes

AMENDEMENT 1: Extensions des systèmes

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 15938-1:2002/Amd.1:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

ICS 35.040 **Ref. No. ISO/IEC 15938-1:2002/Amd.1:2005/Cor.1:2005(E)**

In subclause 7.1, replace:

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.

with:

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 in. The initial schema aggregates all schema components possibly coming from different namespaces in a single code space. On the contrary, additional schemas contain only schema components which are defined in their namespace.

In subclause 7.2.1, replace:

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.

with:

Transmission of additional schemas 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 the following row in the table of subclause 7.2.2:

ReservedBitsZero	FeatureFlags_ Length*8-6	bslbf

with:

ReservedBitsZero	FeatureFlags_Length*8-7	bslbf

Revise the table in subclause 7.2.4 as highlighted below:

AdditionalSchemaConfig () {	**Number of bits**	**Mnemonic**
NumberOfAdditionalSchemas	8+	vluimsbf8
NumberOfKnownAdditionalSchemas	8+	vluimsbf8
for (int t=1;t<NumberOfKnownAdditionalSchemas-1;t++){		
KnownAdditionalSchemaID	8+	
AdditionalSchemaURI_Length[KnownAdditionalSchemaID]	8+	vluimsbf8
AdditionalSchemaURI[KnownAdditionalSchemaID]	8* AdditionalSchema URI_Length [KnownAdditional SchemaID]	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
}		
}		
ExternallyCastableTypeTable(KnownAdditionalSchemaID)		
ExternallySubstitutableElementTable(KnownAdditionalSchemaID)		
}		
SchemaEncodingMethod	8	blsbf
ExternallyCastableTypeTable(InitialSchema)		
ExternallySubstitutableElementTable(InitialSchema)		
ReservedBitsZero	7	blsbf
}		

Replace the following row in the table in subclause 7.2.5:

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.

with:

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. If additional schemas are supported then the value of GlobalSchemaID = `NumberOfSchemas is reserved for` a virtual namespace for attributes that do not belong to any namespace. `The value of` GlobalSchemaID = `NumberOfSchemas +` NumberOfAdditionalSchemas – 1 is reserved for ISO usage.

Replace the following row in the table in subclause 7.2.5:

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`'

with:

KnownAdditionalSchemaID	Identifies a schema known to be updated in the bitstream. This identifier shall only address an additional schema. The value KnownAdditionalSchemaID = `NumberOfSchemas` shall be reserved for attributes that do not belong to any namespace. The value of KnownAdditionalSchemaID = `NumberOfSchemas` + NumberOfAdditionalSchemas – 1 shall be reserved for ISO Use.

In subclause 7.7.1, replace:

A schema update unit is composed of a namespace identifier, a set of code tables to represent global elements, global types and global attributes, followed by a binary encoded schema carrying the schema components definitions. This binary encoded schema is encoded using a specific profile of BiM specified in subclause 7.7.8 using a simple XML schema for schema encoding has been defined for this purpose in this specification.

with:

A schema update unit is composed of a namespace identifier, a set of code tables to represent global elements, global types and global attributes, followed by a binary encoded schema carrying the schema components definitions. This binary encoded schema is encoded using a specific profile of BiM specified in subclause 7.7.8. For that purpose a simple XML schema for schema encoding has been defined in this specification.

Replace subclause 8.5.2.4.5 with the following text:

8.5.2.4.5 Wildcard transition behaviour

8.5.2.4.5.1 Introduction

Wildcard transitions, when crossed, specify to the decoder that an element not known a priori is present in the description. The decoder shall execute the following decoding procedure or skip its decoding if the element in the schema is unknown.

8.5.2.4.5.2 AnyElementDecoding Syntax

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)		
}		
}		

8.5.2.4.5.3 AnyElementDecoding Semantics

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 7.2 (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 7.6.5.2.3. 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 subclause 8.4.1
theAnyType	The type of the element identified by the `SBC_GlobalElement_SelectorCode` as defined in the schema identified by the `GlobalElementSchemaID`.

Replace subclause 8.5.3.2 with the following text:

8.5.3.2 AnyAttribute Decoding

8.5.3.2.1 Syntax

AnyAttributeDecoding() {	**Number of bits**	**Mnemonic**
AnyAttributePresence	1	bslbf
If (AnyAttributePresenceFlag) {		
while (hasNextSchema ==1) {		
SingleSchemaAnyAttributeDecoding()		
hasNextSchema	1	bslbf
}		
}		
}		

8.5.3.2.2 Semantics

Name	*Definition*
AnyAttributePresence	Indicates the presence of at least one attribute matched by the anyAttribute wildcard.
hasNextSchema	Indicates whether there is another schema defined. This variable is set to 1 upon entering the AnyAttributeDecoding()

Replace subclause 8.5.3.3 with the following text:

8.5.3.3 Single Schema Any Attribute Decoding

8.5.3.3.1 Syntax

SingleSchemaAnyAttributeDecoding() {	**Number of bits**	**Mnemonic**
GlobalAttributeSchemaID	ceil(log2(NumberOfSchemas + NumberOfAdditionalSchemas))	uimsbf
SchemaAnyAttribute_Length	5+	vluimsbf
while (hasNextAttribute ==1){		
GlobalAttributeID	5+	vluimsbf
if (inPayloadDecoding()) {		
SimpleType(theGlobalAttributeType)		
}		
hasNextAttribute	1	bslbf
}		
}		

8.5.3.3.2 Semantics

Name	*Definition*
GlobalAttributeSchemaID	Identifies the schema in which the global attribute is defined. Its value is the index of the URI in the SchemaURI array defined in 7.2 (optionally extended with the list of additional schemas).
SchemaAnyAttribute_Length	The length in bits of the SingleSchemaAnyAttributeDecoding syntax element.
GlobalAttributeID	Identifies the global attribute amongst all the attributes defined in the schema identified by the GlobalAttributeSchemaID. Global attributes code words are assigned in lexicographical order of their expanded names.
inPayloadDecoding()	Returns true if the AnyElementDecoding procedure has been triggered from a payload decoding procedure.
SimpleType()	See subclause 8.4.7.
theGlobalAttributeType	The type of the global attribute identified by the GlobalAttributeID defined in the schema identified by the GlobalAttributeSchemaID.
hasNextAttribute	Indicates whether there is another attribute defined in this schema. This variable is set to 1 upon entering the SingleSchemaAnyAttributeDecoding.