INTERNATIONAL STANDARD ISO/IEC 23001-1:2006
TECHNICAL CORRIGENDUM 1

Published 2007-02-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — MPEG systems technologies —

Part 1:
Binary MPEG format for XML

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Technologies des systèmes MPEG —

Partie 1: Format binaire de MPEG pour XML

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 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.*

ICS 35.040 **Ref. No. ISO/IEC 23001-1:2006/Cor.1:2007(E)**

Published in Switzerland

In subclause 3.2.32, replace the paragraph:

The ordered concatenation of either binary or textual access units conveying a single, possibly time-variant, document.

with:

The ordered concatenation of binary access units conveying a single, possibly time-variant, document.

In subclause 5.5.3 replace the third and fourth paragraphs with:

Backward compatibility is provided by the unique reference of the used schema in the `DecoderInit` using its Schema URI as its namespace identifier.

Forward compatibility is ensured by a specific syntax defined in Clauses 6 and 7. Its main principle is to use the namespace of the schema, i.e., the Schema URI, as a unique version identifier. The binary format allows one to keep parts of a document related to different schema in separate chunks of the binary document stream, so that parts related to unknown schema may be skipped by the decoder. In order for this approach to work, an updated schema should not be defined using the ISO/IEC 23001-1 "redefine" construct but should be defined in a new namespace. The Decoder Initialisation identifies schema versions with which compatibility is preserved by listing their Schema URIs. A decoder that knows at least one of the Schema URIs will be able to decode at least part of the binary document stream.

In subclause 5.6, replace the seventh item of the third paragraph with:

- The DL shall provide a means to deliver the DecoderInit information (see subclause 6.2) to the terminal before any access unit decoding occurs.

In subclause 5.7.1, replace the first paragraph with:

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 is in the following form:

- a BiM fragment update payload containing the document fragment data.

In subclause 6.6.2, replace the first table 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
ReservedBitsZero	AdvancedFeatureFlags_Length*8-8	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
}		
}		
}		
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* AdvancedOptimisedDecoderTypeURI_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 (!AdditionalSchemaUpdatesOnlyFlag) {		
InitialDocument_Length	8+	vluimsbf8
InitialDocument()		
}		
}		

Add the following missing semantics at the end of Semantics paragraph:

NoAdvancedFeatures	If true, signals that no advanced features are present in the decoderInit.
AdvancedFeatureFlags_Length	Length (in Bytes) of advanced features in the decoderInit.
InsertFlag	If true, signals that Position Codes represent rational numbers (Rational Position Codes). See subclause 6.6.5.5.
AdvancedOptimisedDecoderFlag	If true, signals that configuration for advanced optimised decoders is present in this decoderInit.
AdditionalSchemaFlag	If true, signals that configuration for additional schemas is present in this decoderInit.

In subclause 7.4.2.2, add the following missing semantics:

in_AnyElementDecoding	If true, signals that the element is decoded as part of AnyElementDecoding procedure (see 7.5.2.5.5).