POLSKI KOMITET NORMALIZACYJNY

POPRAWKA do POLSKIEJ NORMY

ICS 35.040.40

PN-ISO/IEC 15938-1:2005/AC1

Wprowadza
ISO/IEC 15938-1:2002/AC1:2004, IDT

Technika informatyczna

Interfejs opisu zawartości multimediów

Część 1: Systemy

Poprawka do Normy Międzynarodowej ISO/IEC 15938-1:2002/AC1:2004 *Information technology – Multimedia content description interface – Part 1: Systems* ma status Poprawki do Polskiej Normy

 nr ref. PN-ISO/IEC 15938-1:2005/AC1:2019-04

Przedmowa krajowa

Niniejsza poprawka została zatwierdzona przez Prezesa PKN dnia 9 kwietnia 2019 r.

Komitetem krajowym odpowiedzialnym za poprawkę jest KT nr 288 ds. Multimediów.

Istnieje możliwość przetłumaczenia poprawki na język polski na wniosek zainteresowanych środowisk. Decyzję podejmuje właściwy Komitet Techniczny.

W sprawach merytorycznych dotyczących treści normy można zwracać się do właściwego Komitetu Technicznego lub właściwej Rady Sektorowej PKN, kontakt: www.pkn.pl

Nota uznaniowa

Poprawka do Normy Międzynarodowej ISO/IEC 15938-1:2002/AC1:2004 została uznana przez PKN za Poprawkę do Polskiej Normy PN-ISO/IEC 15938-1:2005/AC1:2019-04.

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

Published 2004-05-15

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

Information technology — Multimedia content description interface —

Part 1: Systems

TECHNICAL CORRIGENDUM 1

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

Partie 1: Systèmes

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 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*.

In subclause 3.2, add the following definition at the appropriate alphabetical place:

deferred node

A node which is present in the description tree at encoder side and for which the following is true:

No part of that node has been sent to the decoder but the existence of that node has been signalled to the decoder.

ICS 35.040 **Ref. No. ISO/IEC 15938-1:2002/Cor.1:2004(E)**

Published in Switzerland

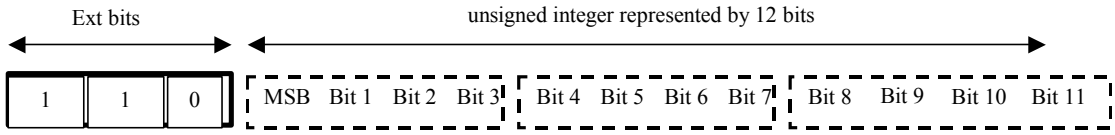
In 4.2.1, replace the formula for ceil(..):

ceil(..) $$ceil(x) = \begin{cases} \text{int}(x)+1 & x \geq 0 \\ \text{int}(x) & x < 0 \end{cases}$$

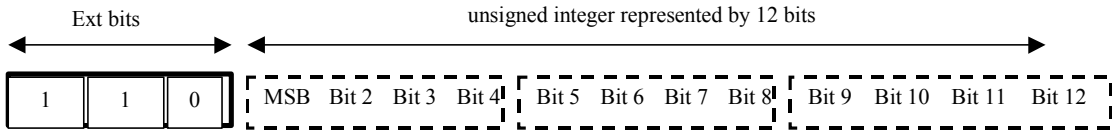
with:

Ceil (x) denotes the smallest integer larger than or equal to x.

In subclause 4.3, replace Figure 2:



with:



In subclause 5.6.4, add the following new paragraph:

BiM allows encoding descriptions for which there exists a schema that imports all necessary schemas. This main schema is identified by its first position in the table of schemas.

Note – If such schema does not exist a priori it can be easily designed for this purpose.

In subclause 6.1, replace:

6.1 Overview

The following subclauses specify the syntax elements and associated semantics of the textual format for ISO/IEC 15938 descriptions, abbreviated TeM. The textual DecoderInit (6.2), the textual AccessUnit (6.3), and the textual fragment update unit (6.4), with its constituent parts: the textual fragment update command (6.5), textual fragment update context (6.6) and textual fragment update payload (6.7).

with:

6.1 Overview

6.1.1 Schema Components

The clause 6 specifies the syntax elements and associated semantics of the textual format for ISO/IEC 15938 descriptions, abbreviated TeM. The main schema wrapper is specified in the following subclauses (6.1.2 and 6.1.3). The textual DecoderInit (6.2), the textual AccessUnit (6.3), and the textual fragment update unit (6.4), with its constituent parts: the textual fragment update command (6.5), textual fragment update context (6.6) and textual fragment update payload (6.7).

In subclause 6.1, replace:

The following schema wrapper shall be applied to the syntax defined in Clause 6.

```
<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>
```

with:

6.1.2 Schema wrapper syntax

The following schema wrapper shall be applied to the syntax defined in Clause 6.

```
<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">

      <element name=”DecoderInit” type=”mpeg7s:DecoderInitType”/>
      <element name=”AccessUnit” type=”mpeg7s:AccessUnitType”/>

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

</schema>
```

6.1.3 Schema wrapper semantics

Name	*Definition*
DecoderInit	A global element which carries decoder init information.
AccessUnit	A global element which carries an access unit.

In subclause 6.2.1, replace:

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

<complexType name="mpeg7s:DecoderInitType">
      <sequence>
            <element   name="SchemaReference"   type="mpeg7s:SchemaReferenceType"
                        maxOccurs="unbounded"/>
            <element    name="InitialDescription"    type="mpeg7s:AccessUnitType"
                        minOccurs=”0”/>
      </sequence>
      <attribute        name="systemsProfileLevelIndication"        type="decimal"
                        use="required"/>
</complexType>

<complexType name="SchemaReferenceType">
      <attribute name="name" type="anyURI" use="required"/>
      <attribute name="locationHint" type="anyURI" use="optional"/>
</complexType>
```

with:

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

<complexType name="DecoderInitType">
<sequence>
<element          name="SchemaReference"          type="mpeg7s:SchemaReferenceType"
                            maxOccurs="unbounded"/>
<element           name="InitialDescription"           type="mpeg7s:AccessUnitType"
                            minOccurs="0"/>
</sequence>
<attribute           name="systemsProfileLevelIndication"           type="decimal"
                            use="required"/>
</complexType>

<complexType name="SchemaReferenceType">
<attribute name="name" type="anyURI" use="required"/>
<attribute name="locationHint" type="anyURI" use="optional"/>
</complexType>
```

Add the following paragraph to subclause 6.2.2:

After processing the DecoderInit element the current context is set to the topmost node of the current description tree.

Add the following paragraph to subclause 6.5.2:

After processing a reset command the current context is set to the topmost node of the current description tree.

In subclause 6.5.2, add:

Note The `replaceNode` command can replace the context node by a node with a different name as well as by a node with the same name. In particular, if the node conveyed within the `FUPayload` has a different name as the context node specified in the context path, the context node specified in the context path is replaced in the current description tree by the node with the different name conveyed within the `FUPayload`.

In subclause 6.5.2, replace the semantics definition for addNode:

Adds the node conveyed within the FUPayload to the context node as the last child of the context node.

with:

Adds the node conveyed within the FUPayload.

In case of an element it adds the node conveyed within the FUPayload as the last child of the context node.

In case of an attribute it adds the node conveyed within the FUPayload to the context node.

Adding an attribute that already exists in the current description tree is forbidden.

In subclause 6.6.1 replace:

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

<simpleType name="FUContextType">
      <restriction base="string">
            <pattern value =

"/?((\.|(\.\.)|(((\i\c*:)?\i\c*)(\[\d+\])?))(/((\.)|(\.\.)|(((\i\c*:)?\i\c*)(\[\d
+\])?)))*(/@(\i\c*:)?\i\c*)?)|(@(\i\c*:)?\i\c*)"

            />

      </restriction>
</simpleType>
```

with:

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

<simpleType name="FragmentUpdateContextType">
      <restriction base="string">
            <pattern value =

"/?((\.|(\.\.)|(((\i\c*:)?\i\c*)(\[\d+\])?))(/((\.)|(\.\.)|(((\i\c*:)?\i\c*)(\[\d
+\])?)))*(/@(\i\c*:)?\i\c*)?)|(@(\i\c*:)?\i\c*)"

            />

      </restriction>
</simpleType>
```

In subclause 6.6.3.delete the last example:

```
<FUContext>/mpeg7:MPEG7[1]/@mpeg7:type</FUContext>
```

In subclause 6.7.1, replace:

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

<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>
```

with:

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

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

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

Add the following to the table in subclause 6.7.2:

	A node is no longer deferred if it is deleted, replaced, or nodes (deferred or not) or attributes are added to it.

Add the following entry to the table in subclause 6.7.2:

anyAttribute	The anyAttribute is used to convey the name and value of the attribute that is to be added or is replacing an existing attribute, respectively in the case of addNode and replaceNode commands.

Add the following informative examples as new subclause "6.7.3 Examples":

6.7.3 Examples

The following example illustrates how an attribute is added, replaced and deleted.

```
<DecoderInit targetNamespace="urn:mpeg:mpeg7:systems:2001"
             systemsProfileLevelIndication="0">
    <SchemaReference name="urn:mpeg:mpeg7:systems:2001"/>
<DecoderInit/>

<AccessUnit targetNamespace="urn:mpeg:mpeg7:systems:2001">
    <FragmentUpdateUnit>
        <FUCommand>addNode</FUCommand>
        <FUContext>./Score</FUContext>
        <FUPayload idref="ID_0"/>
    </FragmentUpdateUnit>
<AccessUnit/>

<AccessUnit targetNamespace="urn:mpeg:mpeg7:systems:2001">
    <FragmentUpdateUnit>
        <FUCommand>replaceNode</FUCommand>
        <FUContext>./Score/@idref</FUContext>
        <FUPayload xpath="./Score"/>
    </FragmentUpdateUnit>
```

```
    <FragmentUpdateUnit>
        <FUCommand>deleteNode</FUCommand>
        <FUContext>./Score/@xpath</FUContext>
    </FragmentUpdateUnit>
<AccessUnit/>
```

Append in InitialDescription semantics in subclause 7.2.3:

After processing the DecoderInit element the current context is set to the topmost node of the current description tree.

In subclause 7.2.3, add the following definition:

TypeCodecURI_Length[k][i]	Indicates the size in bytes of the `TypeCodecURI[k][i]`. A value of zero is forbidden.

In subclause 7.2.3, replace the following sentence:

— The first fragment update unit within the initial description shall use an

"absolute addressing mode", i.e. `CommandModeCode` equal to '001' or '110'.

with:

— The first fragment update unit within the initial description shall use an

"absolute addressing mode", i.e. `ContextModeCode` equal to '001' or '011'.

In subclause 7.2.3, delete the following sentence in the semantics definition for SchemaURI[k]:

Thus SchemaURI[0] identifies the targetNamespace of the description.

Add the following sentence before the last sentence of subclause 7.3.1:

Multiple fragment update units in an access unit are ordered and shall be processed by the terminal such that the result of applying the commands is equivalent to having executed them sequentially by the description composer in the order specified within the access unit.

In subclause 7.5, add the following sentence to the definition of the reset command:

After processing a reset command the current context is set to the topmost node of the current description tree.

In subclause 7.6.5.2.3, replace:

— The SBCs are assigned sequentially to the global elements defined in the schema referred by the `SchemaID`. Lexicographical ordering is performed before the assignment."

with:

— The SBCs are assigned sequentially to the global elements defined in the schema referred by the SchemaID. Lexicographical ordering on the expanded name is performed before the assignment.

In subclause 7.6.5.5.1, replace the second sentence:

It is present only if multiple occurrences are possible for the element referenced by the SBC or for any model group declared in the corresponding complexType definition.

with:

It is present only if multiple occurrences are possible for the element referenced by the SBC or for any model group declared in the corresponding complexType definition or if the content model of the corresponding complexType definition is an ALL group.

In subclause 7.6.5.5.2, replace the first sentence:

A SPC is used, if a Position Code is present according to 7.6.5.5.1. and if the corresponding complexType does not contain model groups with maxOccurs > 1.

with:

A SPC is used, if a Position Code is present according to 7.6.5.5.1. and if the corresponding complexType does not contain model groups with maxOccurs > 1 and if the content model of the corresponding complexType definition is not an ALL group.

In subclause 7.6.5.5.3, replace the first sentence:

In case of complexTypes with complex content which contain model groups with maxOccurs > 1, the positions of all nodes representing child elements declared in this complexType are encoded using the MPC.

with:

In case of complexTypes with complex content which contain model groups with maxOccurs > 1 or which contain an ALL group, the positions of all nodes representing child elements declared in this complexType are encoded using the MPC.

In subclause 7.6.5.5.3, replace the equation and text:

if m_{all}= 'unbounded'

MPA_{all} = 'unbounded'

else

$MPA_{all} = m_{all} * \max (MPA_i)$

where

"MPA_i" is equal to the maximum number of elements that the ith children particle of the all can instantiate

"m_{all}" is equal to the 'max occurs' property of the all particle

with:

if m_{all}= 'unbounded'

MPA_{all} = 'unbounded'

else

$MPA_{all} = m_{all} * number\_of\_children$

where

"number_of_children $_i$" is equal to the number of children of the all particle

"m_{all}" is equal to the 'max occurs' property of the all particle

In subclause 8.4.5.1, replace the table:

SchemaComponent PayloadTypeCode(SchemaComponent defaultType, boolean multi) {	**Number of bits**	**Mnemonic**
if (multi) {		
PayloadTypeIdentificationCode	ceil(log2(number of possible subtypes of defaultType + sizeIncrease))	bslbf
effectiveType = getDerivedType(defaultType, PayloadTypeIdentificationCode)		
}else if ((hasTypeCasting && hasNamedSubtypes(defaultType)) || hasDeferredNodes || elementNillable()) {		
PayloadTypeCastFlag	1	bslbf
if (PayloadTypeCastFlag == 1) {		
PayloadTypeIdentificationCode	ceil(log2(number of possible subtypes of defaultType + sizeIncrease))	bslbf
effectiveType = getDerivedType(defaultType, PayloadTypeIdentificationCode)		
}		
} else {		
effectiveType = defaultType		
}		
return effectiveType		
}		

with:

SchemaComponent PayloadTypeCode(SchemaComponent defaultType, boolean multi) {	**Number of bits**	**Mnemonic**
if (multi) {		
If(firstElementContentChunk()) {		
PayloadTypeCastFlag	1	bslbf
if (PayloadTypeCastFlag == 1) {		
PayloadTypeIdentificationCode	ceil(log2(number of possible subtypes of defaultType + sizeIncrease))	bslbf
effectiveType = getDerivedType(defaultType, PayloadTypeIdentificationCode)		
}		
} else		
PayloadTypeIdentificationCode	ceil(log2(number of possible subtypes of defaultType + sizeIncrease))	bslbf
effectiveType = getDerivedType(defaultType, PayloadTypeIdentificationCode)		
}		
}else if ((hasTypeCasting && hasNamedSubtypes(defaultType)) || hasDeferredNodes || elementNillable()) {		
PayloadTypeCastFlag	1	bslbf
if (PayloadTypeCastFlag == 1) {		
PayloadTypeIdentificationCode	ceil(log2(number of possible subtypes of defaultType + sizeIncrease))	bslbf
effectiveType = getDerivedType(defaultType, PayloadTypeIdentificationCode)		
}		
} else {		
effectiveType = defaultType		
}		
return effectiveType		
}		

In subclause 8.4.7.2, replace in the table the first two rows:

useOptimisedDecoder()	Returns "true" if the type `effectiveType` is associated to an optimised type decoder as conveyed in the `DecoderInit` (refer to subclause 7.2).
optimisedDecoder ()	Triggers the optimised type decoder associated to the type `effectiveType` as conveyed in the `DecoderInit` (refer to subclause 7.2).

with:

useOptimisedDecoder()	Returns "true" if the type `theType` is associated to an optimised type decoder as conveyed in the `DecoderInit` (refer to subclause 7.2).
optimisedDecoder ()	Triggers the optimised type decoder associated to the type `theType` as conveyed in the `DecoderInit` (refer to subclause 7.2)

In subclause 8.5.2.2.2, replace:

— If the type is derived by restriction from another type, the effective content particle of the type is equals to its content,

with:

— If the type is derived by restriction from another type, the effective content particle of the type is equal to its content,

In subclause 8.5.2.2.5.2 Occurrence node automaton, replace case b:

— case b: if minOccurs = 0, maxOccurs = 1

  — two states are added to the child node automaton : a new start state and a new final state,

  — a "Shunt transition" is added between the new start state and the new final state,

  — a "Code transition" is added between the new start state and the old one, its signature is equal to the signature of the child node of this occurrence node,

  — a simple transition is added between the old final state and the new one.

with:

— case b: if minOccurs = 0,

  — maxOccurs = 1

    — two states are added to the child node automaton : a new start state and a new final state,

    — a "Shunt transition" is added between the new start state and the new final state,

    — a "Code transition" is added between the new start state and the old one, its signature is equal to the signature of the child node of this occurrence node,

    — a simple transition is added between the old final state and the new one.

  — maxOccurs = 0

    — two states are added to the child node automaton : a new start state and a new final state,

    — a simple transition is added between the new start state and the new final state,

Note – In case of maxOccurs = 0, the transformation results in discarding the child automaton.

In subclause 8.5.3, add the following new point after the second bullet point:

— All the attribute references and attribute group references are replaced by their referenced definition