

---

---

**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*

**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.



**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2008

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](mailto:copyright@iso.org)  
Web [www.iso.org](http://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*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 23001-1:2006/AMD2:2008

# 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 () {                     | <b>Number of bits</b> | <b>Mnemonic</b> |
| <b>SystemsProfileLevelIndication</b> | 8+                    | vluimsbf8       |
| ...                                  |                       |                 |

with:

|                                                     |                           |                 |
|-----------------------------------------------------|---------------------------|-----------------|
| DecoderInit () {                                    | <b>Number of bits</b>     | <b>Mnemonic</b> |
| <b>SystemsProfileLevelIndication</b>                | 8+                        | vluimsbf8       |
| <b>UnitSizeCode</b>                                 | 3                         | bslbf           |
| <b>NoAdvancedFeatures</b>                           | 1                         | bslbf           |
| <b>ReservedBits</b>                                 | 4                         | bslbf           |
| If (! NoAdvancedFeatures) {                         |                           |                 |
| <b>AdvancedFeatureFlags_Length</b>                  | 8+                        | vluimsbf8       |
| /** FeatureFlags **/                                |                           |                 |
| <b>InsertFlag</b>                                   | 1                         | bslbf           |
| <b>AdvancedOptimisedDecodersFlag</b>                | 1                         | bslbf           |
| <b>AdditionalSchemaFlag</b>                         | 1                         | bslbf           |
| <b>AdditionalSchemaUpdatesOnlyFlag</b>              | 1                         | bslbf           |
| <b>FragmentReferenceFlag</b>                        | 1                         | bslbf           |
| <b>MPCOnlyFlag</b>                                  | 1                         | bslbf           |
| <b>HierarchyBasedSubstitutionCodingFlag</b>         | 1                         | bslbf           |
| <b>ContextPathTableFlag</b>                         | 1                         | bslbf           |
| <b>PrefixTableFlag</b>                              | 1                         | bslbf           |
| <b>ReservedBitsZero</b> /**for additional Flags **/ | 24                        | bslbf           |
| If (PrefixTableFlag) {                              |                           |                 |
| PrefixTable ()                                      |                           |                 |
| }                                                   |                           |                 |
| <b>ReservedBitsZero</b>                             | 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* AdvancedOptimisedDecoderTypeURI_Length[i] | bslbf     |
| }                                                          |                                              |           |
| AdvancedOptimisedDecodersConfig ()                         |                                              |           |
| }                                                          |                                              |           |
| /** FUUConfig - Fragment reference framework **/           |                                              |           |
| If (FragmentReferenceFlag) {                               |                                              |           |
| NumOfSupportedFragmentReferenceFormat                      | 8                                            | uimsbf    |
| for (i=0; i< NumOfSupportedFragmentReferenceFormat; i++) { |                                              |           |
| SupportedFragmentReferenceFormat[i]                        | 8                                            | bslbf     |
| }                                                          |                                              |           |

|                                        |    |           |
|----------------------------------------|----|-----------|
| }                                      |    |           |
| }                                      |    |           |
| /** end FUUConfig **/                  |    |           |
| If (AdditionalSchemaFlag) {            |    |           |
| AdditionalSchemaConfig ()              |    |           |
| }                                      |    |           |
| /** Initial document **/               |    |           |
| If (!AdditionalSchemaUpdateOnlyFlag) { |    |           |
| InitialDocument_Length                 | 8+ | vluimsbf8 |
| InitialDocument()                      |    |           |
| }                                      |    |           |
| }                                      |    |           |

Add the following paragraph:

|                                       |                               |           |
|---------------------------------------|-------------------------------|-----------|
| PrefixTable {                         |                               |           |
| <b>NumberOfPrefixes</b>               | 8+                            | vluimsbf8 |
| for (k=0; k< NumberOfPrefixes; k++) { |                               |           |
| <b>SchemaURI_Length[k]</b>            | 8+                            | vluimsbf8 |
| <b>SchemaURI[k]</b>                   | 8*<br>SchemaURI_Lengt<br>h[k] | bslbf     |
| <b>Prefix_Length[k]</b>               | 8+                            | vluimsbf8 |
| <b>Prefix[k]</b>                      | 8*<br>[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 unprefix 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**

| <i>Index</i> | <i>Systems Profile and Level</i> |
|--------------|----------------------------------|
| 0            | no profile specified             |
| 1 – 127      | Reserved for ISO Use             |

with:

**Table 1 — Index Table for SystemsProfileLevelIndication**

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

In 7.5.2.4.5.2 (AnyElementDecoding Syntax), replace:

| AnyElementDecoding() {                  | Number of bits                                                                   | Mnemonic |
|-----------------------------------------|----------------------------------------------------------------------------------|----------|
| <b>GlobalElementSchemaID</b>            | $\text{ceil}(\log_2(\text{NumberOfSchemas} + \text{NumberOfAdditionalSchemas}))$ | uimsbf   |
| <b>AnyElement_Length</b>                | 5+                                                                               | vluimsbf |
| <b>Any_SBC_Operand_Selector</b>         | 5+                                                                               | vluimsbf |
| If (inPayloadDecoding()) {              |                                                                                  |          |
| Element(ChildrenSchemaMode, theAnyType) |                                                                                  |          |
| }                                       |                                                                                  |          |
| }                                       |                                                                                  |          |

with:

|                                             |                                                           |          |
|---------------------------------------------|-----------------------------------------------------------|----------|
| AnyElementDecoding() {                      | Number of bits                                            | Mnemonic |
| <b>GlobalElementSchemaID</b>                | ceil( log2(NumberOfSchemas + NumberOfAdditionalSchemas) ) | uimsbf   |
| <b>AnyElement_Length</b>                    | 5+                                                        | vluimsbf |
| if (GlobalElementSchemaID == LAXSchemaID) { |                                                           |          |
| //lax encoding                              |                                                           |          |
| AnyElementLaxDecoding();                    |                                                           |          |
| } else {                                    |                                                           |          |
| <b>Any_SBC_Operand_Selector</b>             | 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 <code>SchemaURI</code> array defined in 6.2 (optionally extended with the list of additional schemas).<br><br>If <code>GlobalElementSchemaID</code> is equal to the ID of the virtual lax schema <code>urn:mpeg:mpegb:bim:laxencoding:2007</code> specified in the <code>decoderInit</code> (i.e., <code>GlobalElementSchemaID == LAXSchemaID</code> ), the lax decoding procedure <code>AnyElementLaxDecoding()</code> is called. |
| AnyElement_Length                    | Indicates the length in bits of the remainder of this <code>AnyElementDecoding</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Any_SBC_Operand_Selector             | Selects one global element of the schema referenced by <code>GlobalElementSchemaID</code> using the <code>OperandTBC</code> table for <code>Extended_SBC_Operand_Selector</code> as specified in 6.6.5.2.3. Therefore, <code>Any_SBC_Operand_Selector</code> is equivalent to the <code>Extended_SBC_Operand_Selector</code> but with a bit representation in <code>vlumbsbf5</code> .                                                                                                                                         |
| <code>inPayloadDecoding()</code>     | Returns true if the <code>AnyElementDecoding</code> procedure has been triggered from a payload decoding procedure.                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>Element()</code>               | See 7.4.1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>theAnyType</code>              | The type of the element identified by the <code>SBC_GlobalElement_SelectorCode</code> as defined in the schema identified by the <code>GlobalElementSchemaID</code> .                                                                                                                                                                                                                                                                                                                                                          |
| <code>AnyElementLaxDecoding()</code> | 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

| <code>AnyElementLaxDecoding()</code> {      | Number of bits | Mnemonic |
|---------------------------------------------|----------------|----------|
| <b>ReservedBits</b>                         | 4              | bmsbf    |
| <b>AnyElementLaxCompressionScheme</b>       | 4              | bmsbf    |
| <code>AnyElementLaxDecodingContent()</code> |                |          |
| }                                           |                |          |

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

Table AMD2.1 — Compression methods

| <b><i>AnyElementLaxCompressionScheme</i></b> | <b><i>Definition</i></b>                      |
|----------------------------------------------|-----------------------------------------------|
| 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.

Table AMD2.2 — Event table

| Event name           | UID        | Parameters                       | Semantics                                                |
|----------------------|------------|----------------------------------|----------------------------------------------------------|
| <i>BiM Encoding</i>  | 0          | ∅                                | Fragment update payload syntax defined in subclause 7.3. |
| ADD_NS               | 1          | <i>ns</i>                        | Adds a new namespace in the namespace table.             |
| ADD_ENAME            | 2          | <i>nsid, elt</i>                 | Adds a new element in the element table.                 |
| ADD_ANAME            | 3          | <i>nsid, att</i>                 | Adds a new attribute in the attribute table.             |
| END_ELEMENT          | 4          | ∅                                | SAX — ContentHandler callback                            |
| START_PFX_MAPPING    | 5          | <i>nsid, pfx</i>                 | SAX — ContentHandler callback                            |
| CHARACTERS           | 6          | <i>value</i>                     | SAX — ContentHandler callback                            |
| END_DOCUMENT         | 7          | ∅                                | SAX — ContentHandler callback                            |
| PI                   | 8          | <i>target, data</i>              | SAX — ContentHandler callback                            |
| SKIPPED_ENTITY       | 9          | <i>name</i>                      | SAX — ContentHandler callback                            |
| NOTATION_DECL        | 10         | <i>name, pid, sid</i>            | SAX — DTDHandler callback                                |
| UNPARSED_ENTITY_DECL | 11         | <i>name, pid, sid, not</i>       | SAX — DTDHandler callback                                |
| ATTRIBUTE_DECL       | 12         | <i>el, at, type, mode, value</i> | SAX — DeclHandler callback                               |
| ELEMENT_DECL         | 13         | <i>name, model</i>               | SAX — DeclHandler callback                               |
| EXT_ENTITY_DECL      | 14         | <i>name, pid, sid</i>            | SAX — DeclHandler callback                               |
| INT_ENTITY_DECL      | 15         | <i>name, value</i>               | SAX — DeclHandler callback                               |
| COMMENT              | 16         | <i>value</i>                     | SAX — LexicalHandler callback                            |
| START_CDATA          | 17         | ∅                                | SAX — LexicalHandler callback                            |
| END_CDATA            | 18         | ∅                                | SAX — LexicalHandler callback                            |
| START_DTD            | 19         | <i>name, pid, sid</i>            | SAX — LexicalHandler callback                            |
| END_DTD              | 20         | ∅                                | SAX — LexicalHandler callback                            |
| START_ENTITY         | 21         | <i>name</i>                      | SAX — LexicalHandler callback                            |
| END_ENTITY           | 22         | <i>name</i>                      | SAX — LexicalHandler callback                            |
| <i>Reserved</i>      | 23-24      | —                                | <i>Reserved</i>                                          |
| START_ELEMENT_elfid  | 23+2*elfid | ∅                                | SAX — ContentHandler callback                            |
| ATTRIBUTE_attid      | 24+2*attid | <i>value</i>                     | SAX — ContentHandler callback                            |

Note that there is no START\_ELEMENT generic event. Instead, the SAX START\_ELEMENT event is split into specific ATTRIBUTE\_#att events and a specific START\_ELEMENT\_#elt event. Therefore, an infinite number of ATTRIBUTE\_#att events and START\_ELEMENT\_#elt virtually belongs to the table of events.

#### 7.5.2.4.5.4.2.2 Syntax

|                                       |                       |                 |
|---------------------------------------|-----------------------|-----------------|
| AnyElementLaxDecodingContent() {      | <b>Number of bits</b> | <b>Mnemonic</b> |
| <b>ReservedBits</b>                   | 6                     | bmsbf           |
| <b>Agglutinated</b>                   | 1                     | bmsbf           |
| <b>zlib</b>                           | 1                     | bmsbf           |
| do {                                  |                       |                 |
| event = readEvent();                  |                       |                 |
| switch (event) {                      |                       |                 |
| case ADD_NS:                          |                       |                 |
| <b>NamespaceName</b>                  | 8+                    | string          |
| addNamespace(NamespaceName);          |                       |                 |
| break;                                |                       |                 |
| case ADD_ENAME:                       |                       |                 |
| <b>NSID</b>                           | 8                     | uimsbf8         |
| <b>ElementName</b>                    | 8+                    | string          |
| addElement(NSID,ElementName);         |                       |                 |
| break;                                |                       |                 |
| case ADD_ANAME:                       |                       |                 |
| <b>NSID</b>                           | 8                     | uimsbf8         |
| <b>AttributeName</b>                  | 8+                    | string          |
| addAttribute(NSID,AttributeName);     |                       |                 |
| break;                                |                       |                 |
| case END_ELEMENT:                     |                       |                 |
| sax_endElement();                     |                       |                 |
| break;                                |                       |                 |
| case START_PFX_MAPPING:               |                       |                 |
| <b>NSID</b>                           | 8                     | uimsbf8         |
| <b>PfxName</b>                        | 8+                    | string          |
| sax_startPrefixMapping(NSID,PfxName); |                       |                 |
| break;                                |                       |                 |
| case CHARACTERS:                      |                       |                 |
| <b>Value</b>                          | 8+                    | string          |
| sax_characters(Value);                |                       |                 |
| break;                                |                       |                 |
| case END_DOCUMENT:                    |                       |                 |
| sax_endDocument ();                   |                       |                 |
| break;                                |                       |                 |
| case PI:                              |                       |                 |
| <b>Target</b>                         | 8+                    | string          |

|                                                                  |    |        |
|------------------------------------------------------------------|----|--------|
| <b>Data</b>                                                      | 8+ | string |
| sax_processingInstruction(Target,Data);                          |    |        |
| break;                                                           |    |        |
| case SKIPPED_ENTITY:                                             |    |        |
| <b>Name</b>                                                      | 8+ | string |
| sax_skippedEntity(Name);                                         |    |        |
| break;                                                           |    |        |
| case NOTATION_DECL:                                              |    |        |
| <b>Name</b>                                                      | 8+ | string |
| <b>PublicID</b>                                                  | 8+ | string |
| <b>SystemID</b>                                                  | 8+ | string |
| sax_notationDecl(Name,PublicID,<br>SystemID);                    |    |        |
| break;                                                           |    |        |
| case UNPARSED_ENTITY_DECL:                                       |    |        |
| <b>Name</b>                                                      | 8+ | string |
| <b>PublicID</b>                                                  | 8+ | string |
| <b>SystemID</b>                                                  | 8+ | string |
| <b>NotationName</b>                                              | 8+ | string |
| sax_unparsedEntityDecl(Name,<br>PublicID,SystemID,NotationName); |    |        |
| break;                                                           |    |        |
| case ATTRIBUTE_DECL:                                             |    |        |
| <b>Ename</b>                                                     | 8+ | string |
| <b>Aname</b>                                                     | 8+ | string |
| <b>Type</b>                                                      | 8+ | string |
| <b>Mode</b>                                                      | 8+ | string |
| <b>Value</b>                                                     | 8+ | string |
| sax_attributeDecl(Ename,Aname,Type,<br>Mode,Value);              |    |        |
| break;                                                           |    |        |
| case ELEMENT_DECL:                                               |    |        |
| <b>Name</b>                                                      | 8+ | string |
| <b>Model</b>                                                     | 8+ | string |
| sax_elementDecl(Name,Model);                                     |    |        |
| break;                                                           |    |        |
| case EXT_ENTITY_DECL:                                            |    |        |
| <b>Name</b>                                                      | 8+ | string |
| <b>PublicID</b>                                                  | 8+ | string |
| <b>SystemID</b>                                                  | 8+ | string |
| sax_extEntityDecl(Name,PublicID,<br>SystemID);                   |    |        |
| break;                                                           |    |        |

|                                         |    |        |
|-----------------------------------------|----|--------|
| case INT_ENTITY_DECL:                   |    |        |
| <b>Name</b>                             | 8+ | string |
| <b>PublicID</b>                         | 8+ | string |
| sax_intEntityDecl(Name, PublicID);      |    |        |
| break;                                  |    |        |
| case COMMENT:                           |    |        |
| <b>Value</b>                            | 8+ | string |
| sax_commentl(Value);                    |    |        |
| break;                                  |    |        |
| case START_CDATA:                       |    |        |
| sax_startCDATA();                       |    |        |
| break;                                  |    |        |
| case END_CDATA:                         |    |        |
| sax_endCDATA();                         |    |        |
| break;                                  |    |        |
| case START_DTD:                         |    |        |
| <b>Name</b>                             | 8+ | string |
| <b>PublicID</b>                         | 8+ | string |
| <b>SystemID</b>                         | 8+ | string |
| sax_startDTD(Name, PublicID, PublicID); |    |        |
| break;                                  |    |        |
| case END_DTD:                           |    |        |
| sax_endDTD();                           |    |        |
| break;                                  |    |        |
| case START_ENTITY:                      |    |        |
| <b>Name</b>                             | 8+ | string |
| sax_startEntity(Name);                  |    |        |
| break;                                  |    |        |
| case END_ENTITY:                        |    |        |
| <b>Name</b>                             | 8+ | string |
| sax_endEntity(Name);                    |    |        |
| break;                                  |    |        |
| default:                                |    |        |
| if (event % 2 == 1) {                   |    |        |
| eltid = ( event – END_ENTITY – 1 )/2;   |    |        |
| sax_startElement(eltid,attrs);          |    |        |
| attrs=NULL;                             |    |        |
| } else {                                |    |        |
| <b>AttributeValue</b>                   | 8+ | string |
| atid = ( event – END_ENTITY – 2 )/2;    |    |        |
| attrs.newAttribute(atid,value);         |    |        |
| }                                       |    |        |

|                                  |  |  |
|----------------------------------|--|--|
| break;                           |  |  |
| }                                |  |  |
| } while (event != END_DOCUMENT); |  |  |
| }                                |  |  |

#### 7.5.2.4.5.4.2.3 Semantics

| Name               | Definition                                                                                                                                                                           |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReservedBits       | Reserved bits for future use.                                                                                                                                                        |
| Agglutinated       | A value of 1 indicates the agglutination of events has been activated at encoding time. See 7.5.2.4.5.4.3.1.                                                                         |
| zlib               | A value of 1 indicates that the sequence of symbols is further compressed with zlib.                                                                                                 |
| readEvent          | Returns the next event. See 7.5.2.4.5.4.2.                                                                                                                                           |
| addNamespace       | Adds a new namespace, which is a string, at the end of the namespace table. The index of a namespace in this table is referred as the namespace ID.                                  |
| addElement         | Adds a new element, which is a doublet of a namespace ID and a string, at the end of the element table. The index of an element in this table is referred as the element ID.         |
| addAttribute       | Adds a new attribute, which is a doublet of a namespace ID and a string, at the end of the attribute table. The index of an attribute in this table is referred as the attribute ID. |
| sax_X              | Triggers a SAX callback X with the usual SAX parameters. String parameters can be recovered from ID ones using the namespace, elements and attributes dictionaries.                  |
| attrs              | Set of attributes in a given element.                                                                                                                                                |
| attrs.newAttribute | Adds a new attribute in the current set of attributes, which will be used in the next sax_startElement SAX event.                                                                    |

#### 7.5.2.4.5.4.3 readEvent

##### 7.5.2.4.5.4.3.1 Overview

The `readEvent` function returns the next event to be triggered by the decoder. It relies on a buffered event model, based on an events FIFO buffer of size 3; therefore, the function can return an event without necessarily requiring reading a byte in the bitstream.

A sequence of up to three events can be agglutinated into a single new event. It allows more efficient encoding, especially when not using zlib.

An event or a sequence of up to three events can be encoded with a single byte. Events encoding is done with the help of a fixed-size symbol table, which contains 128 entries. This table maps a set of events or sequence of events to a set of integer values (the symbols) ranging from 0 to 127. The table is sorted in the "most recently seen" order.

After an event or a sequence of events has been decoded, the table of symbols is updated with the `updateModel`, `updateModelOne` and `updateSymbol` procedures. The sequence of the three latest seen events is moved (or added if doesn't still exist in the table) to the front of the table. The sequence of the two latest two decoded events is then moved to the front of the table. At last, the latest encoded event is moved to the front of the table. In case of addition of an event (or sequence of events), the oldest events or sequence of events are deleted from the table. By doing this, the most recent sequences of up to 3 events are kept in the table and need only 1 byte to be encoded. The symbol table is initialized with the first events from the table of events (Table AMD2.2), and the symbol of each event is set to its UID.

#### 7.5.2.4.5.4.3.2 Syntax

| <code>readEvent() {</code>                           | Number of bits | Mnemonic |
|------------------------------------------------------|----------------|----------|
| <code>if (events_fifo.empty()) {</code>              |                |          |
| <b>ExtendedCode</b>                                  | 1              | bmsbf    |
| <code>if (ExtendedCode) {</code>                     |                |          |
| <b>UID</b>                                           | 15             | uimsbf   |
| <code>events = [ UID ];</code>                       |                |          |
| <code>} else {</code>                                |                |          |
| <b>Code</b>                                          | 7              | uimsbf   |
| <code>events = SymbolCodeMap.getSymbol(Code);</code> |                |          |
| <code>}</code>                                       |                |          |
| <code>updateModel( events );</code>                  |                |          |
| <code>events_fifo.push( events );</code>             |                |          |
| <code>}</code>                                       |                |          |
| <code>return events_fifo.pull();</code>              |                |          |
| <code>}</code>                                       |                |          |

The `ExtendedCode` field allows decoding a single event which doesn't belong to the symbol table. In this case, the event is decoded by reading directly its UID. After an event or a sequence of events has been decoded, the symbol table is updated by `updateModel` procedure with a sequence of `events` (which can contain only one event).

| <code>updateModel(events) {</code>                    | Number of bits | Mnemonic |
|-------------------------------------------------------|----------------|----------|
| <code>for ( i=0 ; i &lt; events.size ; i++ ) {</code> |                |          |
| <code>if (Agglutinated) {</code>                      |                |          |
| <code>updateModelOne( events[i] );</code>             |                |          |
| <code>} else {</code>                                 |                |          |
| <code>updateSymbol(events[i]);</code>                 |                |          |
| <code>}</code>                                        |                |          |
| <code>}</code>                                        |                |          |
| <code>}</code>                                        |                |          |

|                                       |                |          |
|---------------------------------------|----------------|----------|
| updateModelOne(evt) {                 | Number of bits | Mnemonic |
| if (last_evt != null) {               |                |          |
| if (last2_evt != null) {              |                |          |
| updateSymbol(last2_evt@last_evt@evt); |                |          |
| }                                     |                |          |
| updateSymbol(last_evt@evt);           |                |          |
| }                                     |                |          |
| updateSymbol(evt);                    |                |          |
| last2_evt = last_evt;                 |                |          |
| last_evt = evt;                       |                |          |
| }                                     |                |          |

|                                     |                |          |
|-------------------------------------|----------------|----------|
| updateSymbol(symbol) {              | Number of bits | Mnemonic |
| if (SymbolTable.contains(symbol)) { |                |          |
| SymbolTable.remove(symbol);         |                |          |
| SymbolTable.moveToFront(symbol);    |                |          |
| } else {                            |                |          |
| oldsym = SymbolTable.removeLast();  |                |          |
| SymbolTable.moveToFront(symbol);    |                |          |
| code = SymbolCodeMap.get(oldsym);   |                |          |
| SymbolCodeMap.put(symbol, code);    |                |          |
| SymbolCodeMap.remove(oldsym);       |                |          |
| }                                   |                |          |
| }                                   |                |          |

**7.5.2.4.5.4.3.3 Semantics**

| <i>Name</i>             | <i>Definition</i>                                                                          |
|-------------------------|--------------------------------------------------------------------------------------------|
| events_fifo             | FIFO (First-In First-Out) events container.                                                |
| events_fifo.empty       | Returns true if the FIFO events container is empty and false otherwise.                    |
| SymbolCodeMap.getSymbol | Returns the symbol (the set of events) associated to a code.                               |
| updateModel             | Updates the Symbol Table with a set of new events.                                         |
| events_fifo.push        | Adds a new event or a set of new events (up to 3) at the end of the FIFO events container. |
| events_fifo.pull        | Pulls a single event from the beginning of the FIFO events container.                      |
| events.size             | Number of newly decoded events (=1,2, or 3).                                               |
| updateModelOne          | Updates the Symbol Table with a new incoming event.                                        |
| updateSymbol            | Updates a symbol in the Symbol Table.                                                      |
| SymbolTable             | Fixed-size table aimed to contain the symbols.                                             |
| SymbolTable.contains    | Returns true if the SymbolTable contains the given symbol and else otherwise.              |
| SymbolTable.moveToFront | Appends a new symbol at the beginning of the SymbolTable.                                  |
| SymbolTable.size        | Returns the current size of the SymbolTable.                                               |
| SymbolTable.removeFirst | Removes the first entry of the SymbolTable.                                                |
| SymbolTable.remove      | Removes a given symbol from the SymbolTable.                                               |
| SymbolCodeMap           | Container aimed to contain the mappings between a symbol and its corresponding code.       |
| SymbolCodeMap.put       | Adds a (symbol, code) mapping into the SymbolCodeMap.                                      |
| SymbolCodeMap.remove    | Removes a mapping form the SymbolCodeMap.                                                  |

**7.5.2.4.5.4.3.4 Encoding example (informative)**

Below is an *example* of the encoding algorithm with the agglutination process. Suppose that we want to encode the following XML document:

```
<a><b></b><b></b>
```

which is equivalent to encoding the following sequence of events:

```
START_ELT_1, START_ELT_2, END_ELT, START_ELT_2, END_ELT
```

Below is a representation of the initial table of symbols (the IDLE event is a non-existent event).

| Events             | Symbol |
|--------------------|--------|
| IDLE               | 0      |
| ADD_ANAME          | 1      |
| ADD_ENAME          | 2      |
| ADD_NS             | 3      |
| END_ELEMENT        | 4      |
| ...                | ...    |
| START_ELEMENT_1    | 25     |
| ...                | ...    |
| START_ATTRIBUTE_51 | 126    |
| START_ELEMENT_52   | 127    |

The START\_ELEMENT\_1 is first encoded with byte #25. The event is then moved to the front of the table, which leads to:

| Events             | Symbol |
|--------------------|--------|
| START_ELEMENT_1    | 25     |
| IDLE               | 0      |
| ADD_ANAME          | 1      |
| ADD_ENAME          | 2      |
| ADD_NS             | 3      |
| END_ELEMENT        | 4      |
| ...                | ...    |
| START_ATTRIBUTE_51 | 126    |
| START_ELEMENT_52   | 127    |

Then, the START\_ELEMENT\_2 is encoded with the byte #27. The double event [START\_ELEMENT\_1 ; START\_ELEMENT\_2] is then added to the front of the table, pulling out the START\_ELEMENT\_52 event. The event START\_ELEMENT\_2 is then moved up. This leads to: