INTERNATIONAL STANDARD

ISO/IEC 14496-20

Second edition
2008-12-01

AMENDMENT 3
2010-05-01

Information technology — Coding of audio-visual objects —

Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)

AMENDMENT 3: Presentation and Modification of Structured Information (PMSI)

Technologies de l'information — Codage des objets audiovisuels —

Partie 20: Représentation de scène d'application allégée (LASeR) et format d'agrégation simple (SAF)

AMENDEMENT 3: Présentation et modification de l'information structurée (PMSI)

Reference number
ISO/IEC 14496-20:2008/Amd.3:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 3 to ISO/IEC 14496-20:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information.*

Information technology — Coding of audio-visual objects —

Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)

AMENDMENT 3: Presentation and Modification of Structured Information (PMSI)

In Clause 2, add the following references:

W3C XPointer Framework, W3C Recommendation 25 March 2003. http://www.w3.org/TR/xptr-framework/

W3C XML Path Language (Xpath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath

In 3.2, add the following abbreviated term:

EBNF: Extended Backus-Naur Form

In Table 3. List of LASeR events, add the following item:

Event name	Namespace	Description	Bubble	Canc.
"xmlUpdate"	urn:mpeg:mpeg4:laser:2005	Informs the update of Structured Information referenced by the presentation engine.	No	No

After 6.5.6.4, add the following:

6.5.6.5 ExternalResourceEvent

```
Interface xmlUpdateEvent : LASeR Event {
     readonly attribute DOMString resourceURL;
     readonly attribute DOMString updatedValue;
};
```

No defined constants

Attributes

- **resourceURL**: This value identifies the updated external resource URL including the mpeg-pmsi() fragment
- **externalResource** : This value is a new value pointed by resource URL

No defined methods

Targets of this event must be elements allowing ***mpeg-pmsi()*** schemed URI.

After 6.8.60, add the following:

6.8.61 SVG1.1 tref

The ***SVG1.1 tref*** element is specified in 10.6 of [W3C SVG11]. The ***SVG1.1 tref*** element used in this standard is also allowed to be a child of the ***SVGT1.2 textArea*** element.

Additionally, the "editable" attribute defined in SVGT1.2 is allowed on the ***tref*** element. Editing the textual content of the ***tref*** element implies updating the original referenced data if possible.

6.8.62 LASeR externalReference

6.8.62.1 Semantics

The purpose of ***externalReference*** is to identify the scope of the scene to be updated regularly with the latest version of Structured Information at the specified interval of time.

The ***LASeR externalReference*** element is a container element indicating all elements included are updated periodically according to a provided ***updateInterval*** attribute. If this element does not include any external reference, ***updateInterval*** attribute is ignored. Thus, the ***updateInterval*** attribute is only used for URIs with ***mpeg-pmsi()*** scheme. Other URIs processing shall occur as usual regardless of the value of the ***updateInterval*** attribute. The ***externalReference*** element allows children.

6.8.62.2 Attributes

- **updateInterval**: is the updating interval of the elements/attributes within the enclosure of the element. This attribute is one of "none", "free", "<Clock-value>".
  - The value "none" indicates that Structured Information shall be evaluated only once.
  - The value "free" indicates the author does not mandate any specific behavior and it is up to the implementation to decide the interval of the update. This is the default value.
  - <Clock-value> indicates that if changes happened during that amount of time, they shall be reflected. This value must be greater than 0. The author must be responsible to assign a meaningful value to reflect the practicality on the implementation.
- **security**: controls the type of allowed updates in the referenced resource. This is defined in 6.8.53.
- **entry**: indicates the entry element of the Structured Information. Any children elements of the ***externalReference*** element using the ***mpeg-pmsi()*** scheme to reference external resources may use an Xpath expression relative to this entry element. In case of nested ***externalReference*** elements, the entry element is computed based on the parent entry element. If the ***entry*** attribute is not present, the entry element is not modified and if no parent ***entry*** attribute is present, the entry element is the root of the Structured Information. The value of this attribute shall only reference one element in the Structured Information.

6.8.63 LASeR externalUpdate

6.8.63.1 Semantics

The ***LASeR externalUpdate*** element supports a modification of Structured Information or elements|attributes thereof. It supports three types of modification: “replace”, “insert”, and “delete”. A part of Structured Information can be replaced with a new element or an attribute value, a new element or an attribute value is inserted into Structured Information, or a part of Structured Information can be deleted according to the provided ***type*** attribute.

This element shall not be placed as a child of ***LASeR externalReference*** element.

6.8.63.2 Attributes

- **xlink:href**: this attribute indicates the element/attribute which is replaced, inserted, or deleted using ***mpeg-pmsi()*** scheme.
- **type**: this attribute indicates the type for the modification of Structured Information.
    - The type “replace” indicates that an existing element or attribute value is replaced with a new element or attribute value.
    - The type “insert” indicates that a new element or attribute value is inserted into Structured Information.
    - The type “delete” indicates that an existing element or attribute value in Structured Information is deleted.
- **attributeName**: this attribute defines the name of attribute in which the insertion or replacement happens, by default “children”.
- **index**: this attribute defines the index at which to insert or replace the child. In the absence of an index, the child is inserted at the end of the children list.
- **operandElementId**: this attribute defines the id of the element from which the inserted or replaced value is taken.
- **operandAttributeName**: this attribute defines the name of the field from which the inserted or replaced value is taken.
- **value**: the value to be replaced, inserted, or deleted.
- **updateInterval**: this attribute defines the updating interval of the elements/attributes within the enclosure of the element. This attribute is one of “none”, “free”, “<Clock-value>”.
    - The value “none” indicates that the referred Structured Information shall be modified only once.
    - The value “free” indicates the author does not mandate any specific behavior and it is up to the implementation to decide the interval of the modification of the referred Structured Information. This is the default value.
    - <Clock-value> indicates that if changes happened during that amount of time, they shall be reflected. This value must be greater than 0. The author must be responsible to assign a meaningful value to reflect the practicality on the implementation.

In 6.9 Table 7: Summary of Possible Attributes per Element, add the following item:

Element name	Attributes
tref	audio-level class color color-rendering display display-align editable fill fill-opacity fill-rule focusable font-family font-size font-style font-variant font-weight id image-rendering line-increment nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left pointer-events requiredExtensions requiredFeatures requiredFonts requiredFormats shape-rendering solid-color solid-opacity stop-color stop-opacity stroke stroke-dasharray stroke-dashoffset stroke-linecap stroke-linejoin stroke-miterlimit stroke-opacity stroke-width style systemLanguage text-anchor text-rendering vector-effect viewport-fill viewport-fill-opacity visibility xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space
externalReference	class entry externalResourcesRequired id nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left requiredExtensions requiredFeatures requiredFonts requiredFormats lsr:rotation lsr:scale security style systemLanguage transform lsr:translation updateInterval xml:base xml:lang xml:space
externalUpdate	attributeName class externalResourcesRequired id index nav-right nav-next nav-up nav-up-right nav-up-left nav-prev nav-down nav-down-right nav-down-left nav-left operandElementId operandAttribute requiredExtensions requiredFeatures requiredFonts requiredFormats systemLanguage type updateInterval value xlink:actuate xlink:arcrole xlink:href xlink:role xlink:show xlink:title xlink:type xml:base xml:lang xml:space

After 6.13, add the following:

6.14 Referencing the fragments of structured information

6.14.1 Introduction

The purpose of the Presentation and Modification of Structured Information (PMSI) is to provide a mechanism to access Structured Information to achieve spatio-temporal presentation and modification of Structured Information in a consistent manner. To access a specific fragment of Structured Information, a pointer scheme for Structured Information is defined.

6.14.2 Pointing scheme for the fragments of structured information

The ***mpeg-pmsi()*** scheme is intended to be used with the XPointer Framework [W3C XPtrFrame] to allow addressing of a fragment of Structured Information under LASeR namespaces. Processing of entities addressed by this scheme will vary according to the type of the element using this scheme.

6.14.2.1 Syntax

The ***mpeg-pmsi()*** scheme is defined in the following EBNF [5] syntax:

```
PointerForStructuredInformation ::= PointerForStructuredInformationSchemeName "(" PointerForStructuredInformationSchemeData ")"
PointerForStructuredInformationSchemeName ::= "mpeg-pmsi"
```

6.14.2.2 Semantics

PointerForStructuredInformationSchemeData complies with the W3C XPath Abbreviated Syntax. The ***mpeg-pmsi()*** scheme supports XPath predicates for filtering out a given element-set from unwanted elements. For instance, the following expression indicates the ***ref*** attribute of ***Resource*** element. ***Resource*** element is the child element of ***Component*** element which has an attribute ***title*** of value 'video_1.'

```
#mpeg-pmsi(Component[@title='video_1']/Resource/@ref))
```

There are other resources which contain Structured Information and which are not expressed in XML. Since this information is structured, it can be mapped to an XML representation and then queried using the PMSI framework. Mapping the structure to XML is beyond the scope of the standard.

In 12.2.1 Main Structure, replace `class element_any` *with*:

```
class element_any {
  uint(extensionIDBits) extensionID;
  vluimsbf5 len; //length in bits
  if (extensionID == 2) { //LASeR AMD1
    const vluimsbf5 reserved = 87; // or "const bit(10) reserved = 599;"
    vluimsbf5 occ2;
    for(int t=0;t<occ2+1;t++) {
      bit(2) ch5;
      switch(ch5){
        case 0:
          SVGAmd1Extension SVGAmd1Extension;
          break;
        case 2:
          LASeRAmd1Extension LASeRAmd1Extension;
          break;
      }
    }
  } else if (extensionID == 3) { //LASeR AMD2
    const vluimsbf5 reserved = 87; // or "const bit(10) reserved = 599;"
    vluimsbf5 occ3;
    for(int t=0;t<occ3+1;t++) {
      bit(2) ch6;
      switch(ch6){
        case 2:
          LASeRAmd2Extension LASeRAmd2Extension;
          break;
        default:
          reserved;
          }
      }
    }
  } else if (extensionID == 4) { //LASeR AMD3
    const vluimsbf5 reserved = 87; // or "const bit(10) reserved = 599;"
    vluimsbf5 occ4;
    for(int t=0;t<occ4+1;t++) {
      bit(2) ch7;
      switch(ch7){
        case 0:
          SVGAmd3Extension SVGAmd3Extension;
          break;
        case 2:
          LASeRAmd3Extension LASeRAmd3Extension;
          break;
        default:
          reserved;
          }
      }
    }
  } else {
    bit[len] toSkip;
  }
}
```

In 12.2.1 Main Structure, add the following classes:

```
class SVGAmd3Extension {
  vluimsbf5 occ1;
  for(int t=0;t<occ1+1;t++) {
    bit(1) ch1;
    switch(ch1){
      case 0:
        element_tref tref;
        break;
      default:
        break;
    }
  }
}

class element_tref {
  bit(1) has_id;
  if (has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if (has_rare) {
    attr_custom_rare rare;
  }
  bit(1) has_fill;
  if (has_fill) {
    attr_custom_paint fill;
  }
  bit(1) has_stroke;
  if (has_stroke) {
    attr_custom_paint stroke;
  }
  bit(1) externalResourcesRequired;
  attr_editable editable;
  bit(1) has_href;
  if(has_href) {
    attr_custom_anyURI href;
  }
  bit(1) has_attr_any;
  if (has_attr_any) {
    attr_any any;
  }
  object_content child0;
}

class LASeRAmd3Extension {
  vluimsbf5 occ2;
  for(int t=0;t<occ2+1;t++) {
    bit(1) ch2;
    switch(ch2){
      case 0:
        element_externalReference externalReference;
        break;
      case 1:
        element_externalUpdate externalUpdate;
        break;
    }
  }
}

class element_externalReferece{
  bit(1) has_id;
  if(has_id) {
    attr_custom_ID id;
  }
  bit(1) has_rare;
  if (has_rare) {
    attr_custom_rare rare;
  }
  bit(1) externalResourcesRequired;
  bit(1) has_updateInterval;
  if(has_updateInterval) {
    attr_updateInterval updateInterval;
  }
  bit(1) has_security;
  if(has_security) {
  // Enumeration: new{0} parent{1}
    bit(2)security;
  }
  bit(1) has_entry;
  if(has_entry) {
    attr_custom_byteAlignedString entry;
  }
  bit(1) has_attr_any;
```

```
    if(has_attr_any) {
      attr_any any;
    }
    object_content child0;
  }

  class element_externalUpdate{
    bit(1) has_id;
    if(has_id) {
      attr_custom_ID id;
    }
    bit(1) has_rare;
    if (has_rare) {
      attr_custom_rare rare;
    }
    bit(1) externalResourcesRequired;
    bit(1) has_updateInterval;
    if(has_updateInterval) {
      attr_updateInterval updateInterval;
    }
    bit(1) has_type;
    if(has_type) {
     attr_externaUpdateType type;
    }
   bit(1) opt_group;
   if(opt_group) {
    vluimsbf5 occ1;
    for(int t=0;t<occ1;t++) {
     privateElementContainer child0[[t]];
    }

  }
```

In 12.2.2 Generic Data Types, add the following classes:

```
  class attr_updateInterval{
     bit(2) choice;
     switch(choice) {
     case 0:
          // Enumeration:none {0}
         break;
     case 1:
          // Enumeration:free {1}
         break;
     case 2:
          // Enumeration:clock-value {2}
         vluimsbf5 updateInterval;
         break;
     default:
         break;
     }
  }

  class attr_externalUpdateType {
     bit(2) type;
     switch(type) {
        case 0:
          attr_Replace Replace;
          break;
        case 1:
          attr_Insert Insert;
          break;
        case 2:
          attr_Delete Delete;
          break;
        default:
          break;
     }
  }

  class attr_Replace {
    bit(1) has_attributeName;
    if(has_attributeName) {
      attr_custom_byteAlignedString  attributeName;
    }
    bit(1) has_index;
    if(has_index) {
      attr_index index;
    }

    bit(1) has_operandAttribute;
```