INTERNATIONAL STANDARD

ISO/IEC 23000-13

First edition
2014-05-15

AMENDMENT 1
2015-10-01

Information technology - Multimedia application format (MPEG-A) —

Part 13:
Augmented reality application format

AMENDMENT 1: ARAF reference software and conformance

Technologies de l'information - Format des applications multimedias —

Partie 13: Format pour les Applications de Realité Augmentée

AMENDEMENT 1: .

Reference number
ISO/IEC 23000-13:2014/Amd.1:2015(E)

Contents

Page

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: Foreword — Supplementary information.

Amendment 1 to ISO/IEC 23000-13:2014 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

— *Part 1: Purpose for multimedia application formats*

— *Part 2: MPEG music player application format*

— *Part 3: MPEG photo player application format*

— *Part 4: Musical slide show application format*

— *Part 5: Media streaming application format*

— *Part 6: Professional archival application format*

— *Part 7: Open access application format*

— *Part 8: Portable video application format*

— *Part 9: Digital Multimedia Broadcasting application format*

— *Part 10: Surveillance application format*

— *Part 11: Stereoscopic video application format*

— *Part 12: Interactive music application format*

— *Part 13: Augmented reality application format*

— *Part 14: VOID*

— *Part 15: Multimedia Preservation Application Format*

— *Part 16: Publish/Subscribe Application Format*

— *Part 17: Multisensorial Media Application Format*

— *Part 18: Media Linking Application FormatPart*

Introduction

Augmented Reality (AR) applications refer to a view of a real-world environment (RWE) whose elements are augmented by content, such as graphics or sound, in a computer driven process. Augmented Reality Application Format (ARAF) is a collection of a subset of the ISO/IEC 14496-11 (MPEG-4 part 11) Scene Description and Application Engine standard, combined with other relevant MPEG standards (e.g. ISO/IEC 23005 - MPEG-V), designed to enable the consumption of 2D/3D multimedia content. Consequently, ISO/IEC 23000-13 focuses not on client or server procedures but on the data formats used to provide an augmented reality presentation.

Information technology — Multimedia application format (MPEG-A) — Part 13: Augmented reality application format, AMENDMENT 1: ARAF reference software and conformance

1 Scope

This part of ISO/IEC 23000 specifies the Reference Software and Conformance.

2 Reference software

The components of the ARAF reference software are implemented in one of the following manners:

- By using existing MPEG-4 scene elements;
- By creating a library that is loaded by a utility software.

The implementation is provided in the electronic attachment of this document.

2.1 Implementation details

Table 1 presents the PROTOs that are implemented using existing MPEG-4 elements as well as the name of the file in which the implementation is included. The implementation is provided using the BT syntax (ISO/IEC 14496-11).

Table 1. ARAF Components implemented using existing MPEG-4 scene elements.

PROTO name	**Filename**
Map	Map.bt
MapMarker	MapMarker.bt
MapOverlay	MapOverlay.bt

Table 2 presents the PROTOs that are implemented by creating a library loaded by a utility software. This is a C/C++ dynamic library acting as a GPAC module. It implements the PROTO by extending GPAC by using its built-in external proto interface.

Table 2. ARAF Components implemented using a library

PROTO name	**Filename**
ReferenceSignal	reference_signal.zip
CameraCalibration	reference_signal_reg.zip
ReferenceSignalDetection	reference_signal_reg.zip

It should be noted that the *CameraCalibration* and *ReferenceSignalDetection* PROTOs share the same library.

The way how the implementation of the plugins was performed is described in the following sections.

2.2 Implementation procedure for adding PROTOs in a library

First a new dynamic library project was added to the main GPAC solution. GPAC loads the modules using predefined functions that need to be implemented in each module. Those functions are the following:

```
const u32 *QueryInterfaces();
GF_BaseInterface *LoadInterface(u32 InterfaceType);
void ShutdownInterface(GF_BaseInterface *ifce);
```

The function **QueryInterface** returns the types of extensions that are supported by the module, in the case of a hardcoded proto it returns GF_HARDCODED_PROTO_INTERFACE.

The function **LoadInterface** and **ShutdownInterface** create and destroy an interface object. The interface object for the hardcoded proto contains two functions:

```
Bool can_load_proto(const char* url);
Bool init(GF_HardcodedProto* itfs, GF_Compositor *compositor, GF_Node *node);
```

The function **can_load_proto** receives a URL as an argument which contains the URL to the PROTO requested by the scene. If the module supports this PROTO, it returns positively.

The function **init** creates a new instance of the PROTO and returns positively on success.

The next step is to define the PROTO object by declaring a new structure. An example of the structure is presented below.

```
struct ReferenceSignalNode
{
    BASE_NODE
    MFString   *source;                 /*exposedField*/
    MFString   *referenceResources;     /*exposedField*/
    SFBool     *enabled;                /*exposedField*/
    MFString   *detectionHints;         /*exposedField*/
    SFVec3f    *translation;            /*exposedField*/
    SFRotation *rotation;               /*exposedField*/
    MFInt32    *onInputDetected;        /*eventOut*/
    MFInt32    *onTranslationChanged;   /*eventOut*/
    MFInt32    *onRotationChanged;      /*eventOut*/
    SFInt32    *onError;                /*eventOut*/
};
```

However since this structure is not known to GPAC internally, it cannot fill in the values, instead it uses a generic representation for the values in the proto. Therefore it is necessary to copy the values from the generic representation in the structure one by one. This is done using the code presented below.

```
1. if (gf_node_get_field(node, 0, &field) != GF_OK) return GF_FALSE;
2. if (field.fieldType != GF_SG_VRML_MFSTRING) return GF_FALSE;
3. rc->source = (MFString *) field.far_ptr;
```

Line 1 uses the function **gf_node_get_field** to get the pointer to the value of the field. Line 2 makes sure that the type of the field value returned corresponds to the expected one. Finally line 3 initializes the value.

The values of the node are initialized each time some of the functions that use them are called.

The initialization of the proto is done few steps.

First step is to create a new instance of the PROTO node and a private stack associated with the node. The association of the stack with the node is done by calling the function:

```
gf_node_set_private(node, stack);
```

The second step is to register the node traversal function with GPAC. This function will be called each time the node is traversed and when the node should be destroyed. The registration is done by calling the function:

```
gf_node_set_callback_function(node, TraverseReferenceSignal);
```

The final step is to register the node to receive image updates from the source. This is done by creating a TextureHandler as setting a texture update function to it. The following code is an example:

```
gf_sc_texture_setup(&(stack->refreshTextureHandler), compositor, node);
stack->refreshTextureHandler.update_texture_fcnt = UpdateTextures;
```

The texture handler function will be called at each step in the simulation. This is where the PROTO can have access to the media to reference resources images and the source image. In order to access the images the PROTO has to request their decoding. This is done by accessing the media object for the image:

```
gf_scene_get_media_object_ex((GF_Scene*)gf_sg_get_private(gf_node_get_graph(node)),
        &murl, GF_MEDIA_OBJECT_VIDEO, GF_FALSE, NULL, GF_FALSE, node);
```

and then requesting the decoding:

```
gf_mo_play(media_object, 0, -1, GF_TRUE);
```

When the image is decoded, a call to the function

```
data = gf_mo_fetch_data(stack->mediaToDetectStreams[i], GF_TRUE, &eos, &ts, &size);
```

will return the image data.

Depending if the image is a reference resource or a source it will analyzed and added to the database, or analyzed and compared to the database respectively. If the result of the comparison is positive, the corresponding event outs are executed and the appropriate fields modified by using the code presented below.

```
1. gf_node_get_field(node, 6, &onInputDetected);
2. gf_node_event_out(node, onInputDetected.fieldIndex);
3. gf_node_changed(node, &onInputDetected);
```

Line 1 gets the field from the PROTO node instance, line 2 fires the event out for that field and line 3 notifies the system that the node field value has changed.

2.3 Utility Software

The utility software of MPEG-U is the Osmo4 player of GPAC (https://github.com/gpac/gpac). GPAC is an open-source project distributed under the LGPL license.

For convenience, an installer for the Windows platform is made available at the following URL: http://gpac.wp.mines-telecom.fr/downloads/gpac-nightly-builds/.

IT IS IMPORTANT TO KEEP IN MIND THAT SOME MODULES IN GPAC MAY USE GPL SOFTWARE. This is especially true for the UPnP module, which relies on a third party library, Platinum, for the UPnP stack. The Platinum library is distributed under GPL.

More details on GPAC features and tools can be found on the project web page.

The reference and utility software have been tested on the following platforms:

- Windows 7
- Android

3 Conformance

The conformance testing is done by using the example provided content.

The following interactive applications implemented using the ARAF components are provided in the electronic attachment:

- Augmented Reality game: **ARQuiz**
- Reference Signal validation: **AudioBook**
- Camera Calibration validation: **CameraCalibration**

All of the applications are provided by using the BT format and can be loaded in a compatible MPEG-4 player.

Table 3 and Table 4 present the ARAF PROTOs and the BIFS nodes used by the applications.

ARQuiz is used to test the conformance of the **Map** related PROTOs.

The **AudioBook** application is used to test the conformance of the **ReferenceSignal** PROTO.

The **CameraCalibration** application is uses to test the conformance of the **CameraCalibration** PROTO.

Table 3. ARAF PROTOs used by the applications

Application name	**ARAF PROTOs used**
ARQuiz	Map MapMarker MapOverlay
AudioBook	ReferenceSignal
CameraCalibration	CameraCalibration

Table 4. MPEG-4 BIFS Nodes used by the applications

Application Name	**Nodes Used**
ARQuiz	Appearance AudioSource Background2D Cylinder FontStyle Group ImageTexture Inline InputSensor Layer2D Layer3D Layout LineProperties Material Material2D MediaControl

	MediaSensor MovieTexture NavigationInfo OrderedGroup OrientationInterpolator Rectangle Script Shape Sound2D Switch Text TimeSensor TouchSensor Transform Transform2D Viewpoint Viewport
Audio Book	Appearance AudioSource Background2D Bitmap FontStyle ImageTexture Material2D MovieTexture OrderedGroup Rectangle Script Shape Sound2D Text Transform2D
Camera Calibration	Appearance Background2D Bitmap FontStyle Material2D MovieTexture OrderedGroup Rectangle Script Shape Text Transform2D

The explanation of each of the applications is provided in Annex A.