INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG2003/M10375
December 2003, Hawaii, USA

Source: **ENST**
Status: **for discussion at the 67th MPEG Meeting**
Title: **various enhancements for MPEG-4 Systems**
Author: **Jean Le Feuvre, Cyril Concolato, Jean-Claude Dufourd, ENST**

Abstract

While designing content we have come across some missing features in the current MPEG-4 systems specification. We therefore propose to amend the standard to bring in these new functionalities.

I Valuator, SFTime and SFString

I.1 Problematic

The MPEG-4 standard includes very powerful tools to deal with media playback, MediaControl and MediaSensor. MediaSensor is extremely interesting for content authors willing to provide more details of media currently playing, such as exact timing and duration. This functionality is usually achieved either through script or valuator nodes. However, the media time is handled in BIFS as an SFTime field, hence expressed in seconds. While this is important, there is a drawback: end-user display of media time is usually (if not always) presented in the hh:mm:ss format, where ‘hh’, ‘mm’, ‘ss’ respectively are hours, minutes and seconds. In order to be able to display media time or media duration in this format within the scene, the content designer must use ECMAScript run-time. This therefore discards simple terminals without Script node support, and forces using scripts for a trivial functionality, which is usually a bit heavier (more RAM, slower processing). We therefore suggest modifying the type-casting of SFTime to SFString in the valuator node, so as to provide a low-cost, efficient way of displaying media time and duration.

I.2 Proposed Solution

Add to the Valuator node semantics:

“If the eventIn is of SFTime type then the conversion to string format shall be in the format “hh:mm:ss” where ‘hh’, ‘mm’, ‘ss’ are respectively hours, minutes and seconds of the input SFTime value.”

II Remote location in Descriptors

II.1 Problematic

The MPEG-4 standard allows referencing remote resources in both OD and ESD descriptors. This is extremely useful for client/server interactivity, such as relocating an inline scene based on client profiles (advertising) or a media stream based on user permissions (IPMP) or transport type. However the specification only provides support for 255 byte length URL, coded as UTF-8. This is obviously not enough, if one wants to integrate server-side scripting indications such as CGI or embed data in the URL using the DATA: format as specified by IETF. Cases have also been seen where transport of the IOD in the ISMA fashion are failing due to too large configuration information in the base64 format. We are therefore suggesting extending the URL usage in the OD framework.

II.2 Proposed Solution

Change the URLlength in the InitialObjectDescriptor, ObjectDescriptor and ES_Descriptor sections from:

```
if (URL_Flag) {
       bit(8) URLlength;
       bit(8) URLstring[URLlength];
} else {
       …
}
```

to

```
if (URL_Flag) {
       bit(8) URLlength;
       if (! URLlength) {
              bit(5) nbBits;
              bit(nbBits) URLlength;
       }
       bit(8) URLstring[URLlength];
} else {
       …
}
```

“`URLlength` – the length of the subsequent `URLstring` in bytes. If `URLlength` is zero, the number of bits used to code the URL length is coded, then the URL length itself. This allows encoding any format of URL with more than 255 bytes.”

III Remote Objects synchronization

III.1 Problematic

The MPEG-4 OD framework provides efficient tools for streams synchronization, but lacks efficient synchronisation in case of distributed objects used in a presentation. The stream synchronization is indeed performed through ES descriptors Identifiers, but these identifiers are only relevant within a same network service, if not a same OD namespace. Therefore synchronizing objects introduced in the scene through remote object descriptors (OD with URL string) one to another is not possible, whereas this kind of application is vital to synchronized advertisement using service redirections and many other market area, be it for authors (movie sub-titling, dividing content in smaller, reusable scenes) or for providers (single content distribution through several servers or even protocols).

We therefore suggest the Object Synchronization on top of the Stream Synchronization to cope with this deficiency of MPEG-4 Systems.

III.2 Proposed Solution

Introduce the notion of Object Clock:

"Object Clock: an object as defined by an ObjectDescriptor is composed of several Elementary streams. Each of these streams may run on its own time base (OTB) or use another stream as its OTB. In case all streams composing an object use the same OTB, this OTB is called Object Clock. In all other cases, the Object Clock is undefined."

Change the reserved field of InitialObjectDescriptor from:

```
const bit(4) reserved=0b1111;
```

to

```
const bit(1) no_sync_od;
if (!no_sync_od) {
       const bit(10) sync_od_id;
       const bit(6) reserved=0b000000;
}
const bit(3) reserved=0b111;
```

Change the reserved field of ObjectDescriptor from:

```
const bit(5) reserved=0b1111.1;
```

to

```
const bit(1) no_sync_od;
if (!no_sync_od) {
       const bit(10) sync_od_id;
       const bit(6) reserved=0b000000;
}
const bit(4) reserved=0b1111;
```

Update the semantics of both descriptors:

"`no_sync_od`: specifies whether another object is used for synchronization.

`sync_od_id`: ID of the object descriptor whose Object Clock is used for synchronization. This ID is only valid in the current OD namespace and shall be different from this ObjectDescriptor ID. If the target descriptor describes a remote resource, the Object

Clock is the one of the resolved resource. If the Object Clock is undefined, synchronization is undefined. If the Object Clock is defined, then all streams defined in this descriptor, or in the resolved resource if this descriptor has a URL string given, shall use the specified Object Clock as their OTB and shall ignore any clock dependencies indicated at the stream level."

Note that the object synchronization could also be specified through extension descriptors, but we strongly prefer this solution since synchronization is a core feature of the OD framework rather than an extension.

IV Extern Proto coding

IV.1 Problematic

As explained in w5980, the current binary coding of externProto does not allow for real proto libraries usage. We strongly support the decision taken by the MPEG systems group to use an externProto addressing scheme similar to the externProto scheme used in VRML, and summarize hereafter the proposed modification.

This scheme allows to directly reference a prototype in an external resource by its binary identifier or string identifier through the resource location field of the ExternProto. For example, referencing proto 10 in the resource http://server/library.mp4 would result in the following URL for the externProto: http://server/library.mp4#10. The proto could also be referenced by name, e.g. if the proto 10 has is named "MyProtoTest" in the extern resource, the ExternProto URL would be http://server/library.mp4#MyTestProto. The ExternProto may then use any binary identifier or any name in the MPEG-4 scene, it is unambiguously pointing to a unique proto in the extern resource.

IV.2 Proposed Solution

Replace in 9.3.7.4.2.2:

"

The EXTERNPROTO opens a BIFS-Command stream that contains a `ReplaceScene` command with a `BIFSScene` containing the PROTO definitions. The EXTERNPROTO code is found in the PROTO contained in this new scene with the same ID in both scenes. Nodes contained in the EXTERNPROTO scene shall be ignored.

"

by

"

The EXTERNPROTO opens a BIFS-Command stream that contains a `ReplaceScene` command with a `BIFSScene` containing the PROTO definitions. The EXTERNPROTO code is found in the PROTO contained in this new scene. The url field allows to uniquely identify the EXTERNPROTO code through the following url scheme : "`resource_URL#ProtoID`" or "`resource_URL#ProtoName`", where `resource_URL` is the location of the scene to open, `ProtoID` the binary ID of the proto in the new scene and `ProtoName` the name of the proto in

the new scene when this scene is encoded with `USENAMES`. In case "`#ProtoID`" or "`#ProtoName`" is omitted in the location, the first proto in the new scene with the same `PROTOinterfaceDefinition` shall be used.

Nodes contained in the EXTERNPROTO scene shall be ignored.
Opening of the scene description stream follows the MPEG-4 content access procedure described in 8.7.3.8.2.
"

We also suggest specifiying that the `resource_URL` may be described by an existing ObjectDescriptor with the usual MPEG-4 scheme "OD:ODID".

V Conclusion

In this contribution, we have explained some problems faced by content authors and have provided simple yet efficient solutions to these problems. We kindly invite the MPEG Systems group to consider them.