SMPTE REGISTERED DISCLOSURE DOCUMENT

MDA Program Specification

The attached document is a Registered Disclosure Document prepared by the proponent identified below. It has been examined by the appropriate SMPTE Technology Committee and is believed to contain adequate information to satisfy the objectives defined in the Scope, and to be technically consistent.

This document is NOT a Standard, Recommended Practice or Engineering Guideline, and does NOT imply a finding or representation of the Society.

Every attempt has been made to ensure that the information contained in this document is accurate. Errors in this document should be reported to the proponent identified below, with a copy to eng@smpte.org.

All other inquiries in respect of this document, including inquiries as to intellectual property requirements that may be attached to use of the disclosed technology, should be addressed to the proponent identified below.

Proponent contact information:

Scott Smyers
DTS, Inc.
130 Knowles Drive, Suite B
Los Gatos, CA 95032

Email: scott.smyers@dts.com

Approved
August 23, 2016

Table of Contents

Conventions

All sections are normative, unless otherwise indicated.

Pseudo-code and property names use font style courier new.

The expressions MAY, NEED NOT, SHALL, SHALL NOT, SHOULD, and SHOULD NOT indicate normative behavior ISO/EIC Directives, Part 2.

VERBAL FORM	SEMANTICS
MAY	It is allowed
NEED NOT	It is not required that
SHALL	Is required that
SHALL NOT	Is required to be not
SHOULD	It is recommended that
SHOULD NOT	It is not recommended that

Introduction (Informative)

The MDA Program, or simply Program hereafter, is a self-contained object-based audio program. As such it consists of a collection of audio objects, each combining an audio waveform with metadata. The metadata indicates, for instance, when the object occurs on the Program timeline or where it is positioned within the soundfield. It is used to control the mapping of the audio object waveform to output loudspeakers at playback.

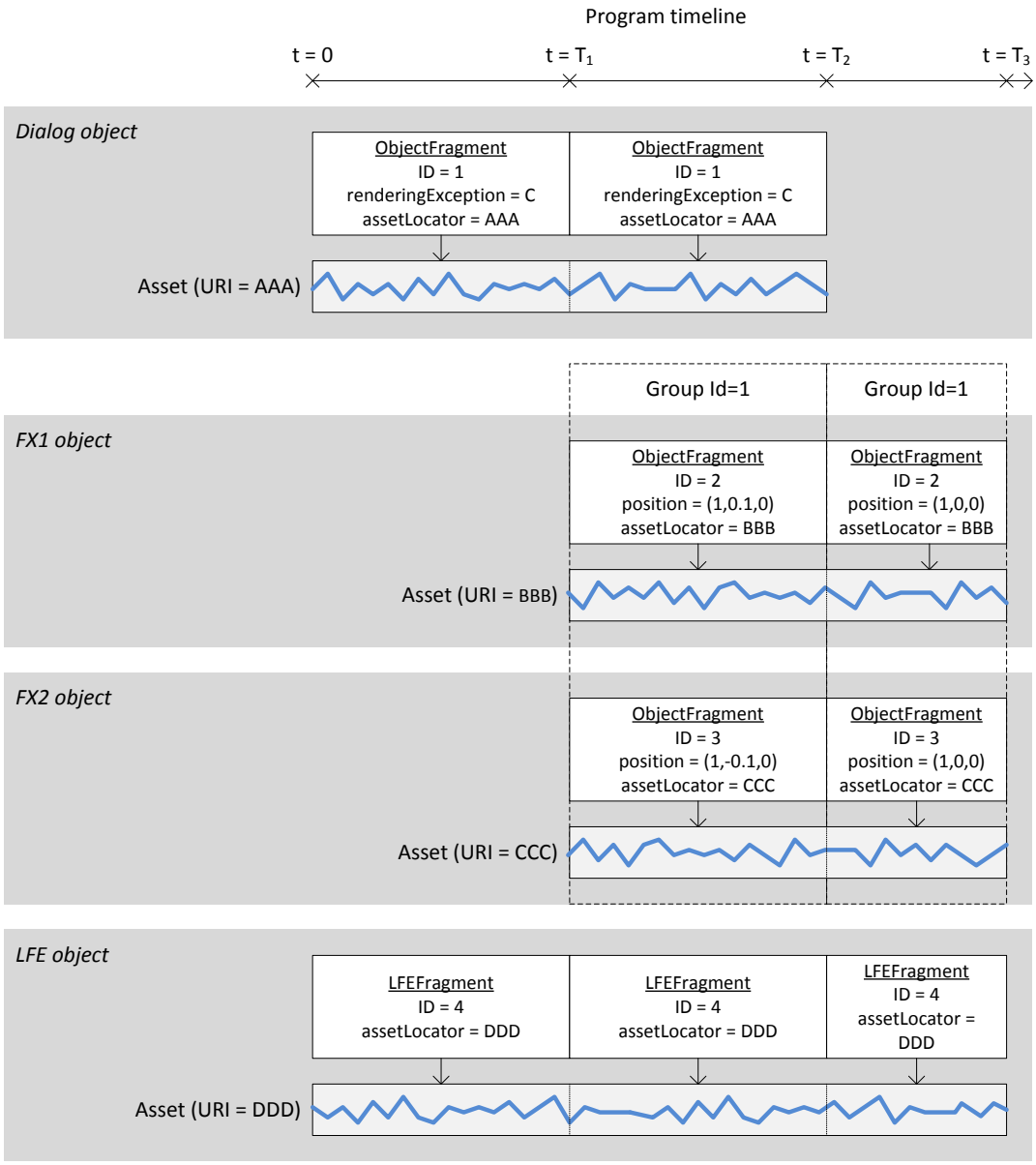

Figure 1 – Sample Program. Only a subset of all Fragment properties are shown

Figure 1 depicts a Program that consists of 4 audio objects: Dialog, FX1, FX2 and LFE. While the LFE object exists from t = 0 to t = T_3, the dialog object exists only from t = 0 to t = T_2, and the FX2 and FX2 objects from t = T_1 to t = T_3. The Program object model allows for any number of audio objects to overlap at any point in time, and an audio object can be as short as a sample or as long as the program. Each audio object is assigned an identifier that is unique within the scope of the Program.

The metadata associated with each object is divided into Fragments, each corresponding to a period of time during which the metadata is static. To simplify the Program structure, Fragment boundaries are aligned.

Two kinds of Fragments, and hence audio objects, are defined:

- An ObjectFragment corresponds to an object associated with a spatial locus. For instance, the position of the FX1 and FX2 objects in Figure 1 changes from t = T_1 to t = T_3. This spatial locus is used to determine the loudspeakers that will output the waveform associated with the object. It is also possible to instruct that an object waveform be routed through a specific loudspeaker, if present.

- An LFEFragment corresponds to an object whose waveform is intended for routing to a Low Frequency Effect (LFE) channel, and is therefore not associated with a spatial location.

Each Fragment references a sequence of audio samples, i.e. the object waveform, within an underlying asset identified using a Uniform Resource Identifier (URI). Depending on applications, the asset can be carried alongside the Fragment metadata or be remote. Multiple Fragments can reference the same audio samples within a single asset.

As illustrated by the FX1 and FX2 objects, Fragments can be combined into a Group, which logically groups the two ObjectFragments and contains metadata common to them . The object model also allows Fragments to be combined into a Switch, which indicates that only one of the Fragments is rendered at any given time. Groups and Switches are recursive entities that can themselves contain Groups and Switches. From the perspective of the object model, Fragments, Groups and Switches are all subclasses of the Entity class, which represents arbitrary entities of the Program timeline.

In order to specify unambiguously how object metadata is used to map object waveforms to loudspeaker outputs, i.e. rendered, a Reference Renderer is fully specified in Section 9. The Reference Renderer uses the Vector Base Amplitude Panning formalism (VBAP), which was introduced by Pulkki et al. and has since been extensively studied. VBAP is an extension of the familiar tangent law for pair-wise panning to three-dimensional speaker configurations. Specifically, given a loudspeaker triplet on the unit sphere and a point source object located within the spherical triangle defined by the loudspeakers, the contribution of the object waveform to each of the loudspeaker is determined by the coordinates of the object within the linear basis formed by the three loudspeakers (see Figure 2). Objects with a finite extent can be rendered as a collection of point sources. More complex loudspeaker configurations can be decomposed into multiple speaker triplets.

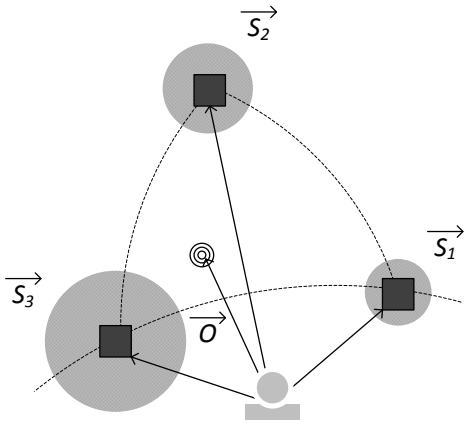

Figure 2 – Rendering audio objects using VBAP. The shaded areas show the relative output power at each of the speakers and are determined by expressing the object vector in the basis formed by the three loudspeakers.

To support a range of applications within its stated scope, the Program object model is designed to be flexible, e.g. the number of simultaneous Fragments is not limited, and offers multiple extension points. Applications are therefore expected to constrain or extend the object model to suit their specific requirements. Similarly, this specification does not define a concrete representation of the Program, and mappings to bitstream structures and transmission mechanisms are left to other documents.

1 Scope

This document specifies the object model and reference renderer for the MDA Program. The MDA Program is a self-contained representation of an object-based soundfield designed for linear content. It is specified independently of transport mechanisms.

2 Normative References

Internet Engineering Task Force (IETF) (January 2005). RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax

ISO/EIC Directives, Part 2. Edition 6.0, 2011-04

OMG, Object Constraint Language (OCL), Version 2.3.1, http://www.omg.org/spec/OCL/2.3.1/PDF

OMG, Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF

3 Timeline

A Program defines a sample-accurate timeline onto which Entity instances (see Section 6.6) are placed.

Positions on the timeline SHALL be expressed as integer multiples of the inverse of the Program audio sample rate (see Section 6.5.2), i.e. as an integer number of audio samples.

The origin of the timeline (t=0) is arbitrary.

4 Audio Object

An Audio Object is the sequence of all Fragment instances (see Section 6.6) with the same `id`, ordered as they appear on the timeline.

Two Fragment instances belong to the same Audio Object if and only if they have the same `id` value.

5 Coordinate System

This specification uses the Cartesian coordinate system illustrated in Figure 3 and specified as follows:

- The listener is located at the origin O=(0,0,0), facing the front of the room;
- The positive z-axis is perpendicular to the floor of the room, and directed to the ceiling;
- The positive y-axis is directed towards the front of the room;
- The positive x-axis is directed to the right of the listener;
- Loudspeakers lie on the unit sphere S; and
- The unit circle in the x-y plane is the locus of traditional horizontal two-dimensional loudspeaker configurations.