

ICWG Interactive Audio

API Rev 2.0

Preface

This document supersedes the CDIAE document sent out previously. This Document is the result of a meeting held at the BBQ in San Antonio, Texas on Oct. 26th 1996.

Attending were:

David Javalosa, Inscape

Mark Miller, Crystal Dynamics

Todor Fay, Microsoft

Ralph Thomas, SSI

Monty Schmidt, Sonic Foundry

Mike Land, LucasArts

Tom White, MMA

The main problem with the CDIAE document was that it was far too specific and represented more of an implementation than a general purpose description of the functionality requirements. Defining the functionality instead of the specifics is closer to the original intent of the ICWG.

New Definitions

Segment: *A bounded chunk of time based data.* (old examples still hold)

Track: *A horizontal (time axis) subdivision of a segment.* (old examples still hold)

Functionality

General Requirements

- 1) System must be able to play multiple segments synchronously
- 2) System must recognize callbacks (with associated values) that are embedded in Segments or Tracks by the authoring tools.
- 3) There will be some number of 32 bit registers (24 minimum?) that are readable and writable from both the main application and the sound driver
- 4) Different data types (digital audio, MIDI, algorithmic generator data, etc.) will have specific, but not necessarily identical, parameters associated with each Segment. These parameters will be stored and updated as the Segment is played and can be changed in real time.
- 5) Interactivity Logic will be defined in a script or in a set of scripts
- 6) Scripts can be called in three ways:
 - a) direct function calls from the main application
 - b) call backs from within a segment
 - c) Periodic interrupts defined within the script itself
- 7) Scripts may be located in the following places:
(Todor, could you take a stab at this)
 - a) within the main application space
 - b) within the segment data (if possible)
 - c) within the sound driver application space
- 8) A consistent, general purpose priority scheme will be used to make sure that the architecture degrades gracefully as resources become scarce. This scheme will be described below.

9) While general functionality will be consistent across platforms and implementations, specific functions may be located in different components of the systems as is most efficient. For example, the function that allows looping of segments may reside in the main application code or in the sound driver specific code as is most efficient for a specific application.

Specific Functions

Flow Control Set:

Play Segment

Stop Segment

Pause Segment

Unpause Segment

Modification Set:

The following commands will be used within the script to modify the parameters associated with specific Segments. The direct MIDI command will be used to change specific parameters that are accessible via MIDI, but are not stored with Segments (such as DLS instrument parameters)

Set Parameter

Get Parameter

Change Parameter over Time

Send Direct MIDI command

Parameters By Data Type:

Multi-track Digital Audio Segments

this includes single track DA Segments by default

This implies that individual tracks are interleaved

Segment Volume

Track Volume (per track)

Segment Pan+

Track Pan+

(Pan+ will be used to identify the location of in 3 space as defined by the 3DWG)

Segment Mute / Unmute

Track Mute / Unmute (per track)

Segment Playback Rate

Segment Priority

Track Priority (per track)

Install Modify Hook

(Install Modify Hook allow for user defined functions to operate on the Segment or Track data in the cue before it is played)

Multi-track MIDI Segments

This includes single track MIDI Segments by default.

This implies that individual tracks are interleaved

Segment Volume

Track Volume (per track)

Segment Pan+

Track Pan+

(Pan+ will be used to identify the location of in 3

space as defined by the 3DWG)

Segment Mute / Unmute

Track Mute / Unmute (per track)

Segment Priority

Track Priority (per track)

Install Modify Hook

(Install Modify Hook allow for user defined functions to operate on the Segment or Track data in the cue before it is played. Install Modify Hook, for example, will allow other MIDI parameters to be defined and stored so that they can be operated on by function described in or called by scripts)

Track Program (as in MIDI Program Change command)

Other Data types will have their own specific parameters to be defined as the data types are included in the general specification.

Priority:

When a request to play a Segment finds that insufficient resources are available to play that Segment, the conflict will be resolved according to the following rules.

If the priority associated with the new Segment is greater than or equal to that of Segments that are currently playing, the new data will steal resources in the following way:

It will look at all currently playing Segments and steal from the one that is using the needed resources with the lowest current priority.

If more than one currently playing Segments has the same, lower priority it will steal need resources from the one that has been playing longest

If the needed resources can be stolen, the Segments from which they are to be stolen will be stopped

If the needed resources can not be found in this fashion, the new Segment will not play.

This same scheme can be applied to MIDI instruments by using MIDI to assign a priority to all notes playing on a specific Track of a specific Segment .

Next Order of Business

1) The 'location' of the various types of functionality (in the sound driver, in the main application, in the Segments themselves)

2) The 'location' and format of the script(s)

((A simple model of these 'locations' was discussed but is clearly not final. It is as follows:

Main Application <—> Registers <—> Script <—> Puppet Strings <—> Sound Driver
(<—> Collection Files) <—> Content))