

MPEG-I Audio CfP Submission

Generated by Doxygen 1.8.18

| | |
|--|-----------|
| 1 MPEG-I Audio Renderer CfP Max External Skeleton | 1 |
| 1 MPEG-I Audio Renderer CfP Max External Skeleton | 1 |
| 2 Real-time fault detection | 1 |
| 3 Module Index | 2 |
| 3.1 Modules | 2 |
| 4 File Index | 2 |
| 4.1 File List | 2 |
| 5 Module Documentation | 3 |
| 5.1 Max API | 3 |
| 5.1.1 Detailed Description | 3 |
| 5.1.2 Function Documentation | 3 |
| 5.2 MPEG-I Audio CfP submission interface | 6 |
| 5.2.1 Detailed Description | 6 |
| 5.2.2 Typedef Documentation | 6 |
| 5.2.3 Enumeration Type Documentation | 6 |
| 5.2.4 Function Documentation | 7 |
| 6 File Documentation | 11 |
| 6.1 mpegi-renderer_tilde.h File Reference | 11 |
| 6.1.1 Typedef Documentation | 11 |
| 6.2 win/get_module_file_name.h File Reference | 12 |
| 6.2.1 Function Documentation | 12 |
| Index | 13 |

1 MPEG-I Audio Renderer CfP Max External Skeleton

For more information about Max Externals see the SDK documentation (<https://cycling74.com/sdk/max-sdk-8.0.3/html/index.html>).

[Real-time fault detection](#)

2 Real-time fault detection

To detect potential audible artifacts caused by renderers under test, each proponent renderer shall incorporate an instance of the detector class in their DSP processing path (i.e.: [renderer_perform64\(\)](#)). The detector class measures the time spent in the DSP callback and reports a warning back to the AEP if a certain threshold is exceeded.

The detector class is defined in the header file **detector.h**

```
#include "detector.h"
```

The detector maintains internal state and must therefore persist between calls. The instance can be stored in the main renderer struct, e.g.:

```
typedef struct _renderer {
    c74::max::t_pxobject ob; // DSP-ready Max object data
    void* messageOutlet;    // stored pointer to the message outlet
    detector_t *detector;    // detector instance
    // More members go here ...
} t_renderer;
```

Warnings are reported back to the AEP via the message outlet of the Max External. The message outlet is connected with the detector instance on initialization:

```
renderer->detector =
    make_detector<detector_t>(window_size, detector_threshold, renderer->messageOutlet);
```

Window size and detector threshold are configuration and sound-card dependent. Values for both threshold and window size are provided by the AEP on initialization of the Max External (in [renderer_new\(\)](#))

Resources allocated by the detector should be freed, e.g. in [renderer_free\(\)](#):

```
free_detector(&renderer->detector)
```

Detector measurements shall be started for each DSP callback, i.e. in [renderer_perform64\(\)](#):

```
auto const timer = renderer->detector->start();
```

The **timer** object returned by the detector instance measures the time until its destruction, therefore it shall be placed immediately at function entry.

See also

Example implementation in **mpegi-renderer_tilde.cpp**.

3 Module Index

3.1 Modules

Here is a list of all modules:

| | |
|--|--------------------------|
| Max API | 3 |
| MPEG-I Audio CfP submission interface | 6 |

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

| | |
|---|---------------------------|
| mpegi-renderer_tilde.h | 11 |
| win/get_module_file_name.h | 12 |

5 Module Documentation

5.1 Max API

Functions

- void * [renderer_new](#) (c74::max::t_symbol *sym, long argc, c74::max::t_atom *argv)
Max/MSP callback for a new renderer object instance.
- void [renderer_free](#) (t_renderer *self)
Max/MSP callback for deletion of a renderer object instance.
- void [renderer_assist](#) (t_renderer *self, void *unused, long io, long index, char *string_dest)
Max/MSP callback to print mouse-over help messages on the inlets and outlets.
- void [renderer_dsp64](#) (t_renderer *self, c74::max::t_object *dsp64, short *count, double samplerate, long maxvectorsize, long flags)
Max/MSP callback for initialization of audio processing.
- void [renderer_perform64](#) (c74::max::t_object *self, c74::max::t_object *dsp64, double **ins, long numins, double **outs, long numouts, long sampleframes, long flags, void *userparam)
Max/MSP callback for main audio processing.

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 [renderer_assist\(\)](#) void [renderer_assist](#) (
 t_renderer * self,
 void * unused,
 long io,
 long index,
 char * string_dest)

Max/MSP callback to print mouse-over help messages on the inlets and outlets.

Parameters

| | |
|--------------------|--|
| <i>self</i> | Pointer to the renderer data structure defined above |
| <i>unused</i> | |
| <i>io</i> | Flag whether an inlet or an outlet help message is requested |
| <i>index</i> | Index of the requested inlet or outlet |
| <i>string_dest</i> | Pointer to the C string that will be printed in the mouse-over |

Note

Implementation of this method is **optional**

```

5.1.2.2 renderer_dsp64() void renderer_dsp64 (
    t_renderer * self,
    c74::max::t_object * dsp64,
    short * count,
    double samplerate,
    long maxvectorsize,
    long flags )

```

Max/MSP callback for initialization of audio processing.

Parameters

| | |
|----------------------|---|
| <i>self</i> | Pointer to the renderer data structure defined above |
| <i>dsp64</i> | Pointer to MaxMSP's internal representation of the DSP object |
| <i>count</i> | Flags if the signal inlets are connected to a source or not |
| <i>samplerate</i> | Samplerate as set in MaxMSP audio settings |
| <i>maxvectorsize</i> | Block size as set in MaxMSP audio settings |
| <i>flags</i> | MaxMSP internal flags |

```

5.1.2.3 renderer_free() void renderer_free (
    t_renderer * self )

```

Max/MSP callback for deletion of a renderer object instance.

Parameters

| | |
|-------------|--|
| <i>self</i> | Pointer to the renderer data structure defined above |
|-------------|--|

```

5.1.2.4 renderer_new() void* renderer_new (
    c74::max::t_symbol * sym,
    long argc,
    c74::max::t_atom * argv )

```

Max/MSP callback for a new renderer object instance.

The module must have exactly **two** signal outputs and **one** message output. The number of inputs is set at construction time

Parameters

| | |
|-------------|---|
| <i>sym</i> | Symbol used to identify the object |
| <i>argc</i> | Number of arguments given for the object instance |
| <i>argv</i> | Pointers to arguments <ul style="list-style-type: none"> • Number of input channels (A_LONG) • Detector threshold in microseconds (A_LONG) • Detector window size in blocks (A_LONG) |

Returns

pointer to new renderer data structure (NULL on error)

```
5.1.2.5 renderer_perform64() void renderer_perform64 (
    c74::max::t_object * self,
    c74::max::t_object * dsp64,
    double ** ins,
    long numins,
    double ** outs,
    long numouts,
    long sampleframes,
    long flags,
    void * userparam )
```

Max/MSP callback for main audio processing.

Parameters

| | |
|---------------------|--|
| <i>self</i> | Pointer to the renderer data structure defined above |
| <i>dsp64</i> | Pointer to MaxMSPs internal representation of the DSP object |
| <i>ins</i> | 2D array of input samples (1st dimension: number of input channels, 2nd dimension: block size) |
| <i>numins</i> | Number of input channels |
| <i>outs</i> | 2D array of output samples (1st dimension: number of output channels, 2nd dimension: block size) |
| <i>numouts</i> | Number of output channels |
| <i>sampleframes</i> | Block size |
| <i>flags</i> | MaxMSP internal flags |
| <i>userparam</i> | Pointer to additional internal payload |

5.2 MPEG-I Audio CfP submission interface

Typedefs

- typedef enum `_renderer_status_code` `STATUS_CODE`
Renderer status code.

Enumerations

- enum `_renderer_status_code` { `RENDERER_SUCCESS` = 0, `RENDERER_ERROR` = 1, `RENDERER_INVALID_UPDATE` = 2, `RENDERER_BUSY` = -1 }
Renderer status code.

Functions

- void `renderer_loadScene` (`t_renderer` *self, `c74::max::t_symbol` uid)
Load a scene.
- void `renderer_loadLSDF` (`t_renderer` *self, `c74::max::t_symbol` lsdfPathAndFilename)
Load an LSDF.
- void `renderer_setMaxDelay` (`t_renderer` *self, long maxDelay)
Max calls this function to inform the renderer about the maximum delay in the test.
- void `renderer_triggerUpdate` (`t_renderer` *self, `c74::max::t_symbol` *s, long argc, `c74::max::t_atom` *argv)
Trigger an update event (L2/L3)
- void `renderer_setUserPose` (`t_renderer` *self, `c74::max::t_symbol` *s, long argc, `c74::max::t_atom` *argv)
Max calls this function to provide the renderer with the latest position/orientation of the user (= head)
- void `renderer_loadHRIRSet` (`t_renderer` *self, `c74::max::t_symbol` *sofaFilePath)
Load head related impulse responses from a sofa file.

5.2.1 Detailed Description

5.2.2 Typedef Documentation

5.2.2.1 STATUS_CODE typedef enum `_renderer_status_code` `STATUS_CODE`

Renderer status code.

Used to signal renderer status to Audio Evaluation Platform. User defined error codes should be > `RENDERER_INVALID_UPDATE`

5.2.3 Enumeration Type Documentation

5.2.3.1 _renderer_status_code enum `_renderer_status_code`

Renderer status code.

Used to signal renderer status to Audio Evaluation Platform. User defined error codes should be > `RENDERER_INVALID_UPDATE`

Enumerator

| | |
|-------------------------|------------------------------------|
| RENDERER_SUCCESS | No error. |
| RENDERER_ERROR | Generic error. |
| RENDERER_INVALID_UPDATE | Received invalid L3 update. |
| RENDERER_BUSY | Renderer busy, e.g. loading scene. |

5.2.4 Function Documentation

5.2.4.1 renderer_loadHRIRSet() `void renderer_loadHRIRSet (`
`t_renderer * self,`
`c74::max::t_symbol * sofaFilePath)`

Load head related impulse responses from a sofa file.

This function may be called by Max/MSP to request loading of HRIRs contained in the specified SOFA file.

Parameters

| | |
|---------------------|--|
| <i>self</i> | pointer to msp-object, provided by the runtime |
| <i>sofaFilePath</i> | Absolute path to the SOFA file |

Note

Implementation of this method is **optional**

If implemented, this function **must** be bound to the message **loadHRIRSet**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_loadHRIRSet, "loadHRIRSet", A_SYM, 0);
```

5.2.4.2 renderer_loadLSDF() `void renderer_loadLSDF (`
`t_renderer * self,`
`c74::max::t_symbol lsdfPathAndFilename)`

Load an LSDF.

Max calls this function to instruct the renderer to load a specific LSDF

This function **must** be bound to the message **loadLSDF**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_loadLSDF, "loadLSDF", A_SYM, 0);
```

Parameters

| | |
|----------------------------|----------------------------------|
| <i>self</i> | Pointer to the renderer instance |
| <i>lsdfPathAndFilename</i> | Full LSDF path and filename |

5.2.4.3 `renderer_loadScene()` `void renderer_loadScene (`
`t_renderer * self,`
`c74::max::t_symbol uid)`

Load a scene.

Max calls this function to instruct the renderer to load a specific scene, identified by an unique identifier, from its bitstream representation

This function **must** be bound to the message **loadScene**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_loadScene, "loadScene", A_SYM, 0);
```

Directory structure

The Max Externals plugin shall request its location from the system using `get_module_file_name()`. With this it knows the <ROOT>\ProponentData\SYS\<X> folder and thus can access its own bitstream file with the identifier argument from the `renderer_loadScene()` call. Hence the bitstream basename must be the same name for all proponents or a mapping must be implemented in the plugin. **Note** that the file extension is not defined

```
<ROOT>
( ... )
ProponentData
  SYS1
    <pluginName1>.mxe64
    <Scene_ID>_<Test_ID>.bin          % E.g. Office_test1
    <Scene_ID>_<Test_ID>.bin
    <Scene_ID>_<Test_ID>.bin          % E.g. Hospital_test1
    <Scene_ID>_<Test_ID>.bin
    <Scene_ID>_<Test_ID>.bin
    <Scene_ID>_<Test_ID>.bin
    ( ... )
  SYS2
    <pluginName2>.mxe64
    <Scene_ID>_<Test_ID>.bs
    <Scene_ID>_<Test_ID>.bs
    <Scene_ID>_<Test_ID>.bs
    <Scene_ID>_<Test_ID>.bs
    <Scene_ID>_<Test_ID>.bs
    <Scene_ID>_<Test_ID>.bs
    ( ... )
( ... )
```

Parameters

| | |
|-------------|--|
| <i>self</i> | Pointer to the renderer instance |
| <i>uid</i> | Unique test/scene identification (see directory structure) |

5.2.4.4 `renderer_setMaxDelay()` `void renderer_setMaxDelay (`
`t_renderer * self,`
`long maxDelay)`

Max calls this function to inform the renderer about the maximum delay in the test.

The PCM input signals for each renderer are delayed by the maximum delay, minus the renderers internal delay value.

This function **must** be bound to the message **setMaxDelay**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_setMaxDelay, "setMaxDelay", A_LONG, 0);
```

Parameters

| | |
|-----------------|---|
| <i>self</i> | Pointer to the renderer instance |
| <i>maxDelay</i> | Maximum delay (samples, integer number) |

5.2.4.5 renderer_setUserPose() void renderer_setUserPose (
 t_renderer * self,
 c74::max::t_symbol * s,
 long argc,
 c74::max::t_atom * argv)

Max calls this function to provide the renderer with the latest position/orientation of the user (= head)

This function **must** be bound to the message **userPose**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_setUserPose, "userPose", A_GIMME, 0);
```

Important: OpenGL coordinate system considered here!

- WCS = world coordinate system (= virtual world)
- RCS = real coordinate system (= tracker)

Note: A_GIMME is used here instead of multiple A_FLOATs (see https://cycling74.com/sdk/Max4-SDK-6.1.1/html/chapter_atoms.html)

Parameters

| | |
|-------------|--|
| <i>self</i> | Pointer to the renderer instance |
| <i>s</i> | Pointer to the symbol that holds the message name |
| <i>argc</i> | Number of arguments in the A_GIMME message (shall be 13) |
| <i>argv</i> | Pointer to arguments <ul style="list-style-type: none"> • User x-coordinate within the virtual world [m] (A_FLOAT) • User y-coordinate within the virtual world [m] (A_FLOAT) • User z-coordinate within the virtual world [m] (A_FLOAT) • User rotation around y-axis (yaw) within the virtual world [deg] (A_FLOAT) • User rotation around x-axis (pitch) within the virtual world [deg] (A_FLOAT) • User rotation around z-axis (roll) within the virtual world [deg] (A_FLOAT) |

Parameters

| | |
|-------------|---|
| <i>argv</i> | Pointer to arguments (cont.) <ul style="list-style-type: none"> • User x-coordinate within the real world [m] (A_FLOAT) • User y-coordinate within the real world [m] (A_FLOAT) • User z-coordinate within the real world [m] (A_FLOAT) • User rotation around y-axis (yaw) within the real world [deg] (A_FLOAT) • User rotation around x-axis (pitch) within the real world [deg] (A_FLOAT) • User rotation around z-axis (roll) within the real world [deg] (A_FLOAT) • Teleport intended? (0 = continuous motion, 1 = immediate teleport) [m] (A_LONG) |
|-------------|---|

5.2.4.6 `renderer_triggerUpdate()` `void renderer_triggerUpdate (`
`t_renderer * self,`
`c74::max::t_symbol * s,`
`long argc,`
`c74::max::t_atom * argv)`

Trigger an update event (L2/L3)

The first argument to this function **MUST** be the update identifier (LONG). If only one parameter is passed into this function (i.e. `argc==1`), the update is of Level2. Additional arguments **MUST** be of type FLOAT and correspond to the update definition conveyed in the bitstream

This function **must** be bound to the message **triggerUpdate**, i.e.

```
// Register method at class
class_addmethod(renderer_class, (method) renderer_triggerUpdate, "triggerUpdate", A_GIMME, 0);
```

Parameters

| | |
|-------------|---|
| <i>self</i> | pointer to msp-object, provided by the runtime |
| <i>s</i> | name of the function, provided by the runtime |
| <i>argc</i> | number of arguments |
| <i>argv</i> | argument list; first argument MUST be the update ID <ul style="list-style-type: none"> • Update identifier (A_LONG) (index in EIF) • ... Update data (A_FLOAT) |

6 File Documentation

6.1 mpeg-renderer_tilde.h File Reference

```
#include "c74_msp.h"
```

Typedefs

- typedef struct _renderer [t_renderer](#)
- typedef enum [_renderer_status_code](#) STATUS_CODE
Renderer status code.

Enumerations

- enum [_renderer_status_code](#) { RENDERER_SUCCESS = 0, RENDERER_ERROR = 1, RENDERER_INVALID_UPDATE = 2, RENDERER_BUSY = -1 }
Renderer status code.

Functions

- void * [renderer_new](#) (c74::max::t_symbol *sym, long argc, c74::max::t_atom *argv)
Max/MSP callback for a new renderer object instance.
- void [renderer_free](#) ([t_renderer](#) *self)
Max/MSP callback for deletion of a renderer object instance.
- void [renderer_assist](#) ([t_renderer](#) *self, void *unused, long io, long index, char *string_dest)
Max/MSP callback to print mouse-over help messages on the inlets and outlets.
- void [renderer_dsp64](#) ([t_renderer](#) *self, c74::max::t_object *dsp64, short *count, double samplerate, long maxvectorsize, long flags)
Max/MSP callback for initialization of audio processing.
- void [renderer_perform64](#) (c74::max::t_object *self, c74::max::t_object *dsp64, double **ins, long numins, double **outs, long numouts, long sampleframes, long flags, void *userparam)
Max/MSP callback for main audio processing.
- void [renderer_loadScene](#) ([t_renderer](#) *self, c74::max::t_symbol uid)
Load a scene.
- void [renderer_loadLSDF](#) ([t_renderer](#) *self, c74::max::t_symbol lsdfPathAndFilename)
Load an LSDF.
- void [renderer_setMaxDelay](#) ([t_renderer](#) *self, long maxDelay)
Max calls this function to inform the renderer about the maximum delay in the test.
- void [renderer_triggerUpdate](#) ([t_renderer](#) *self, c74::max::t_symbol *s, long argc, c74::max::t_atom *argv)
Trigger an update event (L2/L3)
- void [renderer_setUserPose](#) ([t_renderer](#) *self, c74::max::t_symbol *s, long argc, c74::max::t_atom *argv)
Max calls this function to provide the renderer with the latest position/orientation of the user (= head)
- void [renderer_loadHRIIRSet](#) ([t_renderer](#) *self, c74::max::t_symbol *sofaFilePath)
Load head related impulse responses from a sofa file.

6.1.1 Typedef Documentation

6.1.1.1 t_renderer `typedef struct _renderer t_renderer`

6.2 win/get_module_file_name.h File Reference

```
#include <Windows.h>
#include <string>
```

Functions

- `std::string get_module_file_name ()`
Get absolute path of the current module.

6.2.1 Function Documentation

6.2.1.1 get_module_file_name() `std::string get_module_file_name () [inline]`

Get absolute path of the current module.

Returns

Absolute path to module on success, else empty string

Index

- `_renderer_status_code`
 - MPEG-I Audio CfP submission interface, [6](#)
- `get_module_file_name`
 - `get_module_file_name.h`, [12](#)
- `get_module_file_name.h`
 - `get_module_file_name`, [12](#)
- Max API, [3](#)
 - `renderer_assist`, [3](#)
 - `renderer_dsp64`, [3](#)
 - `renderer_free`, [4](#)
 - `renderer_new`, [4](#)
 - `renderer_perform64`, [5](#)
- MPEG-I Audio CfP submission interface, [6](#)
 - `_renderer_status_code`, [6](#)
 - `RENDERER_BUSY`, [7](#)
 - `RENDERER_ERROR`, [7](#)
 - `RENDERER_INVALID_UPDATE`, [7](#)
 - `renderer_loadHRIRSet`, [7](#)
 - `renderer_loadLSDF`, [7](#)
 - `renderer_loadScene`, [8](#)
 - `renderer_setMaxDelay`, [8](#)
 - `renderer_setUserPose`, [9](#)
 - `RENDERER_SUCCESS`, [7](#)
 - `renderer_triggerUpdate`, [10](#)
 - `STATUS_CODE`, [6](#)
- `mpegi-renderer_tilde.h`, [11](#)
 - `t_renderer`, [11](#)
- `renderer_assist`
 - Max API, [3](#)
- `RENDERER_BUSY`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_dsp64`
 - Max API, [3](#)
- `RENDERER_ERROR`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_free`
 - Max API, [4](#)
- `RENDERER_INVALID_UPDATE`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_loadHRIRSet`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_loadLSDF`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_loadScene`
 - MPEG-I Audio CfP submission interface, [8](#)
- `renderer_new`
 - Max API, [4](#)
- `renderer_perform64`
 - Max API, [5](#)
- `renderer_setMaxDelay`
 - MPEG-I Audio CfP submission interface, [8](#)
- `renderer_setUserPose`
 - MPEG-I Audio CfP submission interface, [9](#)
- `RENDERER_SUCCESS`
 - MPEG-I Audio CfP submission interface, [7](#)
- `renderer_triggerUpdate`
 - MPEG-I Audio CfP submission interface, [10](#)
- `STATUS_CODE`
 - MPEG-I Audio CfP submission interface, [6](#)
- `t_renderer`
 - `mpegi-renderer_tilde.h`, [11](#)
- `win/get_module_file_name.h`, [12](#)