OpenSL ES™ is a royalty-free, cross-platform, hardware-accelerated audio API tuned for embedded systems. It provides a standardized, high-performance, low-latency method to access audio functionality for developers of native applications on embedded mobile multimedia devices, enabling straightforward cross-platform deployment of hardware and software audio capabilities, reducing implementation effort, and promoting the market for advanced audio.

• [n] refers to a section in the OpenSL ES 1.1 Specification at www.khronos.org/opensles
• [n] refers to a section for the analogous interface in the OpenMAX AL 1.1 Specification at www.khronos.org/openmax

Object-Interface Mapping Table

This table describes the object-interface mapping and mandated objects per profile.

• The top row shows whether objects are mandated or optional in the profiles.
• The second row lists the objects available in OpenSL ES.
• The left column shows the OpenSL ES interfaces.
• The center columns indicate the object-interface mapping.
• The right column shows analogous interfaces in OpenMAX AL when applicable.

<table>
<thead>
<tr>
<th>INTERFACE</th>
<th>PROFILE</th>
<th>OBJECT</th>
<th>ENGINE</th>
<th>AUDIO PLAYER</th>
<th>MIDI PLAYER</th>
<th>AUDIO RECORDER</th>
<th>LISTENER</th>
<th>3D GROUP</th>
<th>OUTPUT MIX</th>
<th>VIBRA</th>
<th>LED ARRAY</th>
<th>METADATA EXTRACTOR</th>
<th>OpenMAX AL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLObjectItf [8.34]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDLdynamicInterfaceManagementItf [8.17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLEngineItf [8.21]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLEngineCapabilitiesItf [8.22]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLThreadSyncItf [8.44]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLAudioDeviceCapabilitiesItf [8.12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLAudioDecoderCapabilitiesItf [8.9]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLAudioEncoderCapabilitiesItf [8.11]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLCfgExtensionsItf [8.15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLEArrayItf [8.25]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLVibraItf [8.45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPLplayItf [8.37]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLPrecordItf [8.42]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLAudioEncoderItf [8.10]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLPrefetchStatusItf [8.39]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLSampleItf [8.43]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLPQueuebwItf [8.38]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLRatePitchItf [8.41]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLPitchItf [8.36]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLVolumeItf [8.48]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLMuteSoloItf [8.33]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLLBufferQueueItf [8.14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLMIDIMessageItf [8.29]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLMIDITimeItf [8.32]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLMIDITempoItf [8.31]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLMIDIMuteSoloItf [8.30]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DCommitItf [8.2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DGroupingItf [8.4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DHintItf [8.5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DLocationItf [8.6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DSOURCEItf [8.8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DDopplerItf [8.3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DMacroscopicItf [8.7]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DEffectsendItf [8.20]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DBassBoostItf [8.13]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLEqualizerItf [8.24]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DStdReverbItf [8.15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DLatreverbItf [8.23]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLVirtualizerItf [8.46]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DConfigExtensionsItf [8.26]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DMetadataExtractionItf [8.27]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DMetadataMessageItf [8.28]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DTraversalItf [8.29]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DVisualizationItf [8.47]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SL3DOutputMixerItf [8.35]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLDynamicSourceChangeItf [8.18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLDynamicSourceSinkChangeItf [8.19]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLDVEXItf [8.16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A

Arbitrary loop points are not mandated in this profile, only end-to-end looping is mandated.

1

Explicit interface mandated for all players excluding those with Java Tone Sequences (JTS) data sources.

2

Explicit interface mandated only where data source locator is a buffer queue (SLLDataLocator_{BufferQueue, MIDIBufferQueue}).

©2011 Khronos Group - Rev. 0511

www.khronos.org/opensles
## Functions

Following are the functions available in OpenSL ES along with their parameters.

### slCreateEngine() [6.1]

Initializes the engine object and gives the user a handle.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pEngine</td>
<td>Pointer to the resulting engine object.</td>
</tr>
<tr>
<td>numOptions</td>
<td>Number of elements in the options array.</td>
</tr>
<tr>
<td>pEngineOptions</td>
<td>Array of optional configuration data.</td>
</tr>
<tr>
<td>numInterfaces</td>
<td>Number of interfaces that the object is requested to support (not including implicit interfaces).</td>
</tr>
<tr>
<td>plnInterfaces</td>
<td>An array of numinterfaces interface IDs, which the object should support.</td>
</tr>
<tr>
<td>plnfaceRequired</td>
<td>Array of numinterfaces flags, each specifying whether the flag is required on the object or optional.</td>
</tr>
</tbody>
</table>

### slQueryNumSupportedEngineInterfaces() [6.2]

Queries the number of supported interfaces on an object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNumSupportedInterfaces</td>
<td>Identifies the number of supported interfaces available.</td>
</tr>
</tbody>
</table>

### slQuerySupportedEngineInterfaces() [6.3]

Queries the supported interfaces on engine object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Index used to enumerate available interfaces.</td>
</tr>
<tr>
<td>plnfaceid</td>
<td>Identifies supported interface corresponding to the given index.</td>
</tr>
</tbody>
</table>

## Profiles

OpenSL ES is segmented into three profiles: Phone, Music and Game. Combinations of these three profiles are also possible—for example a full-featured game-and-music mobile phone would incorporate all three profiles. Recording functionality, which is commonly used for recording voice memos on mobile phones, is an optional feature and is not part of any profile.

### Phone

This is the basic profile designed for the low-end or “basic” mobile phone market segment. This includes ringtones and alert tone playback (basic MIDI functionality), basic audio playback, and simple 2D audio games.

### Music

This profile is designed for the music-centric mobile device market. Characteristics of such devices include high-quality audio, the ability to support multiple music audio codecs, and the ability to play music from local files. Some high-end devices could also support streaming audio from remote servers (although this is not mandated functionality in OpenSL ES). A mobile phone that has a built-in music player would incorporate both the Phone and Music profiles. A digital music-only mobile device would use only the Music profile.

### Game

This profile is designed for the game-centric mobile device market. Characteristics of such devices include advanced MIDI functionality, and sophisticated audio capabilities such as 3D audio, audio effects, and the ability to handle buffers of audio. A mobile phone that offers sophisticated game-playing ability would incorporate both the Phone and Game profiles. A game-only device would use only the Game profile.

## Interfaces

Refer to the Object-Interview Mapping Table (page 1) to see the objects to which each interface applies.

### SL3DCommitItf [8.2]

Controls 3D interface commit policy.

Object: Engine

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Commits changes to 3D interfaces.</td>
</tr>
<tr>
<td>SetDeferred</td>
<td>Enables/Disables deferred committing of 3D parameters.</td>
</tr>
</tbody>
</table>

### SL3DDopplerItf [8.3]

Controls Doppler for listener, player, or 3D group.

Objects: Audio Player, Midi Player, Listener, 3D Group

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetVelocityCartesian</td>
<td>Sets the object’s velocity using Cartesian coordinates.</td>
</tr>
<tr>
<td>GetVelocityCartesian</td>
<td>Gets the object’s velocity.</td>
</tr>
<tr>
<td>SetVelocitySpherical</td>
<td>Sets the object’s velocity using spherical coordinates.</td>
</tr>
<tr>
<td>GetVelocitySpherical</td>
<td>Gets the object’s velocity.</td>
</tr>
<tr>
<td>SetDopplerFactor</td>
<td>Sets the object’s Doppler factor.</td>
</tr>
<tr>
<td>GetDopplerFactor</td>
<td>Gets the object’s Doppler factor.</td>
</tr>
</tbody>
</table>

### SL3DGroupingItf [8.4]

Sets the player’s 3D group.

Objects: Audio Player, Midi Player

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set3DGroup</td>
<td>Sets the 3D group for player, removing player from previous 3D group.</td>
</tr>
<tr>
<td>Get3DGroup</td>
<td>Gets the 3D group for the player.</td>
</tr>
</tbody>
</table>

### SL3DHintItf [8.5]

Defines the rendering quality hint.

Object: Audio Player, Midi Player, 3D Group

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetRenderHint</td>
<td>Sets 3D source’s rendering quality hint.</td>
</tr>
<tr>
<td>GetRenderHint</td>
<td>Retrieves the rendering quality hint.</td>
</tr>
</tbody>
</table>

### SL3DLocationItf [8.6]

Controls location and orientation.

Objects: Audio Player, Midi Player, Listener, 3D Group

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetLocationCartesian</td>
<td>Sets the 3D location using Cartesian coordinates.</td>
</tr>
<tr>
<td>GetLocationCartesian</td>
<td>Gets the 3D location using Cartesian coordinates.</td>
</tr>
<tr>
<td>SetLocationSpherical</td>
<td>Sets the 3D location using spherical coordinates.</td>
</tr>
<tr>
<td>GetLocationSpherical</td>
<td>Gets the 3D location using spherical coordinates.</td>
</tr>
<tr>
<td>Move</td>
<td>Moves the object.</td>
</tr>
<tr>
<td>GetOrientationCartesian</td>
<td>Gets the 3D location using Cartesian coordinates.</td>
</tr>
<tr>
<td>SetOrientationCartesian</td>
<td>Sets 3D orientation using vectors.</td>
</tr>
<tr>
<td>GetOrientationCartesian</td>
<td>Gets the sound cones.</td>
</tr>
<tr>
<td>Rotate</td>
<td>Rotates the object.</td>
</tr>
<tr>
<td>GetOrientationVectors</td>
<td>Gets 3D orientation as vectors.</td>
</tr>
</tbody>
</table>

### SL3DMacroscopicItf [8.7]

Controls the size of a 3D sound source.

Objects: Audio Player, Midi Player, 3D Group

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetSize</td>
<td>Sets size of 3D sound source.</td>
</tr>
<tr>
<td>GetSize</td>
<td>Gets size of 3D sound source.</td>
</tr>
<tr>
<td>SetOrientationAngles</td>
<td>Sets the 3D orientation of volume using angles.</td>
</tr>
<tr>
<td>GetOrientationAngles</td>
<td>Gets the 3D orientation of volume using vectors.</td>
</tr>
<tr>
<td>Rotate</td>
<td>Rotates macroscopic volume.</td>
</tr>
<tr>
<td>GetOrientationVectors</td>
<td>Gets the 3D orientation.</td>
</tr>
</tbody>
</table>

### SL3DSourcetf [8.8]

Controls 3D parameters unique to 3D sources.

Objects: Audio Player, Midi Player, 3D Group

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetHeadRelative</td>
<td>Sets whether 3D source is head relative.</td>
</tr>
<tr>
<td>GetHeadRelative</td>
<td>Gets the 3D source’s head relative state.</td>
</tr>
<tr>
<td>SetRolloffDistances</td>
<td>Sets the min/max rolloff distances.</td>
</tr>
<tr>
<td>GetRolloffDistances</td>
<td>Gets the min/max rolloff distances.</td>
</tr>
<tr>
<td>SetRolloffMaxDistanceMute</td>
<td>Sets mute policy beyond rolloff distance.</td>
</tr>
<tr>
<td>GetRolloffMaxDistanceMute</td>
<td>Gets mute policy beyond rolloff distance.</td>
</tr>
<tr>
<td>SetRolloffFactor</td>
<td>Sets the distance rolloff factor.</td>
</tr>
<tr>
<td>GetRolloffFactor</td>
<td>Gets the distance rolloff factor.</td>
</tr>
<tr>
<td>SetRoomRolloffFactor</td>
<td>Sets the room rolloff factor.</td>
</tr>
<tr>
<td>GetRoomRolloffFactor</td>
<td>Gets the distance room rolloff factor.</td>
</tr>
<tr>
<td>SetRolloffModel</td>
<td>Sets distance decay rolloff model.</td>
</tr>
<tr>
<td>GetRolloffModel</td>
<td>Gets the distance rolloff model.</td>
</tr>
<tr>
<td>SetCone</td>
<td>Sets the sound cones.</td>
</tr>
<tr>
<td>GetCone</td>
<td>Gets the sound cones.</td>
</tr>
</tbody>
</table>

### SLAudioDecoderCapabilitiesItf [8.9]

Queries the engine decode capabilities.

Object: Engine

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAudioDecoders</td>
<td>Retrieves the available audio decoders.</td>
</tr>
<tr>
<td>GetAudioDecoderCapabilities</td>
<td>Queries the audio decoder capabilities.</td>
</tr>
</tbody>
</table>

### SLAudioEncoderItf [8.10]

Sets audio encoder parameters.

Object: Audio Recorder

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetEncoderSettings</td>
<td>Set audio encoder settings.</td>
</tr>
<tr>
<td>GetEncoderSettings</td>
<td>Get audio encoder settings.</td>
</tr>
</tbody>
</table>

### SLAudioEncoderCapabilitiesItf [8.11]

Queries audio encoding capabilities.

Object: Audio Recorder

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAudioEncoders</td>
<td>Queries the supported audio encoders.</td>
</tr>
<tr>
<td>GetAudioEncoderCapabilities</td>
<td>Queries the audio encoder capabilities.</td>
</tr>
</tbody>
</table>

### SLAudioIODeviceCapabilitiesItf [8.12]

Enumerates audio I/O devices and query capabilities of each available audio I/O device.

Object: Engine

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAvailableAudioInputs</td>
<td>Gets number and IDs of audio input devices.</td>
</tr>
<tr>
<td>GetAvailableAudioOutputs</td>
<td>Gets the number and IDs of audio output devices.</td>
</tr>
<tr>
<td>QueryAudioInputCapabilities</td>
<td>Gets the capabilities of the specified audio input device.</td>
</tr>
<tr>
<td>RegisterAvailableAudioInputsChangedCallback</td>
<td>Sets/clears sIAVAILABLEAUDIOINPUTSCHANGEDCALLBACK().</td>
</tr>
<tr>
<td>RegisterAvailableAudioOutputsChangedCallback</td>
<td>Sets/clears sIAVAILABLEAUDIOOUTPUTSCHANGEDCALLBACK().</td>
</tr>
<tr>
<td>GetAssociatedAudioInputs</td>
<td>Returns array of audio input devices physically associated with this I/O device.</td>
</tr>
<tr>
<td>GetAssociatedAudioOutputs</td>
<td>Returns array of audio output devices physically associated with this I/O device.</td>
</tr>
<tr>
<td>QueryAudioOutputCapabilities</td>
<td>Gets the capabilities of an audio output device.</td>
</tr>
<tr>
<td>RegisterAvailableAudioOutputsChangedCallback</td>
<td>Sets/clears sIAVAILABLEAUDIOOUTPUTSCHANGEDCALLBACK().</td>
</tr>
<tr>
<td>RegisterDefaultDeviceIDMapChangedCallback</td>
<td>Sets/clears sIAVAILABLEAUDIODVICEIDMAPPATCHEDCALLBACK().</td>
</tr>
<tr>
<td>GetDefaultAudioDevices</td>
<td>Gets the number of audio devices currently mapped to the given default device ID.</td>
</tr>
<tr>
<td>GetDefaultAudioDevices</td>
<td>Gets the number of audio devices currently mapped to the given default device ID.</td>
</tr>
<tr>
<td>QuerySampleFormatsSupported</td>
<td>Gets array of sample formats supported by the audio I/O device for the given sampling rate.</td>
</tr>
</tbody>
</table>

©2011 Khronos Group - Rev. 0511 www.khronos.org/opensles
SLBassBoostItf [8.13]
Controls bass boost functionality.
Objects: Audio Player, Midi Player, Audio Recorder, Output Mix

- SetEnabled Enables the effect.
- IsEnabled Gets the enabled status.
- SetStrength Sets the strength of the effect.
- GetRoundedStrength Gets the strength of the effect.
- IsStrengthSupported Indicates if setting strength is supported.

SLBufferQueueItf [8.14]
Streams audio data.
Objects: Audio Player, Midi Player, Audio Recorder

- Enqueue Adds a buffer to the queue.
- Clear Releases all queued buffers.
- GetState Returns current state of buffer queue.
- RegisterCallback Sets callback function to be called on buffer completion.
- SetCallbackEventsMask Enables/disables notification of buffer queue events.
- GetCallbackEventsMask Queries the notification state of buffer queue events.

SLConfigExtensionsItf [8.15]
Sets/query the configuration of the audio engine.
Objects: All

- SetConfiguration Sets configuration as a key-value pair.
- GetConfiguration Gets the configuration setting as a key-value pair.

SLDeviceVolumeItf [8.16]
Exposes controls for manipulating the volume of audio I/O devices.
Object: Engine

- GetVolumeScale Gets the volume scale properties.
- SetVolume Sets the volume level.
- GetVolume Gets the volume level.

SLDynamicInterfaceManagementItf [8.17]
Manages interface exposure on a created and realized object.
Objects: All

- AddInterface Asynchronously exposes an interface.
- RemoveInterface Removes a dynamically exposed interface.
- ResumeInterface Resumes a dynamically exposed interface.
- RegisterCallback Callback for various interface errors or events.

SLDynamicSourceItf [8.18]
Deprecated. Instead use SLDynamicSourceSinkChangeItf.

SLDynamicSourceSinkChangeItf [8.19]
Object: Metadata Extractor
Changes data source or sink during object lifetime.

- ChangeSource Changes an audio data source.
- ChangeSink Changes a data sink.
- RegisterSourceChangeCallback Sets or clears SLSourceChangeCallback.
- RegisterSinkChangeCallback Sets or clears SLSinkChangeCallback.

SLEffectSndItf [8.20]
Controls a sound's contribution to aux effects.
Objects: Audio Player, MIDI Player

- EnableEffectSnd [En/dis]ables player's contribution to an aux effect.
- IsEnabled Returns if output goes to an aux effect.
- SetDirectLevel Sets dry (direct) path level for a sound.
- GetDirectLevel Gets the player's direct path level.
- SetSendLevel Sends path level for an aux effect.
- GetSendLevel Gets player's send path level for an aux effect.

SLEngineItf [8.21]
Exposes creation methods of all object types.
Object: Engine

- CreateLEDDevice Creates an LED device.
- CreateVibraDevice Creates a vibrator device.
- CreateAudioPlayer Creates an audio player.
- CreateAudioRecorder Creates an audio recorder.
- CreateMidiPlayer Creates a MIDI player.
- CreateListener Creates a listener.
- Create3DGroup Creates a 3D group.
- CreateOutputMix Creates an output mix.

- CreateMetadataExtractor Creates a Metadata Extractor.
- CreateExtensionObject Creates an object.
- QueryNumSupportedInterfaces Queries the number of supported interfaces.
- QuerySupportedInterfaces Queries supported interfaces.
- QueryNumSupportedExtensions Queries the number of supported extensions.
- QuerySupportedExtension Queries if extension is supported.
- QueryIsExtensionSupported Queries if extension is supported.
- QueryPresetName Gets the preset name based on the index.

SLEnvironmentalReverbItf [8.23]
Controls properties in a global reverb environment.
Object: Engine

- GetEnvironmentalReverbProperties Returns state of buffer queue.
- Clear Clears buffer queue.
- Enqueue Adds a buffer to the queue.
- GetConfiguration Gets the configuration settings.
- SetConfiguration Sets the configuration settings.

- GetCallbackEventsMask Queries callback events.
- SetCallbackEventsMask Enables/disables notification of buffer queue events.

SLEqualizerItf [8.24]
Manipulates equalizer settings.
Objects: Audio Player, Midi Player, Audio Recorder, Output Mix

- IsEnabled Gets enabled status of the effect.
- GetNumberOfBands Gets number of equalizer bands.
- GetBandLevelRange Returns min/max band levels.
- GetBandLevel Gets a band's gain level.
- GetCenterFreq Gets a band's center frequency.
- GetBandFreqRange Gets a band's frequency range.
- GetBand Gets the band that affect a frequency the most.
- GetCurrentFreq Gets the current preset.
- UsePresets Sets the equalizer according to the given preset.
- GetNumberOfPresets Gets the number of presets supported.

SLEnvironmentalReverbProperties (continued)
Sets all environment reverb properties.

- GetReverbLevel Gets the late reverberation volume level.
- SetReverbLevel Sets the late reverberation volume level.
- GetReverbDelay Gets the reverb delay length.
- SetReverbDelay Sets the reverb delay length.
- GetDiffusion Gets the level of diffusion.
- SetDiffusion Sets the late reverberation decay echo density.
- GetDensity Gets the density level.
- SetDensity Controls the late reverberation decay modal density.

SLLedArrayItf [8.25]
Sets LED array state and color.
Object: LED Array

- ActivateLEDArray Activates or deactivates individual LEDs.
- IsLEDArrayActivated Returns the state of each LED.
- SetColor Sets color of an individual LED.
- GetColor Gets color of an individual LED.

SLMetadataExtractionItf [8.26]
Extracts metadata.
Objects: Audio Player, Midi Player, Metadata Extractor

- GetItemCount Returns the number of metadata items.
- GetKey Returns metadata by key.
- GetKeyFilter Returns metadata by key.
- GetValue Returns metadata by value.
- GetValueSize Gets the byte size of a given metadata value.
- GetPresetName Gets a preset name from the index.
- SetPresetName Sets the preset name based on the index.

SLMetadataMessageItf [8.27]
Sets metadata callbacks during playback.
Objects: Audio Player, Midi Player, Metadata Extractor

- RegisterCallback Sets or clears the metadata callbacks.
Interfaces (continued)

**SLMetadataTraversalItf [8.28]**
Traverses a file’s metadata.
- **SetMode** Sets the metadata traversal mode.
- **GetChildCount** Returns the number of child nodes in scope.
- **GetChildMIMETYPESize** Returns a child’s MIME size.
- **GetChildInfo** Returns information about a child.
- **GetActiveNode** Sets the scope to a child index.

**SLMIDIMessageItf [8.29]**
Sends MIDI messages and sets MIDI callbacks.
- **SendMessage** Sends a MIDI message to a player.
- **RegisterMIDIMessageCallback** Sets or clears a MIDI message callback.
- **AddMIDIMessageCallbackFilter** Adds a message type to a MIDI callback filter.
- **ClearMIDIMessageCallbackFilter** Clears MIDI message callback filter.

**SLMIDIMuteSoloItf [8.30]**
Mutes and solos MIDI channels and tracks.
- **SetChannelMute** Mutes or unmutes a MIDI channel.
- **GetChannelMute** Returns mute state of a MIDI channel.
- **GetChannelSolo** Solos or unsolos a MIDI channel.
- **GetChannelSolo** Returns solo status of a MIDI channel.
- **GetTrackCount** Returns number of MIDI tracks in player’s SMF data.
- **SetTrackMute** Mutes or unmutes a MIDI track.
- **GetTrackMute** Returns mute state of a MIDI track.
- **GetTrackSolo** Solos or unsolos a MIDI track.
- **GetTrackSolo** Returns solo status of a MIDI track.

**SLMIDItempoItf [8.31]**
Manages the MIDI data’s tempo.
- **SetTicksPerQuarterNote** Sets the player’s tick resolution.
- **GetTicksPerQuarterNote** Returns the tick resolution.
- **SetMicrosecondsPerQuarterNote** Sets the player’s tempo.
- **GetMicrosecondsPerQuarterNote** Returns the tempo.

**SLMIDITempoItf [8.32]**
Manages the MIDI data in time (ticks).
- **GetDuration** Returns the duration in MIDI ticks.
- **SetPosition** Sets a player’s position in MIDI ticks.
- **GetPosition** Returns the position in MIDI ticks.
- **SetLoopPoints** Sets loop points in MIDI ticks.
- **GetLoopPoints** Returns loop points in MIDI ticks.

**SLMuteSoloItf [8.33]**
Manages channel mute and solo status.
- **setChannelMute** Mutes or unmutes a channel.
- **GetChannelMute** Retrieves a channel’s mute state.
- **GetChannelSolo** [En/dis]ables soloing of a channel.
- **GetChannelSolo** Retrieves a channel’s soloed state.
- **GetNumChannels** Retrieves the number of audio channels.

**SLOutputMixItf [8.35]**
Manages an output mix object.
- **GetDestinationOutputDeviceIDs** Gets the destination device IDs.
- **RegisterDeviceChangeCallback** Callback for changes to the output device IDs.
- **ReRoute** Changes the specified set of output devices.

**SLPlaybackRateItf [8.38]**
Controls sound pitch shift.
- **GetPitch** Sets a player’s pitch shift.
- **GetPitchCapabilities** Retrieves pitch shifting capabilities.
- **SetPitch** Gets the player’s pitch shift.
- **SetPitchCapabilities** Retrieves pitch shifting capabilities.

**SLPlaybackStateItf [8.37]**
Controls an object’s playback state.
- **SetPlayState** Transitions into the given play state.
- **GetPlayState** Gets the player’s current play state.
- **GetDuration** Gets the duration of the current content.
- **GetPosition** Returns the relative position of the playback head.
- **RegisterCallback** Sets the playback callback function.
- **GetCallbackEventsMask** Gets the destination device IDs.
- **GetMarkerPosition** Clears marker.
- **SetMarkerPosition** Sets the position of the playback marker.
- **GetPosition** Queries the position of playback events.
- **GetPositionUpdatePeriod** Queries the position notification interval.

**SLPresetReverbItf [8.40]**
Controls the sound playback rate.
- **SetRate** Sets a player’s rate.
- **GetRate** Gets the player’s rate.
- **GetRatePitchCapabilities** Retrieves the player’s rate pitch capabilities.

**SLRecordItf [8.42]**
Controls the recording state of an object.
- **SetRecordState** Transitions into the given record state.
- **GetRecordState** Gets the recorder’s record state.
- **GetDuration** Sets the duration of current content.
- **GetPosition** Returns the relative position of the recording head.
- **RegisterCallback** Registers the record callback function.
- **GetCallbackEventsMask** Queries the notification state of record events.
- **GetCallbackEventsMask** Sets the notification state of record events.
- **SetMarkerPosition** Sets the position of the recording marker.
- **ClearMarkerPosition** Clears marker.
- **GetMarkerPosition** Queries the position of the recording marker.
- **SetPositionUpdatePeriod** Sets the position notification interval.
- **GetPositionUpdatePeriod** Queries the position update interval.

**SLSeekItf [8.43]**
Manages a playback head’s position and looping.
- **SetPosition** Sets the position of the playback head.
- **SetLoop** Sets looping parameters.
- **GetLoop** Query looping parameters.

**SLSoundItf [8.44]**
Thread control.
- **EnterCriticalSection** Transitions the engine into critical section state.
- **ExitCriticalSection** Transitions into non-critical section state.

SLOutputMixItf [8.35]
Provides essential utility methods for all objects.
- **Realize** Transitions Unrealized to Realized state.
- **Resume** Transitions Suspended to Realized state.
- **GetState** Retrieves the current object state.
- **GetInterface** Obtains the object’s exposed interface.
- **RegisterCallback** Callback for error or async completion.
- **AbortAsyncOperation** Aborts asynchronous call in progress.
- **Destroy** Destroys the object.
- **SetPriority** Sets the object’s priority.
- **GetPriority** Gets the object’s priority.
- **SetLossOfControllerinterfaces** Sets/unsets loss of control functionality.

SLPresetReverbItf [8.40]
Queries the prefetch status of a player.
- **GetPrefetchStatus** Gets the player’s current prefetch status.
- **GetFillLevel** Queries the fill level of the prefetch.
- **RegisterCallback** Sets the prefetch callback function.
- **GetCallbackEventsMask** Queries the notification state of the prefetch events.
- **GetFillLevelUpdatePeriod** Sets the notification period for fill level updates.
- **GetFillLevelUpdatePeriod** Queries the notification period for fill level updates.
- **GetError** Retrieves the last error code.

Objects: Audio Player, MIDI Player

©2011 Khronos Group - Rev. 0511
**Interfaces (continued)**

**SLVirtualizerItf** [8.46]
Controls the audio virtualizer.
- Objects: Audio Player, MIDI Player, Audio Recorder, Output Mix

**SLVolumeItf** [8.48]
Volume control.
- Objects: Audio Player, MIDI Player, Audio Recorder, Output Mix

**SLVibraItf** [8.45]
Controls the Vibra I/O device.

**SLVibra**
- Activates or deactivates vibration.
- Returns whether the I/O device is vibrating.
- Sets the vibration frequency.
- Returns the vibration frequency.
- Sets the vibration intensity.
- Returns the vibration intensity.

**SLVirtualizerItf** [8.47]
- Gets data for visualization.

**SLVolumeItf** [8.48]
- Controls the audio virtualizer.

**SLVirtualizerItf** [8.46]
- Objects: Audio Player, MIDI Player, Output Mix

**SLVirtualizerItf** [8.46]
- Controls the audio virtualizer.

**SLVolumeItf** [8.48]
- Objects: Audio Player, MIDI Player, Audio Recorder, Output Mix

**Macros**

**SL絷inearBoxing** [9.2.2]
- The encoding audio type.
- PCM, MP3, AMR, AMRV8, AMRV8PLUS, AAC, WMA, REAL, VORBIS

**SL絷inearBoxing** [9.2.3]
- Audio profiles and modes.

**SL絷inearBoxing** [9.2.4]
- A function prototype declaration macro.

**SL絷inearBoxing** [9.2.5]
- An API entry point macro.

**SL絷inearBoxing** [9.2.6]
- Canonical values for boolean type.
- FALSE, TRUE

**SL絷inearBoxing** [9.2.7]
- Flags for a buffer queue event.
- PROCESSED, UNREALIZED, CLEARED, STOPPED, ERROR, CONTENT, END

**SL絷inearBoxing** [9.2.8]
- The byte order of 16- or 32-bit data.

**SL絷inearBoxing** [9.2.9]
- Metadata character encoding.

**Macros continues >**
Macros (Continued)

SL_CONTAINERTYPE_* [9.2.10]
The data source or sink container type.
UNSPECIFIED, RAW, AVI, BMP, JPEG, PNG, DIB, DOL, DTS, DTS-ES, 
DTS-EXPRESS, PCM, ALP ElementRef, ALP_Float, ALP_Float32,
ALP_Float64, ALP_Float16, ALP_Int, ALP_Int8, ALP_Int16,
ALP_Int32, ALP_Int64, ALP_UInt, ALP_UInt8, ALP_UInt16,
ALP_UInt32, ALP_UInt64

SL_DATAFORMAT_* [9.2.11]
The possible data formats.
MIME, PCM, RESERVES, PCM_EX

SL_DATALOCATOR_* [9.2.12]
The possible data locators.
NULL, URI, ADDRESS, IOID, OUTPUTMIX, RESERVES, BUFFERQUEUE, MIDIBUFFERQUEUE,
MEDIADATAOBJECT, CONTENTTYPE

SL_DEFAULTDEVICE_ID * [9.2.13]
Default device IDs.
AUDIOINPUT, AUDIOOUTPUT, LED, VIBRA, RESERVED1

SL_DEVICE_CONNECTION_* [9.2.14]
Types of I/O device connections.
INTEGRATED, ATTACHED_{Wired, Wireless}, NETWORK

SL_DEVICELOCATION_* [9.2.15]
Types of I/O device locations.
HANDSET, HEADSET, CARKIT, DOCK, REMOTE

SL_DEVICESCOPE_* [9.2.16]
The current scope of the node’s parent.
NODE_PARENT

SL_NODE_PARENT [9.2.28]
The current scope of the node’s parent.
NODE_PARENT

SL_NODETYPE_* [9.2.29]
The type of a node.
UNSPECIFIED, AUDIO, VIDEO, IMAGE

SL_OBJECTEVENT_* [9.2.30]
Object event notifications.
RUNTIME_ERROR, ASYNC_TERMINATION, RESOURCES_LOST, RESOURCES_AVAILABLE,
ITF_CONTROL_TAKEN, ITF_CONTROL_RETURNED, ITF_PARAMETERS_CHANGED

SL_OBJECTSTATE_* [9.2.31]
Object states.
UNREALIZED, REALIZED, SUSPENDED

SL_OBJECTID_* [9.2.32]
Object type identifiers.
ENGINE, LEDDEVICE, VIBRADEVICE, AUDIOPLAYER, 
AUDIODECORDER, MIDPLAYER, LISTENER, 
3DGROUP, OUTPUTMIX, MEDIADATATRASCTOR

SL_PCM_REPRESENTATION_* [9.2.33]
PCM data type.
SIGNED_8, UNSIGNED_8, FLOAT

SL_SamplingRate_* [9.2.34]
Audio device sample formats.
FIXED_8, FIXED_16, FIXED_20, FIXED_24, FIXED_32, FIXED_64

SL_PLAYEVENT_* [9.2.35]
Play events.
HEADATEND, HEADATMARKER, HEADATNEWPOS, 
HEADMOVING, HEADSTALLED, BUFFERFULL, BUFFERQUEUE/starved

SL_PLAYSTATE_* [9.2.36]
Playback state.
STOPPED, PAUSED, PLAYING

SL_RECORDEVENT_* [9.2.37]
Record events.
HEADATLIMIT, HEADATMARKER, HEADATNEWPOS, 
HEADMOVING, HEADSTALLED, BUFFER_FULL, BUFFERQUEUE/starved

SL_RECORDSTATE_* [9.2.38]
Object recording state.
STOPPED, PAUSED, RECORDING

SL_RESULT_* [9.2.39]
Return values.
SUCCESS, PRECONDITIONS_VIOLATED, PARAMETER_INVALID, MEMORY_FAILURE, 
RESOURCE_ERROR, RESOURCE_LOST, IO_ERROR, BUFFER_INSUFFICIENT, 
CONTENT_CORRUPTED, CONTENT_UNSUPPORTED, CONTENT_NOT_FOUND, PERMISSION_DENIED, 
FEATURE_UNSUPPORTED, INTERNAL_ERROR, UNKNOWN_ERROR, OPERATION_ABORTED, 
CONTROL_LOST, READONLY, ENGINEOPTION_UNSUPPORTED, 
SOURCE_SINK_INCOMPATIBLE

SL_ROLLOFFMODEL_* [9.2.40]
Rolloff distance models.
EXPONENTIAL, LINEAR

SL_SAMPLINGRATE_* [9.2.41]
Common audio sampling rates.
8, 11_025, 12, 16, 22_05, 24, 32, 44_1, 48, 
64, 88_2, 96, 192

SL_SEEKMODE_* [9.2.42]
Seek modes.
FAST, ACCURATE

SL_SPEAKER_* [9.2.43]
Speaker locations used when specifying channel mask.
FRONT_LEFT, FRONT_RIGHT, CENTER, LOW_FREQUENCY, 
BACK_LEFT, BACK_RIGHT, FRONT_LEFT_OF_CENTER, 
FRONT_RIGHT_OF_CENTER, SIDE_LEFT, SIDE_RIGHT, 
TOP_FRONT, TOP_LEFT, TOP_RIGHT, TOP_BACK, 
TOP_BACK_LEFT, TOP_BACK_RIGHT

SL_TIME_* [9.2.44]
Out of range playback time.
SL_TIME_UNKNOWN

SL_VOICETYPE_* [9.2.45]
Voice types.
2D_AUDIO, MIDI, 3D_AUDIO, 3D_MIDIOUTPUT
Object State Diagram [3.1.1]

This diagram illustrates the object states and state transitions. When the application destroys an object, the object implicitly transitions through the Unrealized state. During the transition, it frees its resources and makes them available to other objects. Every object maintains a state machine with the following states:

- **Unrealized (initial state)**: The object is alive but has not yet allocated any resources. It is not usable, and its interfaces’ methods cannot be called.
- **Realized**: The object’s resources are allocated and the object is usable.
- **Suspending (optional state)**: The object has fewer resources than required to be usable, but it maintains the state it was in at the time of suspension. The system has the option of putting an object either in the Suspending state or the Unrealized state when resources are insufficient.

--- System-induced transitions → Client-induced transitions

Use Case: Sampled Audio Playback [4.6.1]

This example illustrates basic audio playback. An Audio Player is created using the SL::EngineItf interface of the Engine object. Upon creation, the Audio Player is associated with an Output Mix, created using the SL::EngineItf interface, for audio output. The data source of the Audio Player is also set during creation. The data source is a URI pointing to an audio file on the local file system. The Output Mix is by default associated with the system-dependent default output device.

Use Case: 3D Audio [4.6.3]

This example illustrates positional 3D audio rendering using two sampled audio players simultaneously. Both Audio Player objects are created using the SL::EngineItf interface of the Engine object. Upon creation, both Audio Players are associated with the same Output Mix for audio output. Requesting the SL::3DLocationItf interfaces from the Audio Players upon their creation causes them to be rendered as 3D sources.

- The virtual listener is controlled with a Listener object which is created using the SL::EngineItf interface of the engine object. The reverberation of the virtual acoustical space is controlled by the SL::EnvironmentalReverbItf interface of the Output Mix. The SL::EffectSendItf interfaces are exposed on the Audio Players to feed the audio signals to the reverberator of the Output Mix.

Use Case: Recording Audio [4.6.4]

This example illustrates basic audio recording. Recording audio is handled by an Audio Recorder object, created using the SL::EngineItf interface of the engine object. Upon creation, it is associated with an audio data source, in this case a microphone. The data sink of the Audio Recorder is a URI pointing to an audio file in the local file system to which the audio will be recorded.

Use Case: Reading Metadata [4.6.5]

This example illustrates reading the metadata from a file without playback. A Metadata Extractor object reads the metadata of an audio file without allocating resources for audio playback. The Metadata Extractor object is created using the SL::EngineItf interface of the engine object and the data source is set. The data source is a URI pointing to an audio file in the local file system. The SL::MetadataExtractionItf and SL::MetadataTraversalItf interfaces are used for reading and traversing the metadata from the file.

The Metadata Extractor supports the SL::DynamicSourceSinkChangeItf interface which can be used to change the data source. Therefore metadata can be extracted from multiple files (in series) without creating a new Metadata Extractor object for every single file. The SL::MetaDataMessageItf interface is used to set callbacks that execute whenever a metadata item is encountered.
Use Case: MIDI Playback [4.6.2]
This example illustrates the use of OpenSL ES objects in a typical audio player use case using a MIDI Player object for audio playback.

The MIDI Player is created using the SLEngineItf interface of the Engine object. Upon creation, the MIDI Player is associated with an Output Mix for audio output. The data source of the MIDI Player is also set during creation. The Output Mix is by default associated with the system-dependent default output device.

Use Case: Music Interrupted By Message [C3]
This example illustrates how to handle music playback on a headset when a message notification is played.
An incoming message alert requires MIDI to begin. The music playing on the headset is stopped after storing the music play position. A new MIDI player is created to generate a message alert from a file, directing the audio to a different IODevice output, the phone handsfree speaker. When the MIDI player has completed playing, the MIDI player is destroyed and music playback is resumed on the headset from where it was stopped.
See sample code to accompany this illustration in the specification [C.3.2].

Use Case: Grouping 3D Movement [C4]
This example illustrates a combining a stationary 3D-positioned MIDI sound source with two moving PCM sound sources.
The two PCM sources, the sounds of a car engine and siren, are part of the same moving object and a 3DGroup is used to control their location as one. The listener is stationary, looking forward, as the moving sounds travel at a speed of 50kph from the left to the right. The location of the listener is set using the listener object. The 3DGroup object is used to control the location and Doppler of the car as it passes the listener. The environmental reverb and master volume are controlled using the Output mix object.
See sample code to accompany this illustration in the specification [C.4.2].