# SOUND4 IMPACT.CL Library [1.2.7]

Generated by Doxygen 1.9.1

**1 SOUND4 IMPACT.CL Library** 1
1.1 General usage . . . 1
1.2 Examples . . . 1

**2 Deprecated List** 3

**3 Module Index** 5
3.1 Modules . . . 5

**4 Namespace Index** 7
4.1 Namespace List . . . 7

**5 Hierarchical Index** 9
5.1 Class Hierarchy . . . 9

**6 Class Index** 13
6.1 Class List . . . 13

**7 File Index** 15
7.1 File List . . . 15

**8 Module Documentation** 17
8.1 Sample using C interface . . . 17
8.1.1 Sample to create the processing instance . . . 17
8.1.2 Sample processing loop . . . 18
8.1.3 Sample to stop the processing instance . . . 18
8.2 Environment variables and parameters . . . 19
8.2.1 Parameters defaulting from environment . . . 19
8.2.1.1 Webserver authentification setup : . . . 19
8.2.1.2 Webserver secure : . . . 19
8.2.1.3 Network access : . . . 19
8.2.2 Parameters . . . 19
8.2.3 Preset sharing . . . 20
8.3 Global setup . . . 20
8.3.1 Detailed Description . . . 21
8.3.2 Macro Definition Documentation . . . 21
8.3.2.1 IMPACT_AUDIO_INPUT_COUNT . . . 21
8.3.2.2 IMPACT_AUDIO_OUTPUT_COUNT . . . 21
8.3.2.3 IMPACT_AUDIOFRAME_COUNT . . . 21
8.3.2.4 IMPACT_CHANNEL_COUNT . . . 22
8.3.2.5 IMPACT_SAMPLE_RATE . . . 22
8.3.3 Typedef Documentation . . . 22
8.3.3.1 impact_loggerfn . . . 22
8.3.4 Enumeration Type Documentation . . . 22
8.3.4.1 impact_LogSeverity . . . 22

8.3.5 Function Documentation . . . 23
8.3.5.1 impact_GetAudioInputCount() . . . 23
8.3.5.2 impact_GetAudioOutputCount() . . . 23
8.3.5.3 impact_GetChannelCount() . . . 24
8.3.5.4 impact_GetChunkSizeInFrames() . . . 24
8.3.5.5 impact_GetPossibleChunkSizeInFrames() . . . 24
8.3.5.6 impact_GetSampleRate() . . . 25
8.3.5.7 impact_GetVersion() . . . 25
8.3.5.8 impact_SetLoggerCallback() . . . 25
8.3.5.9 impact_SetLogSeverity() . . . 26
8.3.5.10 SOUND4_GetProcessInfo() . . . 26
8.4 Process creation/destruction . . . 26
8.4.1 Detailed Description . . . 27
8.4.2 Function Documentation . . . 27
8.4.2.1 impact_ExitProcess() . . . 27
8.4.2.2 impact_FreeParameters() . . . 27
8.4.2.3 impact_FreeParameterValue() . . . 27
8.4.2.4 impact_GetParameter() . . . 27
8.4.2.5 impact_GetProcessChunkFrames() . . . 28
8.4.2.6 impact_InitProcess() . . . 28
8.4.2.7 impact_InitProcess2() . . . 29
8.4.2.8 impact_InitProcess3() . . . 29
8.4.2.9 impact_NewParameters() . . . 30
8.4.2.10 impact_SetParameter() . . . 31
8.4.2.11 impact_TerminateProcess() . . . 31
8.5 Custom preset managagement . . . 31
8.5.1 Detailed Description . . . 32
8.5.2 Typedef Documentation . . . 32
8.5.2.1 fs_char . . . 32
8.5.2.2 impact_storage_exists . . . 32
8.5.2.3 impact_storage_getall . . . 33
8.5.2.4 impact_storage_getall_free . . . 33
8.5.2.5 impact_storage_reader . . . 33
8.5.2.6 impact_storage_reader_free . . . 34
8.5.2.7 impact_storage_remove . . . 34
8.5.2.8 impact_storage_rename . . . 35
8.5.2.9 impact_storage_writer . . . 35
8.5.3 Enumeration Type Documentation . . . 35
8.5.3.1 impact_PresetChange_Kind . . . 35
8.5.4 Function Documentation . . . 36
8.5.4.1 impact_PresetManager_InformChange() . . . 36
8.5.4.2 impact_SetPresetManager() . . . 36

8.6 Metadata Management . . . 37
8.6.1 Detailed Description . . . 37
8.6.2 Function Documentation . . . 37
8.6.2.1 impact_FreeMetadataInfos() . . . 37
8.6.2.2 impact_GetMetadataInfos() . . . 38
8.6.2.3 impact_SetMetadata() . . . 38
8.6.2.4 impact_SetMetadataMulti() . . . 39
8.7 Update Management . . . 39
8.7.1 Detailed Description . . . 39
8.7.2 Function Documentation . . . 39
8.7.2.1 impact_StartUpdateThread() . . . 39
8.7.2.2 impact_StopUpdateThread() . . . 40
8.7.2.3 impact_WaitUpdateThreadReady() . . . 40
8.8 Direct processing . . . 40
8.8.1 Detailed Description . . . 41
8.8.2 Function Documentation . . . 41
8.8.2.1 impact_GetBufferIn() . . . 41
8.8.2.2 impact_GetBufferOut() . . . 41
8.8.2.3 impact_GetEstimatedDelay() . . . 42
8.8.2.4 impact_ProcessAudio() . . . 42
8.8.2.5 impact_ProcessAudio_Planar() . . . 43
8.9 Audio managing (any audio) . . . 43
8.9.1 Detailed Description . . . 44
8.9.2 Enumeration Type Documentation . . . 44
8.9.2.1 impact_SampleFormat . . . 44
8.9.3 Function Documentation . . . 45
8.9.3.1 impact_AddAudio() . . . 45
8.9.3.2 impact_AddPadAudio() . . . 45
8.9.3.3 impact_GetAudio() . . . 45
8.9.3.4 impact_GetBytesFromFormat() . . . 46
8.9.3.5 impact_GetFormatFromName() . . . 46
8.9.3.6 impact_GetFormatName() . . . 47
8.9.3.7 impact_GetMaxPacketFrame() . . . 47
8.9.3.8 impact_GetOutputCount() . . . 47
8.10 Audio convert helpers . . . 48
8.10.1 Detailed Description . . . 48
8.10.2 Function Documentation . . . 48
8.10.2.1 impact_AudioConvertFrom() . . . 48
8.10.2.2 impact_AudioConvertTo() . . . 49
8.10.2.3 impact_AudioMonoFromLiveStereo() . . . 49
8.10.2.4 impact_AudioMonoToLiveStereo() . . . 49
8.10.2.5 impact_MonoToStereo() . . . 50

8.10.2.6 impact_MonoToStereo_Planar() . . . 50
8.10.2.7 impact_StereoToMono() . . . 51
8.10.2.8 impact_StereoToMono_Planar() . . . 51
8.11 Client's JSon config interfacing . . . 52
8.11.1 Detailed Description . . . 52
8.11.2 Function Documentation . . . 52
8.11.2.1 impact_DeleteClient() . . . 52
8.11.2.2 impact_FreeJsonAnswer() . . . 52
8.11.2.3 impact_NewClient() . . . 53
8.11.2.4 impact_ProcessJson() . . . 53
8.11.2.5 impact_SaveState() . . . 54
8.12 JSON syntax . . . 54
8.12.1 Commands . . . 54
8.12.2 Answers . . . 55
8.12.3 Examples . . . 55
8.12.3.1 Get status . . . 55
8.12.3.2 Get all changes since last request . . . 56
8.12.3.3 Check license . . . 56
8.12.3.4 Set Processor name . . . 56
8.12.3.5 Get presetlist . . . 56
8.12.3.6 Load a preset . . . 56
8.12.3.7 Get information on preset on air . . . 56
8.12.3.8 Set input gain . . . 56
8.12.3.9 Set bypass . . . 57
8.12.3.10 Export a preset . . . 57
8.12.3.11 Import a preset . . . 57
8.13 JSON syntax for internal setup . . . 57
8.13.1 Get process and modules information . . . 57
8.13.2 Authentication management . . . 58
8.13.2.1 Difference between setup and admin account . . . 58
8.13.2.2 Access encoding . . . 58
8.13.2.3 Change setup account . . . 58
8.13.2.4 Get user info . . . 59
8.13.2.5 Add or modify a user . . . 59
8.13.2.6 Removes a user . . . 59
8.13.2.7 List users . . . 59
8.13.2.8 Get/set unlogged access . . . 59
8.13.2.9 Get realm . . . 59
8.13.2.10 Get API version . . . 59
8.13.2.11 Get setup is admin . . . 60
8.13.2.12 Get/set setup account . . . 60
8.13.2.13 Get/set custom storage . . . 60

8.13.3 Get process' interface description . . . 60
8.13.3.1 Get XML decription . . . 60
8.13.3.2 Get Process Image . . . 60
8.13.3.3 Get Process Ico . . . 60
8.14 Web server . . . 61
8.14.1 Detailed Description . . . 61
8.14.2 Macro Definition Documentation . . . 61
8.14.2.1 SOUND4_INVALID_WEBSERVER_ID . . . 61
8.14.2.2 SOUND4_WEBSERVER_HTTP_OK . . . 61
8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK . . . 62
8.14.3 Function Documentation . . . 62
8.14.3.1 impact_Webserver() . . . 62
8.14.3.2 impact_Webserver_Status() . . . 62
8.14.3.3 impact_Webserver_Stop() . . . 63
8.14.3.4 impact_Webserver_tcp() . . . 63
8.14.3.5 impact_Webserver_tcp2() . . . 64
8.15 Sample using C++ interface wrapper . . . 65
8.15.1 Setup and sanity checks . . . 65
8.15.2 Sample to create the processing instance . . . 65
8.15.3 Communicating with the instance . . . 65
8.15.4 Starting a web-server . . . 65
8.15.5 Sample processing loop . . . 65
8.16 Sample using C++ interface wrapper with dynamic library loading . . . 65
8.16.1 Dynamic loading . . . 66
8.16.2 Setup and sanity checks . . . 66
8.16.3 Sample to create the processing instance . . . 66
8.16.4 Communicating with the instance . . . 66
8.16.5 Starting a web-server . . . 66
8.16.6 Sample processing loop . . . 66
8.16.7 Custom preset manager . . . 67
8.16.8 Custom preset manager loading . . . 67
8.17 Sample using Generic C++ interface (multi-process compatible) . . . 68
8.17.1 Loading the library . . . 68
8.17.2 Setup and configure . . . 68
8.17.3 Create the processing instance . . . 68
8.17.4 Processing loop . . . 68
8.17.5 Starting a web-server . . . 69
8.17.6 Communicating with the instance . . . 69
8.17.7 Custom preset manager example . . . 69
8.17.8 Custom preset manager loading . . . 70

**9 Namespace Documentation** **71**

9.1 dyn Namespace Reference . . . 71
9.1.1 Detailed Description . . . 71
9.2 sound4 Namespace Reference . . . 71
9.2.1 Detailed Description . . . 72
9.2.2 Typedef Documentation . . . 72
9.2.2.1 log_cb_t . . . 72
9.2.2.2 LogSeverity . . . 72
9.2.2.3 PresetChange_Kind . . . 72
9.2.3 Variable Documentation . . . 73
9.2.3.1 error . . . 73
9.2.3.2 fatal . . . 73
9.2.3.3 info . . . 73
9.2.3.4 none . . . 73
9.2.3.5 verbose . . . 73
9.2.3.6 verbose2 . . . 74
9.2.3.7 verbose3 . . . 74
9.2.3.8 verbose4 . . . 74
9.2.3.9 verbose5 . . . 74
9.2.3.10 warning . . . 74
9.3 sound4::helper Namespace Reference . . . 74
9.3.1 Detailed Description . . . 75
9.3.2 Typedef Documentation . . . 75
9.3.2.1 DynFunc_t . . . 75
9.3.2.2 DynLib_t . . . 75
9.4 sound4::impact Namespace Reference . . . 75
9.4.1 Detailed Description . . . 76
9.4.2 Typedef Documentation . . . 76
9.4.2.1 log_cb_t . . . 76
9.4.2.2 LogSeverity . . . 76
9.4.2.3 PresetChange_Kind . . . 76
9.4.2.4 SampleFormat . . . 77
9.4.3 Variable Documentation . . . 77
9.4.3.1 ChunkFrames . . . 77
9.4.3.2 InputChannels . . . 77
9.4.3.3 InputSampleSize . . . 77
9.4.3.4 OutputChannels . . . 77
9.4.3.5 OutputSampleSize . . . 78
9.4.3.6 SampleRate . . . 78
9.5 sound4::impact::dyn Namespace Reference . . . 78
9.5.1 Typedef Documentation . . . 79
9.5.1.1 log_cb_t . . . 79
9.5.1.2 LogSeverity . . . 79

9.5.1.3 PresetChange_Kind . . . 79
9.5.1.4 SampleFormat . . . 79
9.5.2 Variable Documentation . . . 79
9.5.2.1 ChunkFrames . . . 79
9.5.2.2 InputChannels . . . 80
9.5.2.3 InputSampleSize . . . 80
9.5.2.4 OutputChannels . . . 80
9.5.2.5 OutputSampleSize . . . 80
9.5.2.6 SampleRate . . . 80
9.6 sound4::impact::dyn::helper Namespace Reference . . . 81
9.6.1 Detailed Description . . . 81
9.6.2 Typedef Documentation . . . 81
9.6.2.1 DynFunc_t . . . 81
9.6.2.2 DynLib_t . . . 81
9.7 sound4::impact::helper Namespace Reference . . . 82
9.7.1 Detailed Description . . . 82
9.8 std Namespace Reference . . . 82

**10 Class Documentation** **83**
10.1 sound4::CBus Class Reference . . . 83
10.1.1 Detailed Description . . . 83
10.1.2 Constructor & Destructor Documentation . . . 83
10.1.2.1 CBus() . . . 83
10.1.2.2 ~CBus() . . . 84
10.1.3 Member Function Documentation . . . 84
10.1.3.1 Get() . . . 84
10.2 sound4::CInstance::CClient Class Reference . . . 84
10.2.1 Detailed Description . . . 84
10.2.2 Constructor & Destructor Documentation . . . 85
10.2.2.1 CClient() [1/2] . . . 85
10.2.2.2 CClient() [2/2] . . . 85
10.2.2.3 ~CClient() . . . 85
10.2.3 Member Function Documentation . . . 85
10.2.3.1 operator=() . . . 85
10.2.3.2 ProcessJson() . . . 85
10.3 sound4::impact::CInstance::CClient Class Reference . . . 86
10.3.1 Detailed Description . . . 86
10.3.2 Constructor & Destructor Documentation . . . 87
10.3.2.1 CClient() [1/2] . . . 87
10.3.2.2 CClient() [2/2] . . . 87
10.3.2.3 ~CClient() . . . 87
10.3.3 Member Function Documentation . . . 87

10.3.3.1 operator=() . . . 87
10.3.3.2 ProcessJson() . . . 87
10.4 sound4::impact::dyn::CInstance::CClient Class Reference . . . 88
10.4.1 Detailed Description . . . 88
10.4.2 Constructor & Destructor Documentation . . . 88
10.4.2.1 CClient() [1/2] . . . 89
10.4.2.2 CClient() [2/2] . . . 89
10.4.2.3 ~CClient() . . . 89
10.4.3 Member Function Documentation . . . 89
10.4.3.1 operator=() . . . 89
10.4.3.2 ProcessJson() . . . 89
10.5 sound4::impact::dyn::CDynLib Class Reference . . . 90
10.5.1 Detailed Description . . . 91
10.5.2 Constructor & Destructor Documentation . . . 92
10.5.2.1 CDynLib() . . . 92
10.5.3 Member Function Documentation . . . 92
10.5.3.1 IsOk() . . . 92
10.5.3.2 Load() . . . 92
10.5.4 Member Data Documentation . . . 92
10.5.4.1 AddAudio . . . 93
10.5.4.2 AddPadAudio . . . 93
10.5.4.3 AudioConvertFrom . . . 93
10.5.4.4 AudioConvertTo . . . 93
10.5.4.5 AudioMonoFromLiveStereo . . . 93
10.5.4.6 AudioMonoToLiveStereo . . . 93
10.5.4.7 DeleteClient . . . 94
10.5.4.8 ExitProcess . . . 94
10.5.4.9 FreeJsonAnswer . . . 94
10.5.4.10 FreeMetadataInfos . . . 94
10.5.4.11 FreeParameters . . . 94
10.5.4.12 FreeParameterValue . . . 94
10.5.4.13 GetAudio . . . 95
10.5.4.14 GetAudioInputCount . . . 95
10.5.4.15 GetAudioOutputCount . . . 95
10.5.4.16 GetBufferIn . . . 95
10.5.4.17 GetBufferOut . . . 95
10.5.4.18 GetBytesFromFormat . . . 95
10.5.4.19 GetChannelCount . . . 96
10.5.4.20 GetChunkSizeInFrames . . . 96
10.5.4.21 GetEstimatedDelay . . . 96
10.5.4.22 GetFormatFromName . . . 96
10.5.4.23 GetFormatName . . . 96

10.5.4.24 GetMaxPacketFrame . . . 96
10.5.4.25 GetMetadataInfos . . . 97
10.5.4.26 GetOutputCount . . . 97
10.5.4.27 GetParameter . . . 97
10.5.4.28 GetPossibleChunkSizeInFrames . . . 97
10.5.4.29 GetProcessChunkFrames . . . 97
10.5.4.30 GetSampleRate . . . 97
10.5.4.31 GetVersion . . . 98
10.5.4.32 InitProcess . . . 98
10.5.4.33 InitProcess2 . . . 98
10.5.4.34 InitProcess3 . . . 98
10.5.4.35 MonoToStereo . . . 98
10.5.4.36 MonoToStereo_Planar . . . 98
10.5.4.37 NewClient . . . 99
10.5.4.38 NewParameters . . . 99
10.5.4.39 PresetManager_InformChange . . . 99
10.5.4.40 ProcessAudio . . . 99
10.5.4.41 ProcessAudio_Planar . . . 99
10.5.4.42 ProcessJson . . . 99
10.5.4.43 SaveState . . . 100
10.5.4.44 SetLoggerCallback . . . 100
10.5.4.45 SetLogSeverity . . . 100
10.5.4.46 SetMetadata . . . 100
10.5.4.47 SetMetadataMulti . . . 100
10.5.4.48 SetParameter . . . 100
10.5.4.49 SetPresetManager . . . 101
10.5.4.50 StartUpdateThread . . . 101
10.5.4.51 StereoToMono . . . 101
10.5.4.52 StereoToMono_Planar . . . 101
10.5.4.53 StopUpdateThread . . . 101
10.5.4.54 TerminateProcess . . . 101
10.5.4.55 WaitUpdateThreadReady . . . 102
10.5.4.56 Webserver . . . 102
10.5.4.57 Webserver_Status . . . 102
10.5.4.58 Webserver_Stop . . . 102
10.5.4.59 Webserver_tcp . . . 102
10.5.4.60 Webserver_tcp2 . . . 103
10.6 sound4::helper::CDynLoader Class Reference . . . 103
10.6.1 Detailed Description . . . 103
10.6.2 Constructor & Destructor Documentation . . . 103
10.6.2.1 CDynLoader() [1/3] . . . 104
10.6.2.2 CDynLoader() [2/3] . . . 104

10.6.2.3 CDynLoader() [3/3] . . . 104
10.6.2.4 ~CDynLoader() . . . 104
10.6.3 Member Function Documentation . . . 104
10.6.3.1 Close() . . . 104
10.6.3.2 GetSymbol() . . . 104
10.6.3.3 GetSymbol_safe() . . . 105
10.6.3.4 GetThisLibraryPath() . . . 105
10.6.3.5 IsOk() . . . 105
10.6.3.6 Load() . . . 105
10.6.3.7 operator bool() . . . 105
10.6.3.8 operator=() . . . 106
10.7 sound4::impact::dyn::helper::CDynLoader Class Reference . . . 106
10.7.1 Detailed Description . . . 106
10.7.2 Constructor & Destructor Documentation . . . 106
10.7.2.1 CDynLoader() [1/3] . . . 107
10.7.2.2 CDynLoader() [2/3] . . . 107
10.7.2.3 CDynLoader() [3/3] . . . 107
10.7.2.4 ~CDynLoader() . . . 107
10.7.3 Member Function Documentation . . . 107
10.7.3.1 Close() . . . 107
10.7.3.2 GetSymbol() . . . 107
10.7.3.3 GetSymbol_safe() . . . 108
10.7.3.4 GetThisLibraryPath() . . . 108
10.7.3.5 IsOk() . . . 108
10.7.3.6 Load() . . . 108
10.7.3.7 operator bool() . . . 108
10.7.3.8 operator=() . . . 109
10.8 sound4::CInstance Class Reference . . . 109
10.8.1 Detailed Description . . . 110
10.8.2 Constructor & Destructor Documentation . . . 110
10.8.2.1 CInstance() [1/2] . . . 110
10.8.2.2 CInstance() [2/2] . . . 110
10.8.2.3 ~CInstance() . . . 110
10.8.3 Member Function Documentation . . . 110
10.8.3.1 AddAudio() . . . 111
10.8.3.2 Create() . . . 111
10.8.3.3 GetAudio() . . . 111
10.8.3.4 GetBufferIn() . . . 112
10.8.3.5 GetBufferOut() . . . 112
10.8.3.6 GetChunkFrames() . . . 112
10.8.3.7 GetEstimatedDelay() . . . 113
10.8.3.8 GetMetadataInfos() . . . 113

10.8.3.9 GetParam() . . . 113
10.8.3.10 IsOk() . . . 113
10.8.3.11 NewClient() . . . 114
10.8.3.12 OnUpdateThreadStart() . . . 114
10.8.3.13 OnUpdateThreadStop() . . . 114
10.8.3.14 operator=() . . . 114
10.8.3.15 PresetManager_InformChange() . . . 114
10.8.3.16 ProcessAnyAudio() . . . 115
10.8.3.17 ProcessAudio() . . . 115
10.8.3.18 ProcessAudio_Planar() . . . 116
10.8.3.19 SetBus() . . . 116
10.8.3.20 SetMetadata() . . . 116
10.8.3.21 SetMetadataMulti() . . . 117
10.8.3.22 SetParam() . . . 117
10.8.3.23 SetPresetManager() . . . 118
10.8.3.24 StartWebServer() . . . 118
10.8.3.25 Stop() . . . 118
10.8.3.26 StopWebServer() . . . 119
10.8.3.27 Terminate() . . . 119
10.9 sound4::impact::CInstance Class Reference . . . 119
10.9.1 Detailed Description . . . 120
10.9.2 Constructor & Destructor Documentation . . . 120
10.9.2.1 CInstance() [1/2] . . . 121
10.9.2.2 CInstance() [2/2] . . . 121
10.9.2.3 ~CInstance() . . . 121
10.9.3 Member Function Documentation . . . 121
10.9.3.1 AddAudio() . . . 121
10.9.3.2 Create() . . . 121
10.9.3.3 GetAudio() . . . 122
10.9.3.4 GetBufferIn() . . . 122
10.9.3.5 GetBufferOut() . . . 123
10.9.3.6 GetChunkFrames() . . . 123
10.9.3.7 GetEstimatedDelay() . . . 123
10.9.3.8 GetMetadataInfos() . . . 124
10.9.3.9 GetParam() . . . 124
10.9.3.10 IsOk() . . . 124
10.9.3.11 NewClient() . . . 124
10.9.3.12 OnUpdateThreadStart() . . . 125
10.9.3.13 OnUpdateThreadStop() . . . 125
10.9.3.14 operator=() . . . 125
10.9.3.15 PresetManager_InformChange() . . . 125
10.9.3.16 ProcessAnyAudio() . . . 125

10.9.3.17 ProcessAudio() . . . 126
10.9.3.18 ProcessAudio_Planar() . . . 126
10.9.3.19 SetMetadata() . . . 127
10.9.3.20 SetMetadataMulti() . . . 127
10.9.3.21 SetParam() . . . 128
10.9.3.22 SetPresetManager() . . . 128
10.9.3.23 StartWebServer() . . . 128
10.9.3.24 Stop() . . . 129
10.9.3.25 StopWebServer() . . . 129
10.9.3.26 Terminate() . . . 129
10.10 sound4::impact::dyn::CInstance Class Reference . . . 129
10.10.1 Detailed Description . . . 130
10.10.2 Constructor & Destructor Documentation . . . 131
10.10.2.1 CInstance() [1/2] . . . 131
10.10.2.2 CInstance() [2/2] . . . 131
10.10.2.3 ∼CInstance() . . . 131
10.10.3 Member Function Documentation . . . 131
10.10.3.1 AddAudio() . . . 131
10.10.3.2 Create() . . . 131
10.10.3.3 GetAudio() . . . 132
10.10.3.4 GetBufferIn() . . . 132
10.10.3.5 GetBufferOut() . . . 133
10.10.3.6 GetChunkFrames() . . . 133
10.10.3.7 GetEstimatedDelay() . . . 133
10.10.3.8 GetMetadataInfos() . . . 134
10.10.3.9 GetParam() . . . 134
10.10.3.10 IsOk() . . . 134
10.10.3.11 NewClient() . . . 134
10.10.3.12 OnUpdateThreadStart() . . . 135
10.10.3.13 OnUpdateThreadStop() . . . 135
10.10.3.14 operator=() . . . 135
10.10.3.15 PresetManager_InformChange() . . . 135
10.10.3.16 ProcessAnyAudio() . . . 135
10.10.3.17 ProcessAudio() . . . 136
10.10.3.18 ProcessAudio_Planar() . . . 136
10.10.3.19 SetMetadata() . . . 137
10.10.3.20 SetMetadataMulti() . . . 137
10.10.3.21 SetParam() . . . 138
10.10.3.22 SetPresetManager() . . . 138
10.10.3.23 StartWebServer() . . . 138
10.10.3.24 Stop() . . . 139
10.10.3.25 StopWebServer() . . . 139

10.10.3.26 Terminate() . . . 139
10.11 sound4::CPresetLoader Class Reference . . . 140
10.11.1 Detailed Description . . . 140
10.11.2 Constructor & Destructor Documentation . . . 140
10.11.2.1 CPresetLoader() . . . 140
10.11.2.2 ∼CPresetLoader() . . . 140
10.11.3 Member Function Documentation . . . 140
10.11.3.1 Exists() . . . 141
10.11.3.2 GetAll() . . . 141
10.11.3.3 IsReadOnly() . . . 141
10.11.3.4 Read() . . . 141
10.11.3.5 Remove() . . . 141
10.11.3.6 Rename() . . . 141
10.11.3.7 Write() . . . 142
10.12 sound4::impact::CPresetLoader Class Reference . . . 142
10.12.1 Detailed Description . . . 142
10.12.2 Constructor & Destructor Documentation . . . 142
10.12.2.1 CPresetLoader() . . . 142
10.12.2.2 ∼CPresetLoader() . . . 143
10.12.3 Member Function Documentation . . . 143
10.12.3.1 Exists() . . . 143
10.12.3.2 GetAll() . . . 143
10.12.3.3 IsReadOnly() . . . 143
10.12.3.4 Read() . . . 143
10.12.3.5 Remove() . . . 143
10.12.3.6 Rename() . . . 144
10.12.3.7 Write() . . . 144
10.13 sound4::impact::dyn::CPresetLoader Class Reference . . . 144
10.13.1 Detailed Description . . . 144
10.13.2 Constructor & Destructor Documentation . . . 144
10.13.2.1 CPresetLoader() . . . 145
10.13.2.2 ∼CPresetLoader() . . . 145
10.13.3 Member Function Documentation . . . 145
10.13.3.1 Exists() . . . 145
10.13.3.2 GetAll() . . . 145
10.13.3.3 IsReadOnly() . . . 145
10.13.3.4 Read() . . . 145
10.13.3.5 Remove() . . . 146
10.13.3.6 Rename() . . . 146
10.13.3.7 Write() . . . 146
10.14 sound4::CProcessor Class Reference . . . 146
10.14.1 Detailed Description . . . 149

10.14.2 Constructor & Destructor Documentation . . . 149
10.14.2.1 CProcessor() . . . 149
10.14.3 Member Function Documentation . . . 149
10.14.3.1 AudioConvertFrom() . . . 149
10.14.3.2 AudioConvertTo() . . . 149
10.14.3.3 AudioMonoFromLiveStereo() . . . 150
10.14.3.4 AudioMonoToLiveStereo() . . . 150
10.14.3.5 GetAudioInputCount() . . . 150
10.14.3.6 GetAudioOutputCount() . . . 150
10.14.3.7 GetBytesFromFormat() . . . 150
10.14.3.8 GetChannelCount() . . . 151
10.14.3.9 GetChunkSizeInFrames() . . . 151
10.14.3.10 GetFormatFromName() . . . 151
10.14.3.11 GetFormatName() . . . 152
10.14.3.12 GetPossibleChunkSizeInFrames() . . . 152
10.14.3.13 GetPrefixSymbol() . . . 153
10.14.3.14 GetSampleRate() . . . 153
10.14.3.15 GetVersion() . . . 153
10.14.3.16 IsOk() . . . 153
10.14.3.17 Load() . . . 153
10.14.3.18 MonoToStereo() . . . 154
10.14.3.19 MonoToStereo_Planar() . . . 154
10.14.3.20 SetLoggerCallback() . . . 154
10.14.3.21 SetLogSeverity() . . . 155
10.14.3.22 StereoToMono() . . . 155
10.14.3.23 StereoToMono_Planar() . . . 155
10.14.4 Friends And Related Function Documentation . . . 155
10.14.4.1 CBus . . . 155
10.14.4.2 CInstance . . . 155
10.14.5 Member Data Documentation . . . 156
10.14.5.1 fnAddAudio . . . 156
10.14.5.2 fnAddPadAudio . . . 156
10.14.5.3 fnAudioConvertFrom . . . 156
10.14.5.4 fnAudioConvertTo . . . 156
10.14.5.5 fnAudioMonoFromLiveStereo . . . 156
10.14.5.6 fnAudioMonoToLiveStereo . . . 157
10.14.5.7 fnDeleteClient . . . 157
10.14.5.8 fnExitProcess . . . 157
10.14.5.9 fnFreeBus . . . 157
10.14.5.10 fnFreeJsonAnswer . . . 157
10.14.5.11 fnFreeMetadataInfos . . . 157
10.14.5.12 fnFreeParameters . . . 158

10.14.5.13 fnFreeParameterValue . . . 158
10.14.5.14 fnGetAudio . . . 158
10.14.5.15 fnGetAudioInputCount . . . 158
10.14.5.16 fnGetAudioOutputCount . . . 158
10.14.5.17 fnGetBufferIn . . . 158
10.14.5.18 fnGetBufferOut . . . 159
10.14.5.19 fnGetBytesFromFormat . . . 159
10.14.5.20 fnGetChannelCount . . . 159
10.14.5.21 fnGetChunkSizeInFrames . . . 159
10.14.5.22 fnGetEstimatedDelay . . . 159
10.14.5.23 fnGetFormatFromName . . . 159
10.14.5.24 fnGetFormatName . . . 160
10.14.5.25 fnGetMaxPacketFrame . . . 160
10.14.5.26 fnGetMetadataInfos . . . 160
10.14.5.27 fnGetOutputCount . . . 160
10.14.5.28 fnGetParameter . . . 160
10.14.5.29 fnGetPossibleChunkSizeInFrames . . . 160
10.14.5.30 fnGetProcessChunkFrames . . . 161
10.14.5.31 fnGetSampleRate . . . 161
10.14.5.32 fnGetVersion . . . 161
10.14.5.33 fnInitProcess . . . 161
10.14.5.34 fnInitProcess2 . . . 161
10.14.5.35 fnInitProcess3 . . . 161
10.14.5.36 fnMonoToStereo . . . 162
10.14.5.37 fnMonoToStereo_Planar . . . 162
10.14.5.38 fnNewBus . . . 162
10.14.5.39 fnNewClient . . . 162
10.14.5.40 fnNewParameters . . . 162
10.14.5.41 fnPresetManager_InformChange . . . 162
10.14.5.42 fnProcessAudio . . . 163
10.14.5.43 fnProcessAudio_Planar . . . 163
10.14.5.44 fnProcessJson . . . 163
10.14.5.45 fnSaveState . . . 163
10.14.5.46 fnSetInstanceBus . . . 163
10.14.5.47 fnSetLoggerCallback . . . 163
10.14.5.48 fnSetLogSeverity . . . 164
10.14.5.49 fnSetMetadata . . . 164
10.14.5.50 fnSetMetadataMulti . . . 164
10.14.5.51 fnSetParameter . . . 164
10.14.5.52 fnSetPresetManager . . . 164
10.14.5.53 fnStartUpdateThread . . . 164
10.14.5.54 fnStereoToMono . . . 165

10.14.5.55 fnStereoToMono_Planar . . . 165
10.14.5.56 fnStopUpdateThread . . . 165
10.14.5.57 fnTerminateProcess . . . 165
10.14.5.58 fnWaitUpdateThreadReady . . . 165
10.14.5.59 fnWebserver . . . 165
10.14.5.60 fnWebserver_Status . . . 166
10.14.5.61 fnWebserver_Stop . . . 166
10.14.5.62 fnWebserver_tcp . . . 166
10.14.5.63 fnWebserver_tcp2 . . . 166
10.15 sound4::helper::DynFuncHolder< T > Class Template Reference . . . 166
10.15.1 Detailed Description . . . 166
10.15.2 Constructor & Destructor Documentation . . . 167
10.15.2.1 DynFuncHolder() [1/2] . . . 167
10.15.2.2 DynFuncHolder() [2/2] . . . 167
10.15.3 Member Function Documentation . . . 167
10.15.3.1 IsOk() . . . 167
10.15.3.2 operator bool() . . . 167
10.15.3.3 operator T∗() . . . 168
10.16 sound4::impact::dyn::helper::DynFuncHolder< T > Class Template Reference . . . 168
10.16.1 Detailed Description . . . 168
10.16.2 Constructor & Destructor Documentation . . . 168
10.16.2.1 DynFuncHolder() [1/2] . . . 168
10.16.2.2 DynFuncHolder() [2/2] . . . 169
10.16.3 Member Function Documentation . . . 169
10.16.3.1 IsOk() . . . 169
10.16.3.2 operator bool() . . . 169
10.16.3.3 operator T∗() . . . 169
10.17 impact_CClientInstance Struct Reference . . . 169
10.17.1 Detailed Description . . . 170
10.18 impact_CInstance Struct Reference . . . 170
10.18.1 Detailed Description . . . 170
10.19 impact_CParameters Struct Reference . . . 170
10.19.1 Detailed Description . . . 170
10.20 MyPresetManager Class Reference . . . 170
10.20.1 Detailed Description . . . 171
10.20.2 Constructor & Destructor Documentation . . . 171
10.20.2.1 MyPresetManager() [1/2] . . . 171
10.20.2.2 ∼MyPresetManager() [1/2] . . . 171
10.20.2.3 MyPresetManager() [2/2] . . . 171
10.20.2.4 ∼MyPresetManager() [2/2] . . . 172
10.20.3 Member Function Documentation . . . 172
10.20.3.1 Exists() [1/2] . . . 172

10.20.3.2 Exists() [2/2] . . . 172
10.20.3.3 GetAll() [1/2] . . . 172
10.20.3.4 GetAll() [2/2] . . . 172
10.20.3.5 IsReadOnly() [1/2] . . . 173
10.20.3.6 IsReadOnly() [2/2] . . . 173
10.20.3.7 Read() [1/2] . . . 173
10.20.3.8 Read() [2/2] . . . 173
10.20.3.9 Remove() [1/2] . . . 173
10.20.3.10 Remove() [2/2] . . . 174
10.20.3.11 Rename() [1/2] . . . 174
10.20.3.12 Rename() [2/2] . . . 174
10.20.3.13 SetupPath() [1/2] . . . 174
10.20.3.14 SetupPath() [2/2] . . . 174
10.20.3.15 Write() [1/2] . . . 175
10.20.3.16 Write() [2/2] . . . 175
10.21 sound4::helper::SampleFormat< T > Struct Template Reference . . . 175
10.21.1 Detailed Description . . . 175
10.22 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference . . . 176
10.22.1 Detailed Description . . . 176
10.23 sound4::impact::helper::SampleFormat< T > Struct Template Reference . . . 176
10.23.1 Detailed Description . . . 176
10.24 sound4::helper::SampleFormat< float > Struct Reference . . . 177
10.24.1 Detailed Description . . . 177
10.24.2 Member Data Documentation . . . 177
10.24.2.1 format . . . 177
10.25 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference . . . 177
10.25.1 Detailed Description . . . 177
10.25.2 Member Data Documentation . . . 177
10.25.2.1 format . . . 178
10.26 sound4::impact::helper::SampleFormat< float > Struct Reference . . . 178
10.26.1 Detailed Description . . . 178
10.26.2 Member Data Documentation . . . 178
10.26.2.1 format . . . 178
10.27 sound4::helper::SampleFormat< int16_t > Struct Reference . . . 178
10.27.1 Detailed Description . . . 179
10.27.2 Member Data Documentation . . . 179
10.27.2.1 format . . . 179
10.28 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference . . . 179
10.28.1 Detailed Description . . . 179
10.28.2 Member Data Documentation . . . 179
10.28.2.1 format . . . 179
10.29 sound4::impact::helper::SampleFormat< int16_t > Struct Reference . . . 180

10.29.1 Detailed Description . . . 180
10.29.2 Member Data Documentation . . . 180
10.29.2.1 format . . . 180
10.30 sound4::helper::SampleFormat< int32_t > Struct Reference . . . 180
10.30.1 Detailed Description . . . 180
10.30.2 Member Data Documentation . . . 180
10.30.2.1 format . . . 181
10.31 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference . . . 181
10.31.1 Detailed Description . . . 181
10.31.2 Member Data Documentation . . . 181
10.31.2.1 format . . . 181
10.32 sound4::impact::helper::SampleFormat< int32_t > Struct Reference . . . 181
10.32.1 Detailed Description . . . 182
10.32.2 Member Data Documentation . . . 182
10.32.2.1 format . . . 182
10.33 SOUND4_ProcessInfo Struct Reference . . . 182
10.33.1 Detailed Description . . . 182
10.33.2 Member Data Documentation . . . 183
10.33.2.1 name . . . 183
10.33.2.2 prefix . . . 183
10.33.2.3 shortname . . . 183
10.33.2.4 struct_size . . . 183
10.33.2.5 version . . . 183
10.34 sound4cl_SOUND4_ProcessInfo Struct Reference . . . 184
10.34.1 Detailed Description . . . 184
10.34.2 Member Data Documentation . . . 184
10.34.2.1 name . . . 184
10.34.2.2 prefix . . . 184
10.34.2.3 shortname . . . 184
10.34.2.4 struct_size . . . 185
10.34.2.5 version . . . 185

**11 File Documentation** **187**
11.1 sound4.impact.cl-pipe-cxx.cpp File Reference . . . 187
11.1.1 Function Documentation . . . 187
11.1.1.1 GetEnv() . . . 187
11.1.1.2 main() . . . 188
11.1.1.3 MyLogger() . . . 188
11.1.1.4 ProcessLoop() . . . 188
11.1.1.5 Usage() . . . 188
11.2 sound4.impact.cl-pipe-cxx.cpp . . . 189
11.3 sound4.impact.cl-pipe-dyn.cpp File Reference . . . 190

11.3.1 Function Documentation . . . 191
11.3.1.1 GetEnv() . . . 191
11.3.1.2 main() . . . 192
11.3.1.3 MyLogger() . . . 192
11.3.1.4 ProcessLoop() . . . 192
11.3.1.5 Usage() . . . 193
11.4 sound4.impact.cl-pipe-dyn.cpp . . . 193
11.5 sound4.impact.cl-pipe.cpp File Reference . . . 196
11.5.1 Macro Definition Documentation . . . 196
11.5.1.1 CHANNEL_COUNT . . . 197
11.5.1.2 CHUNK_SIZE . . . 197
11.5.1.3 SAMPLE_RATE . . . 197
11.5.2 Function Documentation . . . 197
11.5.2.1 main() . . . 197
11.5.2.2 MyLogger() . . . 198
11.5.2.3 ProcessLoop() . . . 198
11.5.2.4 Usage() . . . 198
11.6 sound4.impact.cl-pipe.cpp . . . 198
11.7 sound4.impact.cl.h File Reference . . . 201
11.7.1 Macro Definition Documentation . . . 204
11.7.1.1 IMPACT_DECL . . . 204
11.7.1.2 IMPACT_HAS_CLOUDBUS . . . 204
11.7.1.3 IMPACT_HAS_WEBSERVER . . . 204
11.8 sound4.impact.cl.h . . . 205
11.9 sound4.impact.cl.hpp File Reference . . . 208
11.9.1 Macro Definition Documentation . . . 210
11.9.1.1 fs_strdup . . . 210
11.10 sound4.impact.cl.hpp . . . 210
11.11 sound4.impact.cl_dyn.hpp File Reference . . . 217
11.11.1 Detailed Description . . . 218
11.11.2 Macro Definition Documentation . . . 219
11.11.2.1 fs_strdup . . . 219
11.12 sound4.impact.cl_dyn.hpp . . . 219
11.13 sound4cl-pipe.cpp File Reference . . . 231
11.13.1 Function Documentation . . . 232
11.13.1.1 GetEnv() . . . 232
11.13.1.2 main() . . . 232
11.13.1.3 ProcessLoop() . . . 233
11.14 sound4cl-pipe.cpp . . . 233
11.15 sound4cl.hpp File Reference . . . 236
11.15.1 Detailed Description . . . 238
11.16 sound4cl.hpp . . . 238

11.17 sound4cl_cdef.h File Reference . . . 250
11.17.1 Macro Definition Documentation . . . 252
11.17.1.1 fs_strdup . . . 252
11.17.1.2 SOUND4_INVALID_WEBSERVER_ID . . . 253
11.17.1.3 SOUND4_WEBSERVER_HTTP_OK . . . 253
11.17.1.4 SOUND4_WEBSERVER_HTTPS_OK . . . 253
11.17.2 Typedef Documentation . . . 253
11.17.2.1 fs_char . . . 253
11.17.2.2 sound4cl_loggerfn . . . 253
11.17.2.3 sound4cl_storage_exists . . . 253
11.17.2.4 sound4cl_storage_getall . . . 254
11.17.2.5 sound4cl_storage_getall_free . . . 254
11.17.2.6 sound4cl_storage_reader . . . 254
11.17.2.7 sound4cl_storage_reader_free . . . 254
11.17.2.8 sound4cl_storage_remove . . . 254
11.17.2.9 sound4cl_storage_rename . . . 254
11.17.2.10 sound4cl_storage_writer . . . 254
11.17.3 Enumeration Type Documentation . . . 254
11.17.3.1 sound4cl_LogSeverity . . . 254
11.17.3.2 sound4cl_PresetChange_Kind . . . 255
11.17.3.3 sound4cl_SampleFormat . . . 255
11.17.4 Function Documentation . . . 256
11.17.4.1 sound4cl_AddAudio() . . . 256
11.17.4.2 sound4cl_AddPadAudio() . . . 256
11.17.4.3 sound4cl_AudioConvertFrom() . . . 256
11.17.4.4 sound4cl_AudioConvertTo() . . . 256
11.17.4.5 sound4cl_AudioMonoFromLiveStereo() . . . 257
11.17.4.6 sound4cl_AudioMonoToLiveStereo() . . . 257
11.17.4.7 sound4cl_DeleteClient() . . . 257
11.17.4.8 sound4cl_ExitProcess() . . . 257
11.17.4.9 sound4cl_FreeBus() . . . 257
11.17.4.10 sound4cl_FreeJsonAnswer() . . . 257
11.17.4.11 sound4cl_FreeMetadataInfos() . . . 258
11.17.4.12 sound4cl_FreeParameters() . . . 258
11.17.4.13 sound4cl_FreeParameterValue() . . . 258
11.17.4.14 sound4cl_GetAudio() . . . 258
11.17.4.15 sound4cl_GetAudioInputCount() . . . 258
11.17.4.16 sound4cl_GetAudioOutputCount() . . . 258
11.17.4.17 sound4cl_GetBufferIn() . . . 258
11.17.4.18 sound4cl_GetBufferOut() . . . 259
11.17.4.19 sound4cl_GetBytesFromFormat() . . . 259
11.17.4.20 sound4cl_GetChannelCount() . . . 259

11.17.4.21 sound4cl_GetChunkSizeInFrames() . . . . . 259
11.17.4.22 sound4cl_GetEstimatedDelay() . . . . . 259
11.17.4.23 sound4cl_GetFormatFromName() . . . . . 259
11.17.4.24 sound4cl_GetFormatName() . . . . . 259
11.17.4.25 sound4cl_GetMaxPacketFrame() . . . . . 260
11.17.4.26 sound4cl_GetMetadataInfos() . . . . . 260
11.17.4.27 sound4cl_GetOutputCount() . . . . . 260
11.17.4.28 sound4cl_GetParameter() . . . . . 260
11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames() . . . . . 260
11.17.4.30 sound4cl_GetProcessChunkFrames() . . . . . 260
11.17.4.31 sound4cl_GetSampleRate() . . . . . 260
11.17.4.32 sound4cl_GetVersion() . . . . . 261
11.17.4.33 sound4cl_InitProcess() . . . . . 261
11.17.4.34 sound4cl_InitProcess2() . . . . . 261
11.17.4.35 sound4cl_InitProcess3() . . . . . 261
11.17.4.36 sound4cl_MonoToStereo() . . . . . 261
11.17.4.37 sound4cl_MonoToStereo_Planar() . . . . . 262
11.17.4.38 sound4cl_NewBus() . . . . . 262
11.17.4.39 sound4cl_NewClient() . . . . . 262
11.17.4.40 sound4cl_NewParameters() . . . . . 262
11.17.4.41 sound4cl_PresetManager_InformChange() . . . . . 262
11.17.4.42 sound4cl_ProcessAudio() . . . . . 262
11.17.4.43 sound4cl_ProcessAudio_Planar() . . . . . 263
11.17.4.44 sound4cl_ProcessJson() . . . . . 263
11.17.4.45 sound4cl_SaveState() . . . . . 263
11.17.4.46 sound4cl_SetInstanceBus() . . . . . 263
11.17.4.47 sound4cl_SetLoggerCallback() . . . . . 263
11.17.4.48 sound4cl_SetLogSeverity() . . . . . 263
11.17.4.49 sound4cl_SetMetadata() . . . . . 264
11.17.4.50 sound4cl_SetMetadataMulti() . . . . . 264
11.17.4.51 sound4cl_SetParameter() . . . . . 264
11.17.4.52 sound4cl_SetPresetManager() . . . . . 264
11.17.4.53 sound4cl_SOUND4_GetProcessInfo() . . . . . 264
11.17.4.54 sound4cl_StartUpdateThread() . . . . . 265
11.17.4.55 sound4cl_StereoToMono() . . . . . 265
11.17.4.56 sound4cl_StereoToMono_Planar() . . . . . 265
11.17.4.57 sound4cl_StopUpdateThread() . . . . . 265
11.17.4.58 sound4cl_TerminateProcess() . . . . . 265
11.17.4.59 sound4cl_WaitUpdateThreadReady() . . . . . 265
11.17.4.60 sound4cl_Webserver() . . . . . 266
11.17.4.61 sound4cl_Webserver_Status() . . . . . 266
11.17.4.62 sound4cl_Webserver_Stop() . . . . . 266

11.17.4.63 sound4cl_Webserver_tcp() . . . . . 266
11.17.4.64 sound4cl_Webserver_tcp2() . . . . . 266
11.18 sound4cl_cdef.h . . . . . 267

**Index** **271**

# Chapter 1

# SOUND4 IMPACT.CL Library

## 1.1 General usage

You may check library configuration and setup logging Global setup and Environment variables and parameters

You have to create/destroy an instance Process creation/destruction.

For audio, you can either use direct optimized form with Direct processing or you can use Audio managing (any audio) to have any format. You may use Audio convert helpers.

You need to start a thread calling impact_StartUpdateThread see Update Management

You can start a web server for configuration, see Web server

If it is needed, you can use metadata : see Metadata Management

If you need, you may use your own custom preset manager for saving files and presets. See Custom preset managagement

To access the presets/settings, you can either use JSON socket from Update Management or do all manually with Client's JSon config interfacing using JSON syntax

See full example cpp/sound4.impact.cl-pipe.cpp

You can also use the C++ wrapper from header sound4.impact.cl.hpp or the one with dynamic library loading sound4.impact.cl_dyn.hpp.

**Note**

In all library, all `char*` are **UTF-8** encoded

## 1.2 Examples

See Sample using C interface

See Sample using Generic C++ interface (multi-process compatible)

See Sample using C++ interface wrapper

See Sample using C++ interface wrapper with dynamic library loading

# Chapter 2

# Deprecated List

**Member impact_GetChunkSizeInFrames ()**

Use impact_GetPossibleChunkSizeInFrames

**Member impact_InitProcess (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path)**

Use impact_InitProcess3

**Member impact_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters)**

Use impact_InitProcess3

# Chapter 3

# Module Index

## 3.1 Modules

Here is a list of all modules:


Sample using C interface . . . . . 17
Environment variables and parameters . . . . . 19
Global setup . . . . . 20
Process creation/destruction . . . . . 26
Custom preset managagement . . . . . 31
Metadata Management . . . . . 37
Update Management . . . . . 39
Direct processing . . . . . 40
Audio managing (any audio) . . . . . 43
Audio convert helpers . . . . . 48
Client's JSon config interfacing . . . . . 52
JSON syntax . . . . . 54
JSON syntax for internal setup . . . . . 57
Web server . . . . . 61
Sample using C++ interface wrapper . . . . . 65
Sample using C++ interface wrapper with dynamic library loading . . . . . 65
Sample using Generic C++ interface (multi-process compatible) . . . . . 68

# Chapter 4

# Namespace Index

## 4.1 Namespace List

Here is a list of all namespaces with brief descriptions:


dyn . . . . . 71
sound4 . . . . . 71
sound4::helper . . . . . 74
sound4::impact . . . . . 75
sound4::impact::dyn . . . . . 78
sound4::impact::dyn::helper . . . . . 81
sound4::impact::helper . . . . . 82
std . . . . . 82

# Chapter 5

# Hierarchical Index

## 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


sound4::CBus . . . 83
sound4::CInstance::CClient . . . 84
sound4::impact::CInstance::CClient . . . 86
sound4::impact::dyn::CInstance::CClient . . . 88
sound4::impact::dyn::CDynLib . . . 90
sound4::helper::CDynLoader . . . 103
sound4::impact::dyn::helper::CDynLoader . . . 106
sound4::CInstance . . . 109
sound4::impact::CInstance . . . 119
sound4::impact::dyn::CInstance . . . 129
sound4::CPresetLoader . . . 140
    MyPresetManager . . . 170
sound4::impact::CPresetLoader . . . 142
sound4::impact::dyn::CPresetLoader . . . 144
    MyPresetManager . . . 170
sound4::CProcessor . . . 146
sound4::helper::DynFuncHolder< T > . . . 166
sound4::impact::dyn::helper::DynFuncHolder< T > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddAudio) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddPadAudio) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertFrom) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertTo) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_DeleteClient) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ExitProcess) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameters) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameterValue) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudio) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioInputCount) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferIn) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferOut) > . . . 168

sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChannelCount) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatFromName) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatName) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMetadataInfos) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetOutputCount) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetParameter) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames) > . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetSampleRate) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetVersion) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess2) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess3) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewClient) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewParameters) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_PresetManager_InformChange) > . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessJson) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SaveState) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLoggerCallback) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLogSeverity) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadata) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadataMulti) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetParameter) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetPresetManager) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StartUpdateThread) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StopUpdateThread) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_TerminateProcess) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Status) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Stop) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp) > . . . . . 168
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp2) > . . . . . 168
sound4::helper::DynFuncHolder< decltype(sound4cl_AddAudio) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeBus) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudio) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > . . . . . 166

sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetParameter) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_GetVersion) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_NewBus) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_NewClient) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_NewParameters) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessJson) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SaveState) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadata) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetParameter) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) > . . . . . 166
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) > . . . . . 166
impact_CClientInstance . . . . . 169
impact_CInstance . . . . . 170
impact_CParameters . . . . . 170
sound4::helper::SampleFormat< T > . . . . . 175
sound4::impact::dyn::helper::SampleFormat< T > . . . . . 176
sound4::impact::helper::SampleFormat< T > . . . . . 176
sound4::helper::SampleFormat< float > . . . . . 177
sound4::impact::dyn::helper::SampleFormat< float > . . . . . 177
sound4::impact::helper::SampleFormat< float > . . . . . 178
sound4::helper::SampleFormat< int16_t > . . . . . 178

sound4::impact::dyn::helper::SampleFormat< int16_t > . . . . . 179
sound4::impact::helper::SampleFormat< int16_t > . . . . . 180
sound4::helper::SampleFormat< int32_t > . . . . . 180
sound4::impact::dyn::helper::SampleFormat< int32_t > . . . . . 181
sound4::impact::helper::SampleFormat< int32_t > . . . . . 181
SOUND4_ProcessInfo . . . . . 182
sound4cl_SOUND4_ProcessInfo . . . . . 184

# Chapter 6

# Class Index

## 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


sound4::CBus
Shared Bus . . . 83
sound4::CInstance::CClient . . . 84
sound4::impact::CInstance::CClient . . . 86
sound4::impact::dyn::CInstance::CClient . . . 88
sound4::impact::dyn::CDynLib
Dynamic library interface . . . 90
sound4::helper::CDynLoader
Helper to load dynamic library . . . 103
sound4::impact::dyn::helper::CDynLoader
Helper to load dynamic library . . . 106
sound4::CInstance
Instance handling class . . . 109
sound4::impact::CInstance . . . 119
sound4::impact::dyn::CInstance . . . 129
sound4::CPresetLoader
Custom preset handler helper . . . 140
sound4::impact::CPresetLoader . . . 142
sound4::impact::dyn::CPresetLoader . . . 144
sound4::CProcessor
Dynamic library interface . . . 146
sound4::helper::DynFuncHolder< T >
Helper to wrap a function pointer . . . 166
sound4::impact::dyn::helper::DynFuncHolder< T >
Helper to wrap a function pointer . . . 168
impact_CClientInstance . . . 169
impact_CInstance . . . 170
impact_CParameters . . . 170
MyPresetManager
[Custom preset manager] . . . 170
sound4::helper::SampleFormat< T >
Helper for Sample format types . . . 175
sound4::impact::dyn::helper::SampleFormat< T >
Helper for Sample format types . . . 176
sound4::impact::helper::SampleFormat< T >
Helper for Sample format types . . . 176

sound4::helper::SampleFormat< float > . . . . . 177
sound4::impact::dyn::helper::SampleFormat< float > . . . . . 177
sound4::impact::helper::SampleFormat< float > . . . . . 178
sound4::helper::SampleFormat< int16_t > . . . . . 178
sound4::impact::dyn::helper::SampleFormat< int16_t > . . . . . 179
sound4::impact::helper::SampleFormat< int16_t > . . . . . 180
sound4::helper::SampleFormat< int32_t > . . . . . 180
sound4::impact::dyn::helper::SampleFormat< int32_t > . . . . . 181
sound4::impact::helper::SampleFormat< int32_t > . . . . . 181
SOUND4_ProcessInfo . . . . . 182
sound4cl_SOUND4_ProcessInfo . . . . . 184

# Chapter 7

# File Index

## 7.1 File List

Here is a list of all files with brief descriptions:


sound4.impact.cl-pipe-cxx.cpp
C++ sample using C++ interface wrapper . . . . . 187
sound4.impact.cl-pipe-dyn.cpp
C++ sample using C++ interface wrapper with dynamic library loading . . . . . 190
sound4.impact.cl-pipe.cpp
C++ sample using only C interface . . . . . 196
sound4.impact.cl.h
C interface for library . . . . . 201
sound4.impact.cl.hpp
C++ wrapper for library . . . . . 208
sound4.impact.cl_dyn.hpp
C++ wrapper for dynamic loading library . . . . . 217
sound4cl-pipe.cpp
C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided . . . . . 231
sound4cl.hpp
C++ wrapper for dynamic loading library . . . . . 236
sound4cl_cdef.h
C function signature definition for all sound4.cl libraries . . . . . 250

# Chapter 8

# Module Documentation

## 8.1 Sample using C interface

See full example cpp/sound4.impact.cl-pipe.cpp

### 8.1.1 Sample to create the processing instance

see Process creation/destruction and Update Management

```
    // Setup parameters
    struct impact_CParameters *params = impact_NewParameters();
    // Optional: activate web authentication, set admin password
    // This creates the authentication only for a new storage path
    impact_SetParameter(params,"ADMIN_USER", "admin");
    impact_SetParameter(params,"ADMIN_SECRET", "admin");

    // Create the processing instance
    pInstance=impact_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
    impact_FreeParameters(params);
    if (!pInstance) {
      std::cerr « "Failed to create the process instance"« std::endl;
      return 1;
    }
    // Start the update thread, without JSON server (low priority)
    std::thread updater([pInstance](){
       // will returned when stopped or destroyed
       std::cerr « "impact_StartUpdateThread (no json port)" « std::endl;
       impact_StartUpdateThread(pInstance, 0);
       std::cerr « "impact_StartUpdateThread END" « std::endl;
     });
    // This is not mandatory, but better be sure we are ready right now
    std::cerr « "Waiting update thread" « std::endl;
    if (impact_WaitUpdateThreadReady(pInstance, 5000)<0) {
        std::cerr « "Timeout to create the process instance"« std::endl;
    }
    std::cerr « "Waiting update thread DONE" « std::endl;
#if (IMPACT_HAS_WEBSERVER)
    // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
    std::cerr « "Starting web server" « std::endl;
    int listenport= 8080;
    int listenport_secure= 8443;
    uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
    if (listenport || listenport_secure) {
        webserver = impact_Webserver(listenport,listenport_secure,pInstance);
        if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
            std::cerr « "Starting web server FAILED" « std::endl;
        } else {
            int web_status=impact_Webserver_Status(webserver);
            if (listenport) {
                if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
                    std::cerr « "Web server listening on port " « listenport;
                } else {
                    std::cerr « "Web server failed to listen on port " « listenport;
                }
            }
```

```
            if (listenport_secure) {
                if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
                    std::cerr « "Web server listening on secure port " « listenport_secure;
                } else {
                    std::cerr « "Web server failed to listen on secure port " « listenport_secure;
                }
            }
            std::cerr « "Starting web server DONE" « std::endl;
        }
    }
#endif // (IMPACT_HAS_WEBSERVER)

     // Force load a factory preset now
     // Could be used instead of setting a store_path
     /*{
         impact_CClientInstance *pClient=impact_NewClient(pInstance);
         const char *answer=impact_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }", NULL);
         std::cerr « "Loaded factory preset answer: " « answer« std::endl;
         impact_FreeJsonAnswer(answer);
         impact_DeleteClient(pClient);
     }*/
```

### 8.1.2 Sample processing loop

see Direct processing

```
bool stopthread=false;
alignas(64) float audio_in[CHUNK_SIZE*CHANNEL_COUNT];
alignas(64) float audio_out[CHUNK_SIZE*CHANNEL_COUNT];
    #if defined(_WIN32) || defined(WIN32)
        // Windows: The stdin, stdout, and stderr streams always open in text mode by default
        _setmode(_fileno(stdin), _O_BINARY);
        _setmode(_fileno(stdout), _O_BINARY);
    #endif

    while (!stopthread)
    {
        // Read a chunk from stdin
        if (std::fread(&audio_in[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdin) !=
       CHUNK_SIZE*CHANNEL_COUNT) {
            std::cerr « "ProcessLoop: input stopped"« std::endl;
            stopthread=true;
            break;
        }
        if (stopthread) break;
        // Process it
        impact_ProcessAudio(pInstance, audio_in, audio_out);
        // Write it to stdout
        if (fwrite(&audio_out[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdout) !=
       CHUNK_SIZE*CHANNEL_COUNT) {
            std::cerr « "Failed to write audio output" « std::endl;
            stopthread=true;
        }
    }
```

### 8.1.3 Sample to stop the processing instance

see Process creation/destruction

```
#if (IMPACT_HAS_WEBSERVER)
    // Stopping the webserver
    std::cerr « "Stopping web server" « std::endl;
    impact_Webserver_Stop(webserver, 1000);
    std::cerr « "Stopping web server DONE" « std::endl;
#endif // (IMPACT_HAS_WEBSERVER)

    // Stopping the processing instance
    std::cerr « "Stopping Update thread" « std::endl;
    impact_StopUpdateThread(pInstance); // this will exit the update thread
    std::cerr « "Stopping Update thread DONE" « std::endl;
    updater.join();
    impact_ExitProcess(pInstance);
```

## 8.2 Environment variables and parameters

### 8.2.1 Parameters defaulting from environment

The following environment variables are used by software library:

**Note**

Those variables can be passed as parameters with impact_InitProcess2

#### 8.2.1.1 Webserver authentification setup :

- `SETUP_USER` see impact_InitProcess
- `SETUP_SECRET` see impact_InitProcess
- `ADMIN_USER` see impact_InitProcess
- `ADMIN_SECRET` see impact_InitProcess

#### 8.2.1.2 Webserver secure :

- `SSL_CERTIFICATE_PATH` : path to the SSL public certificate
- `SSL_PRIVATE_PATH` : path to the SSL private certificate

#### 8.2.1.3 Network access :

- `LICENSE_SERVER` : the SOUND4 license server's URL
- `LICENSE_SERVER_PROXY` : proxy server to use for licenses [user:pass@]host:port
- `LICENSE_CERT_DIR` : optional certificates dirs for HTTPS verification
- `LICENSE_CERT_CHECK` : if "false", do not check HTTPS certificates

### 8.2.2 Parameters

The following parameters can be set before creating instance, but are never taken from environment.

Run mode:

- `FILE_MODE=seconds` : force file mode, settings the maximum estimated audio duration (non realtime, wait license ok to process)
- `STATE_FILENAME=filename` : force this instance to store its state in file named `filename.json` (instead of `state.json`)
- `STATE_ONAIR=filename` : force this instance to store its on air preset name in file named `filename.onair` (instead of none). Used for sharing between instances.
- `STORAGE_RO=notempty` : force the storage in read-only mode
- `STATE_RO=notempty` : force the state in read-only mode (but can save presets/auth)
- `WAIT_LICENSE=notempty` : force to wait the license load before starting (always in file mode)

### 8.2.3 Preset sharing

You can use the same folder for multiple instances so they share their presets.

In this case, you should use a different `STATE_FILENAME` per instance so they keep their settings and on air presets separated. If not, each time settings or onair preset are changed, the changes will be reflected to the other intances using same `STATE_FILENAME`.

You may use the same `STATE_ONAIR` for multiple instances sharing the folder, so that changing the on air preset on one immediatly changes the on air preset on other instances using the same `STATE_ONAIR`. If the instance has pending unsaved changes in the on air preset, it will not be updated automatically to avoid loosing changes.

## 8.3 Global setup

### Classes

- struct SOUND4_ProcessInfo

### Macros

- #define IMPACT_AUDIOFRAME_COUNT 64
- #define IMPACT_CHANNEL_COUNT 2
- #define IMPACT_AUDIO_INPUT_COUNT 1
- #define IMPACT_AUDIO_OUTPUT_COUNT 1
- #define IMPACT_SAMPLE_RATE 48000

### Typedefs

- typedef void(∗ impact_loggerfn) (enum impact_LogSeverity, const char ∗)

### Enumerations

- enum impact_LogSeverity {
  none = 0 , fatal = 1 , error = 2 , warning = 3 ,
  info = 4 , verbose = 5 , verbose2 = 6 , verbose3 = 7 ,
  verbose4 = 8 , verbose5 = 9 }

### Functions

- const struct SOUND4_ProcessInfo ∗ SOUND4_GetProcessInfo ()
- const char ∗ impact_GetVersion ()
- unsigned int impact_GetChunkSizeInFrames ()
- unsigned int ∗ impact_GetPossibleChunkSizeInFrames ()
- unsigned int impact_GetChannelCount ()
- unsigned int impact_GetAudioInputCount ()
- unsigned int impact_GetAudioOutputCount ()
- unsigned int impact_GetSampleRate ()
- void impact_SetLoggerCallback (impact_loggerfn logger)
- void impact_SetLogSeverity (enum impact_LogSeverity severity)

### 8.3.1 Detailed Description

Get information on the library. Set logging mode

### 8.3.2 Macro Definition Documentation

#### 8.3.2.1 IMPACT_AUDIO_INPUT_COUNT

```
#define IMPACT_AUDIO_INPUT_COUNT 1
```

This version number of inputs for this process.

**Remarks**

For binary compatibility, this should be runtime checked to be the same as impact_GetAudioInputCount()

Definition at line 168 of file sound4.impact.cl.h.

#### 8.3.2.2 IMPACT_AUDIO_OUTPUT_COUNT

```
#define IMPACT_AUDIO_OUTPUT_COUNT 1
```

This version number of outputs for this process.

**Remarks**

For binary compatibility, this should be runtime checked to be the same as impact_GetAudioOutputCount()

Definition at line 174 of file sound4.impact.cl.h.

#### 8.3.2.3 IMPACT_AUDIOFRAME_COUNT

```
#define IMPACT_AUDIOFRAME_COUNT 64
```

This version number of frames in a process chunk.

**Remarks**

For binary compatibility, this should be runtime checked to be the same as impact_GetChunkSizeInFrames()

Definition at line 156 of file sound4.impact.cl.h.

#### 8.3.2.4 IMPACT_CHANNEL_COUNT

```
#define IMPACT_CHANNEL_COUNT 2
```

This version number of channels.

**Remarks**

For binary compatibility, this should be runtime checked to be the same as impact_GetChannelCount()

Definition at line 162 of file sound4.impact.cl.h.

#### 8.3.2.5 IMPACT_SAMPLE_RATE

```
#define IMPACT_SAMPLE_RATE 48000
```

This input/output sample-rate.

**Remarks**

For binary compatibility, this should be runtime checked to be the same as impact_GetSampleRate()

Definition at line 180 of file sound4.impact.cl.h.

### 8.3.3 Typedef Documentation

#### 8.3.3.1 impact_loggerfn

```
typedef void(* impact_loggerfn) (enum impact_LogSeverity, const char *)
```

The logging callback format

**Warning**

Do not block/wait in the callback, it can be called from any place in code

**Note**

In all library, all `char*` are **UTF-8** encoded

Definition at line 295 of file sound4.impact.cl.h.

### 8.3.4 Enumeration Type Documentation

#### 8.3.4.1 impact_LogSeverity

```
enum impact_LogSeverity
```

The logging levels

**Enumerator**

| | |
|---|---|
| none | |
| fatal | |
| error | |
| warning | |
| info | |
| verbose | |
| verbose2 | |
| verbose3 | |
| verbose4 | |
| verbose5 | |

Definition at line 276 of file sound4.impact.cl.h.

### 8.3.5 Function Documentation

#### 8.3.5.1 impact_GetAudioInputCount()

```
unsigned int impact_GetAudioInputCount ( )
```

Get the number of inputs for this process

**Remarks**

All inputs have impact_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

**Returns**

The process input count

#### 8.3.5.2 impact_GetAudioOutputCount()

```
unsigned int impact_GetAudioOutputCount ( )
```

Get the number of outputs for this process

**Remarks**

All outputs have impact_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

**Returns**

The process outputs count

#### 8.3.5.3 impact_GetChannelCount()

```
unsigned int impact_GetChannelCount ( )
```

Get the number of channels needed by this process

**Remarks**

Fixed to 2, but should be checked for backward compatibility

**Returns**

The process channels count

#### 8.3.5.4 impact_GetChunkSizeInFrames()

```
unsigned int impact_GetChunkSizeInFrames ( )
```

Get the number of frames in a process chunk.

The output will be available only by chunks of this size.

**Note**

A Frame is a set of impact_GetChannelCount samples

**Remarks**

Fixed to 64, but should be checked for backward compatibility

**Returns**

The process chunk size in frame

**Deprecated** Use impact_GetPossibleChunkSizeInFrames

#### 8.3.5.5 impact_GetPossibleChunkSizeInFrames()

```
unsigned int* impact_GetPossibleChunkSizeInFrames ( )
```

Get the possibles number of frames in a process chunk.

The output will be available only by chunks of this size.

**Note**

A Frame is a set of impact_GetChannelCount samples

**Returns**

An array of possible process chunk size in frame, ending with 0

#### 8.3.5.6 impact_GetSampleRate()

```
unsigned int impact_GetSampleRate ( )
```

Get the processing sample-rate

**Remarks**

For now, always return 48000, but should be checked for backward compatibility

**Returns**

The process sample-rate in Hz

#### 8.3.5.7 impact_GetVersion()

```
const char* impact_GetVersion ( )
```

Get the process version string

**Returns**

version string

#### 8.3.5.8 impact_SetLoggerCallback()

```
void impact_SetLoggerCallback (
            impact_loggerfn logger )
```

Set a callback for logging

If not set, a default logger prints to stderr.

**Warning**

Do not block/wait in the callback, it can be called from any place in code

**Parameters**

| | |
|---|---|
| *logger* | A C callback called for logging. |

#### 8.3.5.9 impact_SetLogSeverity()

```
void impact_SetLogSeverity (
            enum impact_LogSeverity severity )
```

Change the global log severity

Logs above this level will not be printed.

**Parameters**

| | |
|---|---|
| *severity* | Log severity |

#### 8.3.5.10 SOUND4_GetProcessInfo()

```
const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo ( )
```

Get this processor description

Common to all SOUND4 processors.

**Returns**

the process description

## 8.4 Process creation/destruction

### Classes

- struct impact_CInstance
- struct impact_CParameters

### Functions

- struct impact_CParameters ∗ impact_NewParameters ()
- void impact_FreeParameters (struct impact_CParameters ∗params)
- void impact_SetParameter (struct impact_CParameters ∗params, const char ∗name, const char ∗value)
- const char ∗ impact_GetParameter (struct impact_CParameters ∗params, const char ∗name)
- void impact_FreeParameterValue (const char ∗value)
- struct impact_CInstance ∗ impact_InitProcess (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path)
- struct impact_CInstance ∗ impact_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters)
- struct impact_CInstance ∗ impact_InitProcess3 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters, unsigned int frames_per_chunk)
- unsigned int impact_GetProcessChunkFrames (struct impact_CInstance ∗instance)
- int impact_TerminateProcess (struct impact_CInstance ∗instance)
- void impact_ExitProcess (struct impact_CInstance ∗instance)

### 8.4.1 Detailed Description

Create and destroy a process instance, and retrieve some general information about the process.

### 8.4.2 Function Documentation

#### 8.4.2.1 impact_ExitProcess()

```
void impact_ExitProcess (
            struct impact_CInstance * instance )
```

Quit and destroy the instance

**Parameters**

| | |
|---|---|
| *instance* | the instance to destroy |

#### 8.4.2.2 impact_FreeParameters()

```
void impact_FreeParameters (
            struct impact_CParameters * params )
```

Free the parameters created with impact_NewParameters.

#### 8.4.2.3 impact_FreeParameterValue()

```
void impact_FreeParameterValue (
            const char * value )
```

Free a parameter value returned by impact_GetParameter

**Parameters**

| | |
|---|---|
| *value* | the returned value |

#### 8.4.2.4 impact_GetParameter()

```
const char* impact_GetParameter (
            struct impact_CParameters * params,
            const char * name )
```

Get current parameter value.

Returned value must be freed with impact_FreeParameterValue

#### 8.4.2.5 impact_GetProcessChunkFrames()

```
unsigned int impact_GetProcessChunkFrames (
            struct impact_CInstance * instance )
```

Get the number of audio frames in each chunk of processing.

The value reflects the frames_per_chunk parameter passed to impact_InitProcess3

**Parameters**

| | |
|---|---|
| *instance* | the instance to destroy |

**Returns**

the number of audio frames in each chunk of processing

**Note**

A Frame is a set of impact_GetChannelCount samples

#### 8.4.2.6 impact_InitProcess()

```
struct impact_CInstance* impact_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path )
```

Create a process instance with default parameters (from env)

Same as impact_InitProcess2, but with default parameters (from env)

**Deprecated** Use impact_InitProcess3

#### 8.4.2.7 impact_InitProcess2()

```
struct impact_CInstance* impact_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct impact_CParameters * parameters )
```

Create a process instance with default frames_per_chunk (64)

Same as impact_InitProcess3, but with default frames_per_chunk

**Deprecated** Use impact_InitProcess3

#### 8.4.2.8 impact_InitProcess3()

```
struct impact_CInstance* impact_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct impact_CParameters * parameters,
            unsigned int frames_per_chunk )
```

Create a process instance

The output will have to be freed with impact_ExitProcess

**Attention**

The **save_path** must be an existing folder with R/W access, or be NULL (then it uses volatile memory storage)

If the save_path is NULL or the folder is empty, a default configuration with a default preset will be loaded, unless using Custom preset managagement.

If used in a container, we recommend to put a volume for this path, so it will survive updates.

**Web login initialization**

If the storage is not yet initialized, you can choose how to protect the web access.
There are 2 modes:

1. Setup mode: you define a setup account, which can then create an administrator account.
   The setup account always keeps the access and can delete the administrator.
2. Admin mode: you define the administrator, which will be the single administrator.

To select the mode, you have to put in the parameters

1. `SETUP_USER` (and optionally `SETUP_SECRET`) to select Setup mode
2. `ADMIN_USER` (and optionally `ADMIN_SECRET`) to select Admin mode
3. none: the web access is opened, no login will be asked

If you do not set the `xxx_SECRET`, then the setuppassword will be randomly generated and print in logs (at info severity).

You can also use the JSON api to change this after the storage is created : see Change setup account

**License access**

The licenses are retrieved via Internet and access some servers. You may use an alternate server by changing `LICENSE_SERVER` in the parameters.

**Parameters**

| | |
|---|---|
| *LoginKey* | SOUND4 Login key (was User ID) |
| *RadioName* | SOUND4 RadioName |
| *Access_Key_ID* | SOUND4 AWS access ID |
| *Access_Key_Secret* | SOUND4 AWS access Secret |
| *save_path* | Path where we can load/save the configuration. If NULL, memory storage will be used. |
| *parameters* | Parameters |
| *frames_per_chunk* | The number of audio frames in each chunk of processing. Must be in the list returned by impact_GetPossibleChunkSizeInFrames |

**Returns**

A new process instance

#### 8.4.2.9 impact_NewParameters()

```
struct impact_CParameters* impact_NewParameters ( )
```

Create parameters to use with impact_InitProcess2.

The output will have to be freed with impact_FreeParameters

The default values are taken from environment (for compatibility).

#### 8.4.2.10 impact_SetParameter()

```
void impact_SetParameter (
            struct impact_CParameters * params,
            const char * name,
            const char * value )
```

Set a new parameter.

Some values are in taken from env, so see Environment variables and parameters for usable variables

See also Custom preset managagement

#### 8.4.2.11 impact_TerminateProcess()

```
int impact_TerminateProcess (
            struct impact_CInstance * instance )
```

Terminate the processing

This is useful only with file processing, to get the final status. For some plugins, this is mandatory

**Parameters**

| | |
|---|---|
| *instance* | the instance to terminate |

**Returns**

0 is success, <0 if error, >0 if some status

## 8.5 Custom preset managagement

This allows to provide a customized preset management.

### Typedefs

- typedef wchar_t fs_char
- typedef char ∗(∗ impact_storage_reader) (const fs_char ∗filename, void ∗userdata)
- typedef void(∗ impact_storage_reader_free) (char ∗content, void ∗userdata)
- typedef int(∗ impact_storage_writer) (const fs_char ∗filename, const char ∗content, void ∗userdata)
- typedef int(∗ impact_storage_exists) (const fs_char ∗filename, void ∗userdata)
- typedef fs_char ∗∗(∗ impact_storage_getall) (void ∗userdata)
- typedef void(∗ impact_storage_getall_free) (fs_char ∗∗all, void ∗userdata)
- typedef int(∗ impact_storage_remove) (const fs_char ∗filename, void ∗userdata)
- typedef int(∗ impact_storage_rename) (const fs_char ∗from, const fs_char ∗to, void ∗userdata)

### Enumerations

- enum impact_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }

## Functions

- void impact_SetPresetManager (struct impact_CParameters *params, impact_storage_reader reader, impact_storage_reader_free, impact_storage_writer writer, impact_storage_exists exists, impact_storage_getall getall, impact_storage_getall_free getall_free, impact_storage_remove remove, impact_storage_rename rename, int IsReadOnly, void *userdata)
- void impact_PresetManager_InformChange (struct impact_CInstance *instance, const fs_char *relative_path, enum impact_PresetChange_Kind change_kind)

### 8.5.1 Detailed Description

This allows to provide a customized storage for presets and settings.

By default, the library store data on disk or in memory (see `save_path` parameter in impact_InitProcess2), but you may use these functions to store data.

It can be used for remote storage, database storage, ...

Known files:

- Files with name ending with .pst are presets
- `state.json` is the instance state file, holding settings and unsaved preset changes.
- `auth.json` is the storage for web authentication settings.
- *.onair are used for sharing preset on air name between instances (see `STATE_ONAIR` in Parameters).

**Attention**

state.json name can be overriden per instance with `STATE_FILENAME` in parameters, see Parameters.

File names are in OS native characters ( `wchar_t` for Windows, `char` in **UTF-8** for others)

### 8.5.2 Typedef Documentation

#### 8.5.2.1 fs_char

```
typedef wchar_t fs_char
```

Definition at line 563 of file sound4.impact.cl.h.

#### 8.5.2.2 impact_storage_exists

```
typedef int(* impact_storage_exists) (const fs_char *filename, void *userdata)
```

Function to call to test if a stored file exists.

**Parameters**

| | |
|---|---|
| *filename* | the name of the file to test. |
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

-1 if not exists, 0 if it exists

Definition at line 602 of file sound4.impact.cl.h.

#### 8.5.2.3 impact_storage_getall

```
typedef fs_char**(* impact_storage_getall) (void *userdata)
```

Function to call to get all stored files' name. Result has to be freed with impact_storage_getall_free callback

**Parameters**

| | |
|---|---|
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

A NULL terminated array of file names

Definition at line 611 of file sound4.impact.cl.h.

#### 8.5.2.4 impact_storage_getall_free

```
typedef void(* impact_storage_getall_free) (fs_char **all, void *userdata)
```

Function to call to free the result of getall.

**Parameters**

| | |
|---|---|
| *all* | the returned value from impact_storage_getall |
| *userdata* | the userdata passed to impact_SetPresetManager |

Definition at line 619 of file sound4.impact.cl.h.

#### 8.5.2.5 impact_storage_reader

```
typedef char*(* impact_storage_reader) (const fs_char *filename, void *userdata)
```

Function to call to read a stored file, returning its content.

**Parameters**

| | |
|---|---|
| *filename* | the name of the file to read. |
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

the **UTF-8** content of the file

Definition at line 575 of file sound4.impact.cl.h.

#### 8.5.2.6 impact_storage_reader_free

```
typedef void(* impact_storage_reader_free) (char *content, void *userdata)
```

Function to call to free the result of impact_storage_reader.

**Parameters**

| | |
|---|---|
| *content* | the returned value from impact_storage_reader |
| *userdata* | the userdata passed to impact_SetPresetManager |

Definition at line 583 of file sound4.impact.cl.h.

#### 8.5.2.7 impact_storage_remove

```
typedef int(* impact_storage_remove) (const fs_char *filename, void *userdata)
```

Function to call to remove a stored file.

**Parameters**

| | |
|---|---|
| *filename* | the name of the file to remove. |
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

-1 if failed, 0 if OK

Definition at line 628 of file sound4.impact.cl.h.

#### 8.5.2.8 impact_storage_rename

```
typedef int(* impact_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)
```

Function to call to rename a stored file.

**Parameters**

| | |
|---|---|
| *from* | The source file name (must exist) |
| *to* | The destination file name (must not exist) |
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

-1 if failed, 0 if OK

Definition at line 638 of file sound4.impact.cl.h.

#### 8.5.2.9 impact_storage_writer

```
typedef int(* impact_storage_writer) (const fs_char *filename, const char *content, void *userdata)
```

Function to call to write a stored file.

**Parameters**

| | |
|---|---|
| *filename* | the name of the file to write. |
| *content* | the **UTF-8** content to write to the file |
| *userdata* | the userdata passed to impact_SetPresetManager |

**Returns**

-1 if failed, 0 if OK

Definition at line 593 of file sound4.impact.cl.h.

### 8.5.3 Enumeration Type Documentation

#### 8.5.3.1 impact_PresetChange_Kind

```
enum impact_PresetChange_Kind
```

Kind of change on a storage path

**Enumerator**

| | |
|---|---|
| change_kind_created | Path has been created. |
| change_kind_modified | Path has been modified. |
| change_kind_deleted | Path has been deleted. |

Definition at line 671 of file sound4.impact.cl.h.

### 8.5.4 Function Documentation

#### 8.5.4.1 impact_PresetManager_InformChange()

```
void impact_PresetManager_InformChange (
            struct impact_CInstance * instance,
            const fs_char * relative_path,
            enum impact_PresetChange_Kind change_kind )
```

Inform that a change happened in the storage. This must be used only in conjonction with impact_SetPresetManager

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *relative_path* | The path which changed, relative to the root storage |
| *change_kind* | What happened to this path |

#### 8.5.4.2 impact_SetPresetManager()

```
void impact_SetPresetManager (
            struct impact_CParameters * params,
            impact_storage_reader reader,
            impact_storage_reader_free ,
            impact_storage_writer writer,
            impact_storage_exists exists,
            impact_storage_getall getall,
            impact_storage_getall_free getall_free,
            impact_storage_remove remove,
            impact_storage_rename rename,
            int IsReadOnly,
            void * userdata )
```

Setup a custom preset manager. This overrides the save_path in impact_InitProcess2

To dynamically update when storage change, see impact_PresetManager_InformChange

**Parameters**

| | |
|---|---|
| *params* | The params to set up |
| *reader* | The function to read a file |
| *writer* | The function to write a file |
| *exists* | The function to test for a file presence |
| *getall* | The function to get all files in storage |
| *getall_free* | The function to free the result from getall |
| *remove* | The function to remove a file from storage |
| *rename* | The function to rename a file |
| *IsReadOnly* | Set to 1 if storage is read-only |
| *userdata* | The userdata parameter for callbacks |

## 8.6 Metadata Management

### Functions

- void impact_SetMetadata (struct impact_CInstance ∗instance, const char ∗key, const char ∗value)
- void impact_SetMetadataMulti (struct impact_CInstance ∗instance, const char ∗∗keyvalue)
- const char ∗∗ impact_GetMetadataInfos (struct impact_CInstance ∗instance)
- void impact_FreeMetadataInfos (struct impact_CInstance ∗instance, const char ∗∗infos)

### 8.6.1 Detailed Description

Metadata handling.

Metadata is used on-need, and might be used by extensions modules (Watermarking, Streaming, ...)

### 8.6.2 Function Documentation

#### 8.6.2.1 impact_FreeMetadataInfos()

```
void impact_FreeMetadataInfos (
            struct impact_CInstance * instance,
            const char ** infos )
```

Frees the information returned by impact_GetMetadataInfos.

**Parameters**

| | |
|---|---|
| *instance* | the instance |
| *infos* | the infos |

#### 8.6.2.2 impact_GetMetadataInfos()

```
const char** impact_GetMetadataInfos (
            struct impact_CInstance * instance )
```

Get information on known metadata keys.

**Parameters**

| | |
|---|---|
| *instance* | the instance |

**Returns**

a list of key,descriptions, null terminated, of known metadata keys used in process and extensions. May be null. Should be freed with impact_FreeMetadataInfos

#### 8.6.2.3 impact_SetMetadata()

```
void impact_SetMetadata (
            struct impact_CInstance * instance,
            const char * key,
            const char * value )
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible.

Setting a null value erase the metadata, which is different from setting an empty string. Initial metadata might be set as soon as instance is created (and before update thread).

**See also**

impact_SetMetadataMulti

**Parameters**

| | |
|---|---|
| *instance* | the instance |
| *key* | the metadata key. If key contains a dot (like wm.KEY), it will be sent only to the matching plugin (ie wm here) without the prefix (ie KEY here). Use empty prefix for routing to main only. |
| *value* | the metadata value or nullptr to erase it. |

#### 8.6.2.4 impact_SetMetadataMulti()

```
void impact_SetMetadataMulti (
            struct impact_CInstance * instance,
            const char ** keyvalue )
```

Adds multiple metadata to the process.

This works like impact_SetMetadata but with multiple metadata at once.

**See also**

impact_SetMetadata

**Parameters**

| | |
|---|---|
| *instance* | the instance |
| *keyvalue* | a list of key/value pair, ending with null key. |

## 8.7 Update Management

### Functions

- void impact_StartUpdateThread (struct impact_CInstance ∗instance, unsigned int port)
- void impact_StopUpdateThread (struct impact_CInstance ∗instance)
- int impact_WaitUpdateThreadReady (struct impact_CInstance ∗instance, int milliseconds)

### 8.7.1 Detailed Description

Manage the process update.

### 8.7.2 Function Documentation

#### 8.7.2.1 impact_StartUpdateThread()

```
void impact_StartUpdateThread (
            struct impact_CInstance * instance,
            unsigned int port )
```

Update function

This needs to run in a thread, which can have low priority. It will also allow the JSON access via TCP.

**Remarks**

This function does not return until it is stopped with impact_StopUpdateThread

**Parameters**

| | |
|---|---|
| *instance* | the instance |
| *port* | the port to listen for incoming JSON connections. If 0, the JSON server is not accessible |

#### 8.7.2.2 impact_StopUpdateThread()

```
void impact_StopUpdateThread (
            struct impact_CInstance * instance )
```

Stop the update thread

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

#### 8.7.2.3 impact_WaitUpdateThreadReady()

```
int impact_WaitUpdateThreadReady (
            struct impact_CInstance * instance,
            int milliseconds )
```

Wait for the update thread to be ready

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *milliseconds* | maximum time to wait in milliseconds |

**Returns**

-1 if timeout waiting for the thread, 0 if OK

## 8.8 Direct processing

### Functions

- void impact_ProcessAudio (struct impact_CInstance ∗instance, const float ∗input, float ∗output)
- void impact_ProcessAudio_Planar (struct impact_CInstance ∗instance, float const ∗const ∗input, float ∗const ∗output)
- float ∗ impact_GetBufferIn (struct impact_CInstance ∗instance)
- float ∗ impact_GetBufferOut (struct impact_CInstance ∗instance)
- unsigned int impact_GetEstimatedDelay (struct impact_CInstance ∗instance)

### 8.8.1 Detailed Description

Optimized processing with direct in->out.

### 8.8.2 Function Documentation

#### 8.8.2.1 impact_GetBufferIn()

```
float* impact_GetBufferIn (
            struct impact_CInstance * instance )
```

Get preallocated aligned buffer for audio input for impact_ProcessAudio()

The buffer is a float[256 ∗ impact_GetAudioInputCount() ∗ impact_GetChannelCount()]

So for this process/version : float[256∗1∗2]

**Note**

The calling process can allocate the buffer itself, but it must be aligned.

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

An audio input dedicated buffer.

#### 8.8.2.2 impact_GetBufferOut()

```
float* impact_GetBufferOut (
            struct impact_CInstance * instance )
```

Get preallocated aligned buffer for audio output for impact_ProcessAudio()

The buffer is a float[256 ∗ impact_GetAudioOutputCount() ∗ impact_GetChannelCount()]

So for this process/version : float[256∗1∗2]

**Note**

The calling process can allocate the buffer itself, but it must be aligned.

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

An audio output dedicated buffer.

#### 8.8.2.3 impact_GetEstimatedDelay()

```
unsigned int impact_GetEstimatedDelay (
            struct impact_CInstance * instance )
```

Get the delay (estimated) of current processing algorithm

**Remarks**

This is not very accurate, it does not take into account group delays.

**Returns**

Estimated delay in frames

#### 8.8.2.4 impact_ProcessAudio()

```
void impact_ProcessAudio (
            struct impact_CInstance * instance,
            const float * input,
            float * output )
```

Process an audio chunk

A chunk contains impact_GetProcessChunkFrames frames, set at init by frames_per_chunk in impact_InitProcess3

Input has impact_GetAudioInputCount() ∗ impact_GetChannelCount() channels

Ouput has impact_GetAudioOutputCount() ∗ impact_GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

**Warning**

Buffers must be 64-byte aligned

**Note**

With 64 frames per chunk and 2 channels and 1 audio input, input buffer size is 64∗1∗2∗4=512 bytes

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *input* | audio input buffer (64 byte aligned) or NULL to use internal buffer impact_GetBufferIn |
| *output* | audio output buffer (64 byte aligned) or NULL to use internal buffer impact_GetBufferOut |

#### 8.8.2.5 impact_ProcessAudio_Planar()

```
void impact_ProcessAudio_Planar (
            struct impact_CInstance * instance,
            float const *const * input,
            float *const * output )
```

Process an audio chunk of planar audio

A chunk contains impact_GetProcessChunkFrames frames, set at init by frames_per_chunk in impact_InitProcess3

Input has impact_GetAudioInputCount() ∗ impact_GetChannelCount() channels

Ouput has impact_GetAudioOutputCount() ∗ impact_GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are NOT interlaced, so the buffers are arrays of planar audio : [ [L0,L1,...,Ln] , [R0,R1,...,Rn] ] (L=Left, R=Right)

In case of multiple audio input/output, buffer are [ [aL...], [aR...], [bL...], [bR...], ... ]

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *input* | array of audio input buffers for each input channel (no alignment required) |
| *output* | array of audio output buffers for each input channel (no alignment required) |

## 8.9 Audio managing (any audio)

### Enumerations

- enum impact_SampleFormat {
  INVALID_FORMAT =0 , S16_LE , S16_BE , S24_LE ,
  S24_BE , S32_LE , S32_BE , F32_LE ,
  F32_BE , S16_NATIVE =S16_LE , S24_NATIVE =S24_LE , S32_NATIVE =S32_LE ,
  F32_NATIVE =F32_LE }

### Functions

- const char ∗ impact_GetFormatName (const enum impact_SampleFormat fmt)
- enum impact_SampleFormat impact_GetFormatFromName (const char ∗name)

- unsigned int impact_GetBytesFromFormat (const enum impact_SampleFormat fmt)
- unsigned int impact_GetMaxPacketFrame (struct impact_CInstance *instance)
- unsigned int impact_AddAudio (struct impact_CInstance *instance, const uint8_t *payload, unsigned int n↩ Frame, enum impact_SampleFormat fmt)
- unsigned int impact_AddPadAudio (struct impact_CInstance *instance)
- unsigned int impact_GetOutputCount (struct impact_CInstance *instance)
- unsigned int impact_GetAudio (struct impact_CInstance *instance, uint8_t *payload, unsigned int max_n↩ Frame, enum impact_SampleFormat fmt)

## 8.9.1 Detailed Description

This allows various payload size and formats, at the expense of internal conversions. This can be useful when receiving audio from network.

**Remarks**

It is more optimized to use Direct processing.

## 8.9.2 Enumeration Type Documentation

### 8.9.2.1 impact_SampleFormat

`enum impact_SampleFormat`

Supported formats for audio I/O

**Enumerator**

| | |
|---|---|
| INVALID_FORMAT | Invalid. |
| S16_LE | 16-bit signed integer, little-endian |
| S16_BE | 16-bit signed integer, big-endian |
| S24_LE | 24-bit signed integer, little-endian |
| S24_BE | 24-bit signed integer, big-endian |
| S32_LE | 32-bit signed integer, little-endian |
| S32_BE | 32-bit signed integer, big-endian |
| F32_LE | 32-bit floating-point, little-endian |
| F32_BE | 32-bit floating-point, big-endian |
| S16_NATIVE | 16-bit signed integer, native |
| S24_NATIVE | 24-bit signed integer, native |
| S32_NATIVE | 32-bit signed integer, native |
| F32_NATIVE | 32-bit floating-point, native |

Definition at line 894 of file sound4.impact.cl.h.

### 8.9.3 Function Documentation

#### 8.9.3.1 impact_AddAudio()

```
unsigned int impact_AddAudio (
            struct impact_CInstance * instance,
            const uint8_t * payload,
            unsigned int nFrame,
            enum impact_SampleFormat fmt )
```

Convert and process the payload.

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *payload* | the binary raw audio buffer |
| *nFrame* | the number of frames in the payload |
| *fmt* | the format of the payload |

**Returns**

number of unprocessed frames (if can not store output) if non zero, consume output and restart where it stopped

#### 8.9.3.2 impact_AddPadAudio()

```
unsigned int impact_AddPadAudio (
            struct impact_CInstance * instance )
```

Add silence to finish the current audio chunk and process it

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

number of added silence frame

#### 8.9.3.3 impact_GetAudio()

```
unsigned int impact_GetAudio (
            struct impact_CInstance * instance,
```

```
uint8_t * payload,
unsigned int max_nFrame,
enum impact_SampleFormat fmt )
```

Consume the output

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |
| *payload* | the binary raw audio buffer to store output |
| *max_nFrame* | the maximum number of frames that can be stored in the payload |
| *fmt* | the wanted format of the payload |

**Returns**

number of frames put in the buffer

#### 8.9.3.4 impact_GetBytesFromFormat()

```
unsigned int impact_GetBytesFromFormat (
            const enum impact_SampleFormat fmt )
```

Get the format byte size

**Parameters**

| | |
|---|---|
| *fmt* | the format |

**Returns**

the number of bytes used by this format for a single channel audio sample

#### 8.9.3.5 impact_GetFormatFromName()

```
enum impact_SampleFormat impact_GetFormatFromName (
            const char * name )
```

Convert string to format

**Parameters**

| | |
|---|---|
| *name* | the format string |

**Returns**

the matching format, or INVALID_FORMAT if invalid.

#### 8.9.3.6 impact_GetFormatName()

```
const char* impact_GetFormatName (
            const enum impact_SampleFormat fmt )
```

Convert format to string

**Parameters**

| | |
|---|---|
| *fmt* | the format |

**Returns**

the string name of the format

#### 8.9.3.7 impact_GetMaxPacketFrame()

```
unsigned int impact_GetMaxPacketFrame (
            struct impact_CInstance * instance )
```

Maximum packet size (in frames) that can be queued at output.

**Parameters**

| | |
|---|---|
| *instance* | The process instance |

**Returns**

Maximum packet size (in frames) that can be queued at output.

#### 8.9.3.8 impact_GetOutputCount()

```
unsigned int impact_GetOutputCount (
            struct impact_CInstance * instance )
```

Get the available output size

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

number of available frames at output

## 8.10 Audio convert helpers

### Functions

- void impact_AudioConvertFrom (const uint8_t ∗payload, float ∗output, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_AudioConvertTo (const float ∗input, uint8_t ∗payload, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_StereoToMono (const float ∗input, float ∗output, size_t nFrame)
- void impact_MonoToStereo (const float ∗input, float ∗output, size_t nFrame)
- void impact_StereoToMono_Planar (const float ∗inputL, const float ∗inputR, float ∗output, size_t nFrame)
- void impact_MonoToStereo_Planar (const float ∗input, float ∗outputL, float ∗outputR, size_t nFrame)
- void impact_AudioMonoFromLiveStereo (const uint8_t ∗payload, float ∗output)
- void impact_AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)

### 8.10.1 Detailed Description

### 8.10.2 Function Documentation

#### 8.10.2.1 impact_AudioConvertFrom()

```
void impact_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum impact_SampleFormat fmt )
```

Convert the payload to floating-point array

**Warning**

Output buffer must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *payload* | the binary raw audio buffer |
| *output* | audio output buffer (64 byte aligned) |
| *nSpl* | the number of samples (not frames) |
| *fmt* | the format of the payload |

#### 8.10.2.2 impact_AudioConvertTo()

```
void impact_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum impact_SampleFormat fmt )
```

Convert floating-point array to payload

**Warning**

Input buffer must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *input* | audio input buffer (64 byte aligned) |
| *payload* | the binary raw audio buffer |
| *nSpl* | the number of samples (not frames) |
| *fmt* | the format of the payload |

#### 8.10.2.3 impact_AudioMonoFromLiveStereo()

```
void impact_AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )
```

Convert Livewire Livestereo payload to floating-point Mono (12 samples)

**Parameters**

| | |
|---|---|
| *payload* | Livewire Live Stereo raw buffer |
| *output* | audio output mono buffer (64 byte aligned) |

#### 8.10.2.4 impact_AudioMonoToLiveStereo()

```
void impact_AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )
```

Convert floating-point Mono (12 samples) to Livewire Livestereo payload

**Parameters**

| | |
|---|---|
| *input* | audio input mono buffer (64 byte aligned) |
| *payload* | Livewire Live Stereo raw buffer |

#### 8.10.2.5 impact_MonoToStereo()

```
void impact_MonoToStereo (
            const float * input,
            float * output,
            size_t nFrame )
```

Convert floating-point Mono to Stereo (L,L)

**Warning**

Buffers must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *input* | audio input mono buffer (64 byte aligned) |
| *output* | audio output stereo buffer (64 byte aligned) |
| *nFrame* | the number of frames |

#### 8.10.2.6 impact_MonoToStereo_Planar()

```
void impact_MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nFrame )
```

Convert floating-point Mono to Stereo (planar) (L,L)

**Warning**

Buffers must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *input* | audio input mono buffer (64 byte aligned) |
| *outputL* | audio output left buffer (64 byte aligned) |
| *outputR* | audio output right buffer (64 byte aligned) |
| *nFrame* | the number of frames |

#### 8.10.2.7 impact_StereoToMono()

```
void impact_StereoToMono (
            const float * input,
            float * output,
            size_t nFrame )
```

Convert floating-point Stereo to Mono (L+R)/2

**Warning**

Buffers must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *input* | audio input stereo buffer (64 byte aligned) |
| *output* | audio output mono buffer (64 byte aligned) |
| *nFrame* | the number of frames |

#### 8.10.2.8 impact_StereoToMono_Planar()

```
void impact_StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nFrame )
```

Convert floating-point Stereo (planar) to Mono (L+R)/2

**Warning**

Buffers must be 64-byte aligned

**Parameters**

| | |
|---|---|
| *inputL* | audio input left buffer (64 byte aligned) |
| *inputR* | audio input right buffer (64 byte aligned) |
| *output* | audio output mono buffer (64 byte aligned) |
| *nFrame* | the number of frames |

## 8.11 Client's JSon config interfacing

### Classes

- struct impact_CClientInstance

### Functions

- struct impact_CClientInstance ∗ impact_NewClient (struct impact_CInstance ∗instance)
- void impact_DeleteClient (struct impact_CClientInstance ∗client)
- const char ∗ impact_ProcessJson (struct impact_CClientInstance ∗client, const char ∗json_str, int ∗need_save)
- void impact_FreeJsonAnswer (const char ∗json_str)
- int impact_SaveState (struct impact_CInstance ∗instance)

### 8.11.1 Detailed Description

Allow low-level JSON configuration. see also JSON syntax

### 8.11.2 Function Documentation

#### 8.11.2.1 impact_DeleteClient()

```
void impact_DeleteClient (
            struct impact_CClientInstance * client )
```

Deletes a client

**Parameters**

| | |
|---|---|
| *client* | the client instance |

#### 8.11.2.2 impact_FreeJsonAnswer()

```
void impact_FreeJsonAnswer (
            const char * json_str )
```

Free a JSON answer returned by impact_ProcessJson

**Parameters**

| | |
|---|---|
| *json_str* | the JSON request |

#### 8.11.2.3 impact_NewClient()

```
struct impact_CClientInstance* impact_NewClient (
            struct impact_CInstance * instance )
```

Creates a new client

**Remarks**

not needed if using a port for JSON socket in Update Management

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

A new client instance

#### 8.11.2.4 impact_ProcessJson()

```
const char* impact_ProcessJson (
            struct impact_CClientInstance * client,
            const char * json_str,
            int * need_save )
```

Process a JSON request

For syntax, see JSON syntax

**Note**

In all library, all `char*` are **UTF-8** encoded

**Attention**

the return value needs to be freed with impact_FreeJsonAnswer

**Parameters**

| | |
|---|---|
| *client* | the client instance |
| *json_str* | the JSON request |
| *need_save* | on return will be 1 is saving is required (maybe delayed) |

**Returns**

a JSON answer (to be freed with impact_FreeJsonAnswer)

#### 8.11.2.5 impact_SaveState()

```
int impact_SaveState (
            struct impact_CInstance * instance )
```

Save the current state

Needed when impact_ProcessJson returned with need_save to 1. However it can be postponed as much as wanted.

**Remarks**

Works only when update thread is running

**Parameters**

| | |
|---|---|
| *instance* | the processing instance |

**Returns**

0 if success, <0 on error

## 8.12 JSON syntax

JSON protocol to configure the process

### 8.12.1 Commands

`{ "get" : 'object' }`

get values.
If `object` is null, get all parameters, and so show the possibilities
`{ "set" : 'object' }`

set values.
`object` can be any subpart, only those will be changed
`{ "change" : 'object' }`

like get, but shows only changes since last request.
If `object` is null, get all changes.
Each call resets the state for this connection to current.

**Note**

In Impact.cl version <= 1.1.7, the state resetted also for filtered out objects. So avoid alterning `object` filter between calls, or you will miss some changes.

```
{ "load" : "`presetname`" }
```

load the preset

```
{ "save" : "`presetname`" }
```

save the preset.
If null is given, save onair preset

```
{ "delete" : "`presetname`" }
```

delete the preset.
can also be an array of presetnames

```
{ "factoryload" : "`presetname`" }
```

Create a preset from factory and load it

```
{ "rename" : { "preset" : "`presetname`", "name": "`newname`" } }
```

Rename a preset

```
{ "import" : { "preset" : "`presetname`", "content": `presetcontent` } }
```

Import a preset

```
{ "export" : "`presetname`"  }
```

Export a preset: get preset's content so it can be imported later

```
{ "new" : { "name": "`presetname`", "factory": "`presetname`" } }
```

Create a new preset from a factory.
Can use array for multiple.

```
{ "undo" : `count`  }
```

Undo last(s) `count` preset change

```
{ "redo" : `count`  }
```

Redo last(s) `count` preset change

See also JSON syntax for internal setup

### 8.12.2 Answers

Answer is also a JSON object, starting with the command for `get` and `change`.

```
{ "get" : ... }
{ "change" : ... }
```

For other commands, if there is no error the answer is

```
{ "OK" : {"command" : "original command" } }
```

If there is an error, answer has the syntax

```
{ "ERROR" : {"command" : "faulty command", "message" : "explaining"} }
```

and may have more properties. The message may be empty.

### 8.12.3 Examples

#### 8.12.3.1 Get status

```
{"get":{"status":null}}
```

answer:

```
{"get":{"status":{"preset_changed":false,"license_valid":true,"version":"1.1.19"}}}
```

#### 8.12.3.2 Get all changes since last request

```
{"change":null}
```

answer:

```
{"change":{
        "processorname":"Best Radio of the world",
        "settings":{"general":{"operate":true},...},
        "preset":{"setup": {"gate_on": true, ...},...},
        "vu":{"setup": null,"in":{...}},
        ...
        }
```

#### 8.12.3.3 Check license

```
{"get":{"status":{"license_valid":null}}}
```

answer:

```
{"get":{"status":{"license_valid":true}}}
```

#### 8.12.3.4 Set Processor name

```
{"set":{"processorname":"Best Radio of the world"}}
```

answer:

```
{"OK":"set"}
```

#### 8.12.3.5 Get presetlist

```
{"get":{"presetlist":null}}
```

answer:

```
{"get":{"presetlist":[
            {"name":"Urban","factory_name":"Urban","create":"2020-10-15 16:19:17","modify":"2020-10-15
        16:19:17"},
            {"name":"Rock Classic","factory_name":"Rock Classic","create":"2020-10-15
        16:19:17","modify":"2020-10-15 16:19:17"},
            {"name":"Rock","factory_name":"Rock","create":"2020-10-15 16:19:17","modify":"2020-10-15
        16:19:17"},
            ...
        ]}}
```

#### 8.12.3.6 Load a preset

```
{"load":"Rock Classic"}
```

answer:

```
{"OK":"load"}
```

#### 8.12.3.7 Get information on preset on air

```
{"get":{"presetinfo":null}}
```

answer:

```
{"get":{"presetinfo":{"name":"Rock Classic","factory_name":"Rock Classic","create":"2020-10-15
        16:19:17","modify":"2020-10-15 16:19:17"}}}
```

#### 8.12.3.8 Set input gain

```
{"set":{"settings":{"gain":{"in":-3.0}}}}
```

answer:

```
{"OK":"set"}
```

#### 8.12.3.9 Set bypass

```
{"set":{"settings":{"general":{"operate":false}}}}
```

answer:
```
{"OK":"set"}
```

#### 8.12.3.10 Export a preset

```
{"export":"MyPresetName"}
```

answer:
```
{"export":{"processor":"Best Radio of the world","presets":{
           "MyPresetName":{
               "presetinfo":{"name":"MyPresetName","factory_name":"Open Sound","create":"2020-10-16
       06:53:20","modify":"2020-10-16 06:53:26"},
               "preset":{"setup":{"gate":-39.0}},
               "undoredo":{"undo":[{"time":"2020-10-16
       06:53:23","action":{"preset":{"setup":{"gate":-30.0}}}}],"redo":[],"has_undo":true,"has_redo":false}
               }
           }}}
```

**Note**

the object *export.presets["MyPresetName"]* in the answer is the one to use as *content* in Import a preset

#### 8.12.3.11 Import a preset

```
{"import":{"preset":"MyNewPreset","content":{
           "presetinfo":{"name":"MyPresetName","factory_name":"Open Sound","create":"2020-10-16
       06:53:20","modify":"2020-10-16 06:53:26"},
           "preset":{"setup":{"gate":-39.0}},
           "undoredo":{"undo":[{"time":"2020-10-16
       06:53:23","action":{"preset":{"setup":{"gate":-30.0}}}}],"redo":[],"has_undo":true,"has_redo":false}
           }
       }}
```

answer:
```
{"OK":"import"}
```

## 8.13 JSON syntax for internal setup

JSON commands for internal setup and information.

This is used internally for the web server.

This API might change in the future.

Those commands can be used when directly interfacing to the process (see Client's JSon config interfacing). They are blocked web going through web server ( `http://.../json`)

The Authentication is effective only through the web server. The direct JSON commands are not filtered.

### 8.13.1 Get process and modules information

```
{"_info": null}
```

answer:
```
{"_info":{"product":"SOUND4 x1.CLOUD","version": "1.3.7","title": "x1
       .CLOUD","preset_extension":"clx1","use_presets":true,"use_undoredo":true,
       "_modules_":["","wm"],
       "_modules_infos_":{
            "wm":{"product":"SOUND4 WM","version": "0.1.0","title":"SOUND4
       WM","use_presets":false,"use_undoredo":false}
       }}}
```

### 8.13.2 Authentication management

Some API are available only if there is a setup/admin account.

Secret stored is
`MD5(username+":"+realm+":"+password)`

#### 8.13.2.1 Difference between setup and admin account

There are 2 modes:

1. Setup mode: you define a setup account, which can then create an administrator account.
   The setup account always keeps the access and can delete the administrator.
2. Admin mode: you define the administrator, which will be the single administrator.

From JSON, changing setup_is_admin can change the mode.

#### 8.13.2.2 Access encoding

Access is a combinaison of letters.

- **'S'** Setup account : Allow Admin and delete admin
- **'A'** Admin access : Access to all overrides any other except setup
- **'x'** eXecute : change preset, list presets and preset infos and access settings.general fields
- **'v'** View : read preset and settings values
- **'w'** Write : change preset/settings (and undo/redo)
- **'s'** Save : create/save/delete/rename/import preset
- **'p'** Password : change own password
- **'-'** inactive : account is inactive

#### 8.13.2.3 Change setup account

Allows to setup a setup account (even when is was not configured) or to remove it

##### 8.13.2.3.1 Set the setup/admin account

```
{"_auth":{"init":{"setup_account":{"user":"admin","secret":"be666f22c03f199e05d3805430d69599"},"setup_is_admin":true,"un
```

answer:
```
{"OK":{"command":"_auth"}}
```

##### 8.13.2.3.2 Set the setup/admin account with clear password

```
{"_auth":{"init":{"setup_account":{"user":"admin","secret_clear":"admin"},"setup_is_admin":true,"unlogged_access":"rx"}
```

answer:
```
{"OK":{"command":"_auth"}}
```

#### 8.13.2.3.3 Remove the setup/admin account `{"_auth":{"init":{"setup_account":null}}}`

answer:
```
{"OK":{"command":"_auth"}}
```

### 8.13.2.4 Get user info

```
{"_auth":{"get":{"user":"username"}}}
```

answer:
```
{"_auth":{"get":{"user":"username","secret":"secrethash","access":"rwx"}}}
```

### 8.13.2.5 Add or modify a user

```
{"_auth":{"add":{"user":"username","secret":"secrethash","access":"rwx"}}}
```

answer:
```
{"_auth":{"add":{"user":"username","secret":"secrethash","access":"rwx"}}}
```

### 8.13.2.6 Removes a user

```
{"_auth":{"remove":{"user":"username"}}
```

answer:
```
{"_auth":{"remove":{"user":"username"}}
```

### 8.13.2.7 List users

```
{"_auth":{"list":null}}
```

answer:
```
{"_auth":{"list":["user1","user2",...]}}
```

### 8.13.2.8 Get/set unlogged access

```
{"_auth":{"get":{"unlogged_access":null}}}
```

answer:
```
{"_auth":{"get":{"unlogged_access":"rwx"}}}
```

### 8.13.2.9 Get realm

```
{"_auth":{"get":{"realm":null}}}
```

answer:
```
{"_auth":{"get":{"realm":"Authentification is required"}}}
```

### 8.13.2.10 Get API version

```
{"_auth":{"get":{"api_version":null}}}
```

answer:
```
{"_auth":{"get":{"api_version":2}}}
```

#### 8.13.2.11 Get setup is admin

```
{"_auth":{"get":{"setup_is_admin":null}}}
```

answer:
```
{"_auth":{"get":{"setup_is_admin":true}}}
```

#### 8.13.2.12 Get/set setup account

```
{"_auth":{"get":{"setup_account":null}}}
```

answer:
```
{"_auth":{"get":{"setup_account":{"user":"setup","secret":"secrethash","access":"sarwx"}}}}
```

#### 8.13.2.13 Get/set custom storage

Implementation defined extra storage, stored in auth.json
```
{"_auth":{"get":{"storage":null}}}
```

answer:
```
{"_auth":{"get":{"storage":whateverjson...}}}
```

### 8.13.3 Get process' interface description

#### 8.13.3.1 Get XML decription

```
{"_vars_xml": null}
```

answer:
```
<?xml version="1.0"?>
<Process title="x1 .CLOUD" default_name="" preset_extension="clx1">
...
</Process>
```

**Attention**

The answer is XML, not JSON !

#### 8.13.3.2 Get Process Image

```
{"_image": null}
```

answer:
```
{"_image":"iVBORw0KGgoAA..." }
```

The string is the base64 encoding of the image (png format)

#### 8.13.3.3 Get Process Ico

```
{"_ico": null}
```

answer:
```
{"_ico":"AAABAAUAEBAAAAEAIABo..." }
```

The string is the base64 encoding of the icon (ico format)

## 8.14 Web server

### Macros

- #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
- #define SOUND4_WEBSERVER_HTTP_OK (1<<0)
- #define SOUND4_WEBSERVER_HTTPS_OK (1<<1)

### Functions

- uint64_t impact_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket↩_ip, unsigned int socket_port)
- uint64_t impact_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct impact_CParameters *parameters)
- uint64_t impact_Webserver (unsigned int listenport, unsigned int listenport_secure, struct impact_CInstance *instance)
- int impact_Webserver_Stop (uint64_t id, int timeout_ms)
- int impact_Webserver_Status (uint64_t id)

### 8.14.1 Detailed Description

Run a web server to configure the processor

See also how to setup authentication with Environment variables and parameters. Their use is documented in impact_InitProcess.

### 8.14.2 Macro Definition Documentation

#### 8.14.2.1 SOUND4_INVALID_WEBSERVER_ID

```
#define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
```

This represents an invalid webserver id impact_Webserver and impact_Webserver_tcp will never return this value

Definition at line 1541 of file sound4.impact.cl.h.

#### 8.14.2.2 SOUND4_WEBSERVER_HTTP_OK

```
#define SOUND4_WEBSERVER_HTTP_OK (1<<0)
```

If this bit is set in the return value from impact_Webserver_Status, then HTTP server is OK

Definition at line 1546 of file sound4.impact.cl.h.

#### 8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK

```
#define SOUND4_WEBSERVER_HTTPS_OK (1<<1)
```

If this bit is set in the return value from impact_Webserver_Status, then HTTPS server is OK

Definition at line 1550 of file sound4.impact.cl.h.

### 8.14.3 Function Documentation

#### 8.14.3.1 impact_Webserver()

```
uint64_t impact_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct impact_CInstance * instance )
```

Run a web server, connecting to local instance internally (no expose port needed)

This function returns after connecting to the process, and continue to run in background See impact_Webserver_Stop too stop it gracefully

For secure connection, it will use privatekey.pem and certificate.pem from current directory. Environment variables `SSL_PRIVATE_PATH` and `SSL_CERTIFICATE_PATH` can override those (see Environment variables and parameters). If none is valid, it will try also /etc/ssl/certs/server.key and /etc/ssl/certs/server.crt.

**Warning**

On Windows, calling this from Go<1.20 code will crash.

**Parameters**

| | |
|---|---|
| *listenport* | The port to listen for HTTP connections (0 to disable) |
| *listenport_secure* | The port to listen for HTTPS connections (0 to disable) |
| *instance* | The process instance to connect to |

**Returns**

id of webserver (starting at 0), to use with impact_Webserver_Stop

#### 8.14.3.2 impact_Webserver_Status()

```
int impact_Webserver_Status (
            uint64_t id )
```

Get the web server status

**Warning**

On Windows, calling this from Go<1.20 code will crash.

**Parameters**

| | |
|---|---|
| *id* | The web server Id, returned by impact_Webserver_tcp or impact_Webserver |

**Returns**

a combinaison of SOUND4_WEBSERVER_HTTP_OK and SOUND4_WEBSERVER_HTTPS_OK depending on the server status

#### 8.14.3.3 impact_Webserver_Stop()

```
int impact_Webserver_Stop (
            uint64_t id,
            int timeout_ms )
```

Stop a web server started with impact_Webserver_tcp

**Warning**

On Windows, calling this from Go<1.20 code will crash.

**Parameters**

| | |
|---|---|
| *id* | The web server Id, returned by impact_Webserver_tcp or impact_Webserver |
| *timeout_ms* | The timeout to wait gracefull shutdown |

**Returns**

0 if succeed, -1 if the id is not found

#### 8.14.3.4 impact_Webserver_tcp()

```
uint64_t impact_Webserver_tcp (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port )
```

Run a web server, connecting to process with JSON over TCP

This function returns after connecting to the process, and continue to run in background See impact_Webserver_Stop too stop it gracefully

For secure connection, it will use privatekey.pem and certificate.pem from current directory. Environment variables `SSL_PRIVATE_PATH` and `SSL_CERTIFICATE_PATH` can override those (see Environment variables and parameters). If none is valid, it will try also /etc/ssl/certs/server.key and /etc/ssl/certs/server.crt.

**Warning**

On Windows, calling this from Go<1.20 code will crash.

**Parameters**

| | |
|---|---|
| *listenport* | The port to listen for HTTP connections |
| *listenport_secure* | The port to listen for HTTPS connections |
| *socket_ip* | The ip of the process to connect to |
| *socket_port* | The port of the process's JSON server to connect to |

**Returns**

id of webserver (starting at 0), to use with impact_Webserver_Stop

#### 8.14.3.5 impact_Webserver_tcp2()

```
uint64_t impact_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct impact_CParameters * parameters )
```

Run a web server, connecting to process with JSON over TCP, using parameters

This function returns after connecting to the process, and continue to run in background See impact_Webserver_Stop too stop it gracefully

For secure connection, it will use privatekey.pem and certificate.pem from current directory. Environment variables `SSL_PRIVATE_PATH` and `SSL_CERTIFICATE_PATH` can override those (see Environment variables and parameters). If none is valid, it will try also /etc/ssl/certs/server.key and /etc/ssl/certs/server.crt.

**Warning**

On Windows, calling this from Go<1.20 code will crash.

**Parameters**

| | |
|---|---|
| *listenport* | The port to listen for HTTP connections |
| *listenport_secure* | The port to listen for HTTPS connections |
| *socket_ip* | The ip of the process to connect to |
| *socket_port* | The port of the process's JSON server to connect to |
| *parameters* | The parameters to use, created with impact_NewParameters |

**Returns**

id of webserver (starting at 0), to use with impact_Webserver_Stop

## 8.15 Sample using C++ interface wrapper

See full example cpp/sound4.impact.cl-pipe-cxx.cpp

### 8.15.1 Setup and sanity checks

```
sound4::impact::SetLogSeverity(verbose5);
sound4::impact::SetLoggerCallback(MyLogger);
sound4::impact::CInstance instance;
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

### 8.15.2 Sample to create the processing instance

```
if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
    std::cerr « "Failed to create the processing instance" « std::endl;
    return 1;
}
```

### 8.15.3 Communicating with the instance

**See also**

JSON syntax

```
auto client = instance.NewClient();
auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
std::cerr « "Request for processor name returned "«answer«std::endl;
```

### 8.15.4 Starting a web-server

```
if (!instance.StartWebServer(listen_port,0)) {
    std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
    return 1;
}
```

### 8.15.5 Sample processing loop

```
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::impact::InputSampleSize, stdin)
   != sound4::impact::InputSampleSize) {
        std::cerr « "ProcessLoop: input stopped"« std::endl;
        stopthread=true;
        break;
    }
    if (stopthread) break;
    // Process it
    instance.ProcessAudio();
    // Write it to stdout
    if (fwrite(instance.GetBufferOut().data(), sizeof(float), sound4::impact::OutputSampleSize, stdout)
   != sound4::impact::OutputSampleSize) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

## 8.16 Sample using C++ interface wrapper with dynamic library loading

See full example cpp/sound4.impact.cl-pipe-dyn.cpp

### 8.16.1 Dynamic loading

```
sound4::impact::dyn::CDynLib dynlib;
if (!dynlib.Load()) {
    std::cerr « "Failed to load processing library" « std::endl;
    return 1;
}
```

### 8.16.2 Setup and sanity checks

```
sound4::impact::dyn::SetLogSeverity(dynlib, verbose5);
sound4::impact::dyn::SetLoggerCallback(dynlib, MyLogger);
sound4::impact::dyn::CInstance instance(dynlib);
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

### 8.16.3 Sample to create the processing instance

```
if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
    std::cerr « "Failed to create the processing instance" « std::endl;
    return 1;
}
```

### 8.16.4 Communicating with the instance

#### See also

JSON syntax

```
auto client = instance.NewClient();
auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
std::cerr « "Request for processor name returned "«answer«std::endl;
```

### 8.16.5 Starting a web-server

```
if (!instance.StartWebServer(listen_port,0)) {
    std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
    return 1;
}
```

### 8.16.6 Sample processing loop

```
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::impact::dyn::InputSampleSize,
   stdin) != sound4::impact::dyn::InputSampleSize) {
        std::cerr « "ProcessLoop: input stopped"« std::endl;
        stopthread=true;
        break;
    }
    if (stopthread) break;
    // Process it
    instance.ProcessAudio();
    // Write it to stdout
    if (fwrite(instance.GetBufferOut().data(), sizeof(float), sound4::impact::dyn::OutputSampleSize,
   stdout) != sound4::impact::dyn::OutputSampleSize) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

### 8.16.7 Custom preset manager

```
class MyPresetManager : public sound4::impact::dyn::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;
    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
       std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(entry.path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
            if ( f.fail() ) {
                return false;
            }
            return true;
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};
```

### 8.16.8 Custom preset manager loading

```
    MyPresetManager preset_manager;
    if (!store_path.empty()) {
        if (!preset_manager.SetupPath(store_path)) {
            std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
       std::endl;
            return 1;
```

```
    }
    instance.SetPresetManager(&preset_manager);
}
```

## 8.17 Sample using Generic C++ interface (multi-process compatible)

This wrapper abstracts the processing specifics, working with any SOUND4 .CL process.

It uses sound4cl.hpp header.

See full example cpp/sound4cl-pipe.cpp

### 8.17.1 Loading the library

```
std::string libname =  "libsound4.impact.cl.so";
sound4::CProcessor processor;
if (!processor.Load(libname)) {
    std::cerr « "Failed to load processing library" « std::endl;
    return 1;
}
```

### 8.17.2 Setup and configure

```
processor.SetLogSeverity(sound4::verbose5);
processor.SetLoggerCallback([](sound4::LogSeverity severity,const std::string& msg) {
        std::cerr « msg « std::endl;
    });
sound4::CInstance instance(processor);
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

### 8.17.3 Create the processing instance

```
if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
    std::cerr « "Failed to create the processing instance" « std::endl;
    return 1;
}
```

### 8.17.4 Processing loop

```
size_t input_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
size_t output_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn(), sizeof(float), input_sample_count, stdin) !=
   input_sample_count) {
        std::cerr « "ProcessLoop: input stopped"« std::endl;
        stopthread=true;
        break;
    }
    if (stopthread) break;
    // Process it
    instance.ProcessAudio();
    // Write it to stdout
    if (fwrite(instance.GetBufferOut(), sizeof(float), output_sample_count, stdout) !=
   output_sample_count) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

### 8.17.5 Starting a web-server

```
if (!instance.StartWebServer(listen_port,0)) {
    std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
    return 1;
}
```

### 8.17.6 Communicating with the instance

#### See also

JSON syntax

```
auto client = instance.NewClient();
auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
std::cerr « "Request for processor name returned "«answer«std::endl;
```

### 8.17.7 Custom preset manager example

```
class MyPresetManager : public sound4::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;
    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
       std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(entry.path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
```

```
            if ( f.fail() ) {
                return false;
            }
            return true;
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};
```

### 8.17.8 Custom preset manager loading

```
    MyPresetManager preset_manager;
    if (!store_path.empty()) {
        if (!preset_manager.SetupPath(store_path)) {
            std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
       std::endl;
            return 1;
        }
        instance.SetPresetManager(&preset_manager);
    }
```

# Chapter 9

# Namespace Documentation

## 9.1 dyn Namespace Reference

### 9.1.1 Detailed Description

Dynamic loading namespace

## 9.2 sound4 Namespace Reference

### Namespaces

- helper
- impact

### Classes

- class CProcessor

  *Dynamic library interface.*
- class CBus

  *Shared Bus.*
- class CPresetLoader

  *Custom preset handler helper.*
- class CInstance

  *Instance handling class.*

### Typedefs

- using LogSeverity = sound4cl_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = sound4cl_PresetChange_Kind

## Variables

- const LogSeverity none = (LogSeverity) sound4cl_none
- const LogSeverity fatal = (LogSeverity) sound4cl_fatal
- const LogSeverity error = (LogSeverity) sound4cl_error
- const LogSeverity warning = (LogSeverity) sound4cl_warning
- const LogSeverity info = (LogSeverity) sound4cl_info
- const LogSeverity verbose = (LogSeverity) sound4cl_verbose
- const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2
- const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3
- const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4
- const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5

### 9.2.1 Detailed Description

SOUND4 namespace

### 9.2.2 Typedef Documentation

#### 9.2.2.1 log_cb_t

```
using sound4::log_cb_t = typedef std::function<void(LogSeverity,const std::string&)>
```

Log callback type

Definition at line 252 of file sound4cl.hpp.

#### 9.2.2.2 LogSeverity

```
using sound4::LogSeverity = typedef sound4cl_LogSeverity
```

Log Severity levels

Definition at line 237 of file sound4cl.hpp.

#### 9.2.2.3 PresetChange_Kind

```
using sound4::PresetChange_Kind = typedef sound4cl_PresetChange_Kind
```

Definition at line 728 of file sound4cl.hpp.

### 9.2.3 Variable Documentation

#### 9.2.3.1 error

```
const LogSeverity sound4::error = (LogSeverity) sound4cl_error
```

Definition at line 240 of file sound4cl.hpp.

#### 9.2.3.2 fatal

```
const LogSeverity sound4::fatal = (LogSeverity) sound4cl_fatal
```

Definition at line 239 of file sound4cl.hpp.

#### 9.2.3.3 info

```
const LogSeverity sound4::info = (LogSeverity) sound4cl_info
```

Definition at line 242 of file sound4cl.hpp.

#### 9.2.3.4 none

```
const LogSeverity sound4::none = (LogSeverity) sound4cl_none
```

Definition at line 238 of file sound4cl.hpp.

#### 9.2.3.5 verbose

```
const LogSeverity sound4::verbose = (LogSeverity) sound4cl_verbose
```

Definition at line 243 of file sound4cl.hpp.

#### 9.2.3.6 verbose2

```
const LogSeverity sound4::verbose2 = (LogSeverity) sound4cl_verbose2
```

Definition at line 244 of file sound4cl.hpp.

#### 9.2.3.7 verbose3

```
const LogSeverity sound4::verbose3 = (LogSeverity) sound4cl_verbose3
```

Definition at line 245 of file sound4cl.hpp.

#### 9.2.3.8 verbose4

```
const LogSeverity sound4::verbose4 = (LogSeverity) sound4cl_verbose4
```

Definition at line 246 of file sound4cl.hpp.

#### 9.2.3.9 verbose5

```
const LogSeverity sound4::verbose5 = (LogSeverity) sound4cl_verbose5
```

Definition at line 247 of file sound4cl.hpp.

#### 9.2.3.10 warning

```
const LogSeverity sound4::warning = (LogSeverity) sound4cl_warning
```

Definition at line 241 of file sound4cl.hpp.

## 9.3 sound4::helper Namespace Reference

### Classes

- struct SampleFormat

  *Helper for Sample format types.*
- struct SampleFormat< int16_t >
- struct SampleFormat< int32_t >
- struct SampleFormat< float >
- class DynFuncHolder

  *Helper to wrap a function pointer.*
- class CDynLoader

  *Helper to load dynamic library.*

### Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

### 9.3.1 Detailed Description

helper functions for unicode strings in Windows

### 9.3.2 Typedef Documentation

#### 9.3.2.1 DynFunc_t

```
using sound4::helper::DynFunc_t = typedef FARPROC
```

Definition at line 106 of file sound4cl.hpp.

#### 9.3.2.2 DynLib_t

```
using sound4::helper::DynLib_t = typedef HMODULE
```

Definition at line 105 of file sound4cl.hpp.

## 9.4 sound4::impact Namespace Reference

### Namespaces

- dyn
- helper

### Classes

- class CPresetLoader
- class CInstance

### Typedefs

- using SampleFormat = impact_SampleFormat
- using LogSeverity = impact_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = impact_PresetChange_Kind

## Variables

- const size_t ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t InputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_INPUT_COUNT
- const size_t OutputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_OUTPUT_COUNT
- const size_t InputSampleSize =ChunkFrames∗InputChannels
- const size_t OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t SampleRate =IMPACT_SAMPLE_RATE

### 9.4.1 Detailed Description

impact namespace

### 9.4.2 Typedef Documentation

#### 9.4.2.1 log_cb_t

```
using sound4::impact::log_cb_t = typedef std::function<void(LogSeverity,const std::string&)>
```

Log callback type

Definition at line 320 of file sound4.impact.cl.hpp.

#### 9.4.2.2 LogSeverity

```
using sound4::impact::LogSeverity = typedef impact_LogSeverity
```

Log Severity levels

Definition at line 316 of file sound4.impact.cl.hpp.

#### 9.4.2.3 PresetChange_Kind

```
using sound4::impact::PresetChange_Kind = typedef impact_PresetChange_Kind
```

Definition at line 380 of file sound4.impact.cl.hpp.

#### 9.4.2.4 SampleFormat

```
using sound4::impact::SampleFormat = typedef impact_SampleFormat
```

Definition at line 53 of file sound4.impact.cl.hpp.

### 9.4.3 Variable Documentation

#### 9.4.3.1 ChunkFrames

```
const size_t sound4::impact::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
```

Number of frames in a chunk

Definition at line 57 of file sound4.impact.cl.hpp.

#### 9.4.3.2 InputChannels

```
const size_t sound4::impact::InputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT
```

Number of float values in an input frame

Definition at line 61 of file sound4.impact.cl.hpp.

#### 9.4.3.3 InputSampleSize

```
const size_t sound4::impact::InputSampleSize =ChunkFrames*InputChannels
```

Number of float values for an input chunk

Definition at line 69 of file sound4.impact.cl.hpp.

#### 9.4.3.4 OutputChannels

```
const size_t sound4::impact::OutputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT
```

Number of float values in an output frame

Definition at line 65 of file sound4.impact.cl.hpp.

#### 9.4.3.5 OutputSampleSize

```
const size_t sound4::impact::OutputSampleSize =ChunkFrames*OutputChannels
```

Number of float values for an output chunk

Definition at line 73 of file sound4.impact.cl.hpp.

#### 9.4.3.6 SampleRate

```
const size_t sound4::impact::SampleRate =IMPACT_SAMPLE_RATE
```

Sample rate of library

Definition at line 77 of file sound4.impact.cl.hpp.

## 9.5 sound4::impact::dyn Namespace Reference

### Namespaces

- helper

### Classes

- class CDynLib

  *Dynamic library interface.*
- class CPresetLoader
- class CInstance

### Typedefs

- using SampleFormat = impact_SampleFormat
- using LogSeverity = impact_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = impact_PresetChange_Kind

### Variables

- const size_t ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t InputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT
- const size_t OutputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT
- const size_t InputSampleSize =ChunkFrames*InputChannels
- const size_t OutputSampleSize =ChunkFrames*OutputChannels
- const size_t SampleRate =IMPACT_SAMPLE_RATE

## 9.5.1 Typedef Documentation

### 9.5.1.1 log_cb_t

```
using sound4::impact::dyn::log_cb_t = typedef std::function<void(LogSeverity,const std::string&)>
```

Log callback type

Definition at line 717 of file sound4.impact.cl_dyn.hpp.

### 9.5.1.2 LogSeverity

```
using sound4::impact::dyn::LogSeverity = typedef impact_LogSeverity
```

Log Severity levels

Definition at line 713 of file sound4.impact.cl_dyn.hpp.

### 9.5.1.3 PresetChange_Kind

```
using sound4::impact::dyn::PresetChange_Kind = typedef impact_PresetChange_Kind
```

Definition at line 785 of file sound4.impact.cl_dyn.hpp.

### 9.5.1.4 SampleFormat

```
using sound4::impact::dyn::SampleFormat = typedef impact_SampleFormat
```

Definition at line 446 of file sound4.impact.cl_dyn.hpp.

## 9.5.2 Variable Documentation

### 9.5.2.1 ChunkFrames

```
const size_t sound4::impact::dyn::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
```

Number of frames in a chunk

Definition at line 450 of file sound4.impact.cl_dyn.hpp.

#### 9.5.2.2 InputChannels

```
const size_t sound4::impact::dyn::InputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT
```

Number of float values in an input frame

Definition at line 454 of file sound4.impact.cl_dyn.hpp.

#### 9.5.2.3 InputSampleSize

```
const size_t sound4::impact::dyn::InputSampleSize =ChunkFrames*InputChannels
```

Number of float values for an input chunk

Definition at line 462 of file sound4.impact.cl_dyn.hpp.

#### 9.5.2.4 OutputChannels

```
const size_t sound4::impact::dyn::OutputChannels =IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT
```

Number of float values in an output frame

Definition at line 458 of file sound4.impact.cl_dyn.hpp.

#### 9.5.2.5 OutputSampleSize

```
const size_t sound4::impact::dyn::OutputSampleSize =ChunkFrames*OutputChannels
```

Number of float values for an output chunk

Definition at line 466 of file sound4.impact.cl_dyn.hpp.

#### 9.5.2.6 SampleRate

```
const size_t sound4::impact::dyn::SampleRate =IMPACT_SAMPLE_RATE
```

Sample rate of library

Definition at line 470 of file sound4.impact.cl_dyn.hpp.

## 9.6 sound4::impact::dyn::helper Namespace Reference

### Classes

- class DynFuncHolder

  *Helper to wrap a function pointer.*
- class CDynLoader

  *Helper to load dynamic library.*
- struct SampleFormat

  *Helper for Sample format types.*
- struct SampleFormat< int16_t >
- struct SampleFormat< int32_t >
- struct SampleFormat< float >

### Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

### 9.6.1 Detailed Description

helper functions for unicode strings in Windows

### 9.6.2 Typedef Documentation

#### 9.6.2.1 DynFunc_t

```
using sound4::impact::dyn::helper::DynFunc_t = typedef FARPROC
```

Definition at line 77 of file sound4.impact.cl_dyn.hpp.

#### 9.6.2.2 DynLib_t

```
using sound4::impact::dyn::helper::DynLib_t = typedef HMODULE
```

Definition at line 76 of file sound4.impact.cl_dyn.hpp.

## 9.7 sound4::impact::helper Namespace Reference

### Classes

- struct SampleFormat

  *Helper for Sample format types.*
- struct SampleFormat< int16_t >
- struct SampleFormat< int32_t >
- struct SampleFormat< float >

### 9.7.1 Detailed Description

helper functions for unicode strings in Windows

## 9.8 std Namespace Reference

# Chapter 10

# Class Documentation

## 10.1 sound4::CBus Class Reference

Shared Bus.

```
#include <sound4cl.hpp>
```

### Public Member Functions

- CBus (CProcessor &dynlib)
- ~CBus ()
- sound4cl_CBus * Get () const

### 10.1.1 Detailed Description

Some processor can share a bus for realtime data exchange

Definition at line 688 of file sound4cl.hpp.

### 10.1.2 Constructor & Destructor Documentation

#### 10.1.2.1 CBus()

```
sound4::CBus::CBus (
            CProcessor & dynlib )  [inline]
```

Definition at line 690 of file sound4cl.hpp.

#### 10.1.2.2 ~CBus()

```
sound4::CBus::~CBus ( ) [inline]
```

Definition at line 697 of file sound4cl.hpp.

### 10.1.3 Member Function Documentation

#### 10.1.3.1 Get()

```
sound4cl_CBus* sound4::CBus::Get ( ) const [inline]
```

Definition at line 704 of file sound4cl.hpp.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.2 sound4::CInstance::CClient Class Reference

```
#include <sound4cl.hpp>
```

### Public Member Functions

- CClient (sound4cl_CInstance *instance, CProcessor &dynlib)
- CClient (const CInstance &)=delete
- CClient & operator= (const CInstance &)=delete
- ~CClient ()
- std::string ProcessJson (const std::string &request, bool *NeedSave=nullptr)

  *Process a JSON request and returns the answer.*

### 10.2.1 Detailed Description

Low-level JSON client.

**See also**

JSON syntax

Definition at line 1176 of file sound4cl.hpp.

### 10.2.2 Constructor & Destructor Documentation

#### 10.2.2.1 CClient() [1/2]

```
sound4::CInstance::CClient::CClient (
            sound4cl_CInstance * instance,
            CProcessor & dynlib )  [inline]
```

Definition at line 1178 of file sound4cl.hpp.

#### 10.2.2.2 CClient() [2/2]

```
sound4::CInstance::CClient::CClient (
            const CInstance &  )  [delete]
```

#### 10.2.2.3 ~CClient()

```
sound4::CInstance::CClient::~CClient ( )  [inline]
```

Definition at line 1188 of file sound4cl.hpp.

### 10.2.3 Member Function Documentation

#### 10.2.3.1 operator=()

```
CClient& sound4::CInstance::CClient::operator= (
            const CInstance &  )  [delete]
```

#### 10.2.3.2 ProcessJson()

```
std::string sound4::CInstance::CClient::ProcessJson (
            const std::string & request,
            bool * NeedSave = nullptr )  [inline]
```

**Parameters**

| | |
|---|---|
| *request* | the JSON request. |

**See also**

JSON syntax

**Parameters**

| | |
|---|---|
| *NeedSave* | if non null, returns if the saving is required (maybe delayed) |

**Returns**

the JSON answer

Definition at line 1202 of file sound4cl.hpp.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.3 sound4::impact::CInstance::CClient Class Reference

```
#include <sound4.impact.cl.hpp>
```

### Public Member Functions

- CClient (impact_CInstance *instance)
- CClient (const CInstance &)=delete
- CClient & operator= (const CInstance &)=delete
- ~CClient ()
- std::string ProcessJson (const std::string &request, bool *NeedSave=nullptr)

  *Process a JSON request and returns the answer.*

### 10.3.1 Detailed Description

Low-level JSON client.

**See also**

JSON syntax

Definition at line 804 of file sound4.impact.cl.hpp.

## 10.3.2 Constructor & Destructor Documentation

### 10.3.2.1 CClient() [1/2]

```
sound4::impact::CInstance::CClient::CClient (
            impact_CInstance * instance )  [inline]
```

Definition at line 806 of file sound4.impact.cl.hpp.

### 10.3.2.2 CClient() [2/2]

```
sound4::impact::CInstance::CClient::CClient (
            const CInstance &  )  [delete]
```

### 10.3.2.3 ~CClient()

```
sound4::impact::CInstance::CClient::~CClient ( )  [inline]
```

Definition at line 815 of file sound4.impact.cl.hpp.

## 10.3.3 Member Function Documentation

### 10.3.3.1 operator=()

```
CClient& sound4::impact::CInstance::CClient::operator= (
            const CInstance &  )  [delete]
```

### 10.3.3.2 ProcessJson()

```
std::string sound4::impact::CInstance::CClient::ProcessJson (
            const std::string & request,
            bool * NeedSave = nullptr )  [inline]
```

**Parameters**

| | |
|---|---|
| *request* | the JSON request. |

**See also**

JSON syntax

**Parameters**

| | |
|---|---|
| *NeedSave* | if non null, returns if the saving is required (maybe delayed) |

**Returns**

the JSON answer

Definition at line 829 of file sound4.impact.cl.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl.hpp

## 10.4 sound4::impact::dyn::CInstance::CClient Class Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Member Functions

- CClient (impact_CInstance *instance, CDynLib &dynlib)
- CClient (const CInstance &)=delete
- CClient & operator= (const CInstance &)=delete
- ~CClient ()
- std::string ProcessJson (const std::string &request, bool *NeedSave=nullptr)

  *Process a JSON request and returns the answer.*

### 10.4.1 Detailed Description

Low-level JSON client.

**See also**

JSON syntax

Definition at line 1231 of file sound4.impact.cl_dyn.hpp.

### 10.4.2 Constructor & Destructor Documentation

#### 10.4.2.1 CClient() [1/2]

```
sound4::impact::dyn::CInstance::CClient::CClient (
            impact_CInstance * instance,
            CDynLib & dynlib )  [inline]
```

Definition at line 1233 of file sound4.impact.cl_dyn.hpp.

#### 10.4.2.2 CClient() [2/2]

```
sound4::impact::dyn::CInstance::CClient::CClient (
            const CInstance &  )  [delete]
```

#### 10.4.2.3 ~CClient()

```
sound4::impact::dyn::CInstance::CClient::~CClient ( )  [inline]
```

Definition at line 1243 of file sound4.impact.cl_dyn.hpp.

### 10.4.3 Member Function Documentation

#### 10.4.3.1 operator=()

```
CClient& sound4::impact::dyn::CInstance::CClient::operator= (
            const CInstance &  )  [delete]
```

#### 10.4.3.2 ProcessJson()

```
std::string sound4::impact::dyn::CInstance::CClient::ProcessJson (
            const std::string & request,
            bool * NeedSave = nullptr )  [inline]
```

**Parameters**

| | |
|---|---|
| *request* | the JSON request. |

**See also**

JSON syntax

**Parameters**

| | |
|---|---|
| *NeedSave* | if non null, returns if the saving is required (maybe delayed) |

**Returns**

the JSON answer

Definition at line 1257 of file sound4.impact.cl_dyn.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.5 sound4::impact::dyn::CDynLib Class Reference

Dynamic library interface.

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Member Functions

- CDynLib ()=default
- bool Load (const std::filesystem::path &path={})

  *Loads the library.*
- bool IsOk () const

  *Check if the library was loaded correctly.*

### Public Attributes

- helper::DynFuncHolder< decltype(impact_GetVersion) > GetVersion
- helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) > GetChunkSizeInFrames
- helper::DynFuncHolder< decltype(impact_GetChannelCount) > GetChannelCount
- helper::DynFuncHolder< decltype(impact_GetAudioInputCount) > GetAudioInputCount
- helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) > GetAudioOutputCount
- helper::DynFuncHolder< decltype(impact_GetSampleRate) > GetSampleRate
- helper::DynFuncHolder< decltype(impact_SetLoggerCallback) > SetLoggerCallback
- helper::DynFuncHolder< decltype(impact_SetLogSeverity) > SetLogSeverity
- helper::DynFuncHolder< decltype(impact_NewParameters) > NewParameters
- helper::DynFuncHolder< decltype(impact_FreeParameters) > FreeParameters
- helper::DynFuncHolder< decltype(impact_SetParameter) > SetParameter
- helper::DynFuncHolder< decltype(impact_GetParameter) > GetParameter
- helper::DynFuncHolder< decltype(impact_FreeParameterValue) > FreeParameterValue
- helper::DynFuncHolder< decltype(impact_InitProcess) > InitProcess
- helper::DynFuncHolder< decltype(impact_InitProcess2) > InitProcess2

- helper::DynFuncHolder< decltype(impact_TerminateProcess) > TerminateProcess
- helper::DynFuncHolder< decltype(impact_ExitProcess) > ExitProcess
- helper::DynFuncHolder< decltype(impact_StartUpdateThread) > StartUpdateThread
- helper::DynFuncHolder< decltype(impact_StopUpdateThread) > StopUpdateThread
- helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) > WaitUpdateThreadReady
- helper::DynFuncHolder< decltype(impact_ProcessAudio) > ProcessAudio
- helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) > ProcessAudio_Planar
- helper::DynFuncHolder< decltype(impact_GetBufferIn) > GetBufferIn
- helper::DynFuncHolder< decltype(impact_GetBufferOut) > GetBufferOut
- helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) > GetEstimatedDelay
- helper::DynFuncHolder< decltype(impact_GetFormatName) > GetFormatName
- helper::DynFuncHolder< decltype(impact_GetFormatFromName) > GetFormatFromName
- helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) > GetBytesFromFormat
- helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) > GetMaxPacketFrame
- helper::DynFuncHolder< decltype(impact_AddAudio) > AddAudio
- helper::DynFuncHolder< decltype(impact_AddPadAudio) > AddPadAudio
- helper::DynFuncHolder< decltype(impact_GetOutputCount) > GetOutputCount
- helper::DynFuncHolder< decltype(impact_GetAudio) > GetAudio
- helper::DynFuncHolder< decltype(impact_AudioConvertFrom) > AudioConvertFrom
- helper::DynFuncHolder< decltype(impact_AudioConvertTo) > AudioConvertTo
- helper::DynFuncHolder< decltype(impact_StereoToMono) > StereoToMono
- helper::DynFuncHolder< decltype(impact_MonoToStereo) > MonoToStereo
- helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo
- helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) > AudioMonoToLiveStereo
- helper::DynFuncHolder< decltype(impact_NewClient) > NewClient
- helper::DynFuncHolder< decltype(impact_DeleteClient) > DeleteClient
- helper::DynFuncHolder< decltype(impact_ProcessJson) > ProcessJson
- helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) > FreeJsonAnswer
- helper::DynFuncHolder< decltype(impact_SaveState) > SaveState
- helper::DynFuncHolder< decltype(impact_Webserver_tcp) > Webserver_tcp
- helper::DynFuncHolder< decltype(impact_Webserver_tcp2) > Webserver_tcp2
- helper::DynFuncHolder< decltype(impact_Webserver) > Webserver
- helper::DynFuncHolder< decltype(impact_Webserver_Stop) > Webserver_Stop
- helper::DynFuncHolder< decltype(impact_Webserver_Status) > Webserver_Status
- helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) > StereoToMono_Planar
- helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) > MonoToStereo_Planar
- helper::DynFuncHolder< decltype(impact_SetMetadata) > SetMetadata
- helper::DynFuncHolder< decltype(impact_GetMetadataInfos) > GetMetadataInfos
- helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) > FreeMetadataInfos
- helper::DynFuncHolder< decltype(impact_SetPresetManager) > SetPresetManager
- helper::DynFuncHolder< decltype(impact_PresetManager_InformChange) > PresetManager_InformChange
- helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames) > GetPossibleChunkSizeInFrames
- helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) > GetProcessChunkFrames
- helper::DynFuncHolder< decltype(impact_InitProcess3) > InitProcess3
- helper::DynFuncHolder< decltype(impact_SetMetadataMulti) > SetMetadataMulti

### 10.5.1 Detailed Description

Once you loaded the library with Load(), if it succeed you can use all the standard C functions from this library.

Definition at line 210 of file sound4.impact.cl_dyn.hpp.

## 10.5.2 Constructor & Destructor Documentation

### 10.5.2.1 CDynLib()

```
sound4::impact::dyn::CDynLib::CDynLib ( ) [default]
```

## 10.5.3 Member Function Documentation

### 10.5.3.1 IsOk()

```
bool sound4::impact::dyn::CDynLib::IsOk ( ) const [inline]
```

**Returns**

true if OK.

false if not OK.

Definition at line 436 of file sound4.impact.cl_dyn.hpp.

### 10.5.3.2 Load()

```
bool sound4::impact::dyn::CDynLib::Load (
            const std::filesystem::path & path = {} ) [inline]
```

**Parameters**

| | |
|---|---|
| *path* | the path of the library to load. |

**Returns**

true if library was loaded correctly.

false if library load failed. Do not use in this case !

Definition at line 305 of file sound4.impact.cl_dyn.hpp.

## 10.5.4 Member Data Documentation

#### 10.5.4.1 AddAudio

```
helper::DynFuncHolder< decltype(impact_AddAudio ) > sound4::impact::dyn::CDynLib::AddAudio
```

Definition at line 247 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.2 AddPadAudio

```
helper::DynFuncHolder< decltype(impact_AddPadAudio ) > sound4::impact::dyn::CDynLib::AddPad↩
Audio
```

Definition at line 248 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.3 AudioConvertFrom

```
helper::DynFuncHolder< decltype(impact_AudioConvertFrom ) > sound4::impact::dyn::CDynLib::↩
AudioConvertFrom
```

Definition at line 251 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.4 AudioConvertTo

```
helper::DynFuncHolder< decltype(impact_AudioConvertTo ) > sound4::impact::dyn::CDynLib::↩
AudioConvertTo
```

Definition at line 252 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.5 AudioMonoFromLiveStereo

```
helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > sound4::impact::dyn::CDyn↩
Lib::AudioMonoFromLiveStereo
```

Definition at line 255 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.6 AudioMonoToLiveStereo

```
helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo ) > sound4::impact::dyn::CDyn↩
Lib::AudioMonoToLiveStereo
```

Definition at line 256 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.7 DeleteClient

```
helper::DynFuncHolder< decltype(impact_DeleteClient ) > sound4::impact::dyn::CDynLib::Delete↩
Client
```

Definition at line 258 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.8 ExitProcess

```
helper::DynFuncHolder< decltype(impact_ExitProcess ) > sound4::impact::dyn::CDynLib::Exit↩
Process
```

Definition at line 234 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.9 FreeJsonAnswer

```
helper::DynFuncHolder< decltype(impact_FreeJsonAnswer ) > sound4::impact::dyn::CDynLib::Free↩
JsonAnswer
```

Definition at line 260 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.10 FreeMetadataInfos

```
helper::DynFuncHolder< decltype(impact_FreeMetadataInfos ) > sound4::impact::dyn::CDynLib::↩
FreeMetadataInfos
```

Definition at line 277 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.11 FreeParameters

```
helper::DynFuncHolder< decltype(impact_FreeParameters ) > sound4::impact::dyn::CDynLib::Free↩
Parameters
```

Definition at line 227 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.12 FreeParameterValue

```
helper::DynFuncHolder< decltype(impact_FreeParameterValue ) > sound4::impact::dyn::CDynLib::↩
FreeParameterValue
```

Definition at line 230 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.13 GetAudio

```
helper::DynFuncHolder< decltype(impact_GetAudio ) > sound4::impact::dyn::CDynLib::GetAudio
```

Definition at line 250 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.14 GetAudioInputCount

```
helper::DynFuncHolder< decltype(impact_GetAudioInputCount ) > sound4::impact::dyn::CDynLib::↩
GetAudioInputCount
```

Definition at line 221 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.15 GetAudioOutputCount

```
helper::DynFuncHolder< decltype(impact_GetAudioOutputCount ) > sound4::impact::dyn::CDynLib↩
::GetAudioOutputCount
```

Definition at line 222 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.16 GetBufferIn

```
helper::DynFuncHolder< decltype(impact_GetBufferIn ) > sound4::impact::dyn::CDynLib::Get↩
BufferIn
```

Definition at line 240 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.17 GetBufferOut

```
helper::DynFuncHolder< decltype(impact_GetBufferOut ) > sound4::impact::dyn::CDynLib::Get↩
BufferOut
```

Definition at line 241 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.18 GetBytesFromFormat

```
helper::DynFuncHolder< decltype(impact_GetBytesFromFormat ) > sound4::impact::dyn::CDynLib::↩
GetBytesFromFormat
```

Definition at line 245 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.19 GetChannelCount

```
helper::DynFuncHolder< decltype(impact_GetChannelCount ) > sound4::impact::dyn::CDynLib::Get↩
ChannelCount
```

Definition at line 220 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.20 GetChunkSizeInFrames

```
helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames ) > sound4::impact::dyn::CDynLib↩
::GetChunkSizeInFrames
```

Definition at line 219 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.21 GetEstimatedDelay

```
helper::DynFuncHolder< decltype(impact_GetEstimatedDelay ) > sound4::impact::dyn::CDynLib::↩
GetEstimatedDelay
```

Definition at line 242 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.22 GetFormatFromName

```
helper::DynFuncHolder< decltype(impact_GetFormatFromName ) > sound4::impact::dyn::CDynLib::↩
GetFormatFromName
```

Definition at line 244 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.23 GetFormatName

```
helper::DynFuncHolder< decltype(impact_GetFormatName ) > sound4::impact::dyn::CDynLib::Get↩
FormatName
```

Definition at line 243 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.24 GetMaxPacketFrame

```
helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame ) > sound4::impact::dyn::CDynLib::↩
GetMaxPacketFrame
```

Definition at line 246 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.25 GetMetadataInfos

```
helper::DynFuncHolder< decltype(impact_GetMetadataInfos ) > sound4::impact::dyn::CDynLib::↩
GetMetadataInfos
```

Definition at line 276 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.26 GetOutputCount

```
helper::DynFuncHolder< decltype(impact_GetOutputCount ) > sound4::impact::dyn::CDynLib::Get↩
OutputCount
```

Definition at line 249 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.27 GetParameter

```
helper::DynFuncHolder< decltype(impact_GetParameter ) > sound4::impact::dyn::CDynLib::Get↩
Parameter
```

Definition at line 229 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.28 GetPossibleChunkSizeInFrames

```
helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames ) > sound4::impact::dyn↩
::CDynLib::GetPossibleChunkSizeInFrames
```

Definition at line 284 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.29 GetProcessChunkFrames

```
helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames ) > sound4::impact::dyn::CDyn↩
Lib::GetProcessChunkFrames
```

Definition at line 285 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.30 GetSampleRate

```
helper::DynFuncHolder< decltype(impact_GetSampleRate ) > sound4::impact::dyn::CDynLib::Get↩
SampleRate
```

Definition at line 223 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.31 GetVersion

```
helper::DynFuncHolder< decltype(impact_GetVersion ) > sound4::impact::dyn::CDynLib::GetVersion
```

Definition at line 218 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.32 InitProcess

```
helper::DynFuncHolder< decltype(impact_InitProcess ) > sound4::impact::dyn::CDynLib::Init↩
Process
```

Definition at line 231 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.33 InitProcess2

```
helper::DynFuncHolder< decltype(impact_InitProcess2 ) > sound4::impact::dyn::CDynLib::Init↩
Process2
```

Definition at line 232 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.34 InitProcess3

```
helper::DynFuncHolder< decltype(impact_InitProcess3 ) > sound4::impact::dyn::CDynLib::Init↩
Process3
```

Definition at line 286 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.35 MonoToStereo

```
helper::DynFuncHolder< decltype(impact_MonoToStereo ) > sound4::impact::dyn::CDynLib::MonoTo↩
Stereo
```

Definition at line 254 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.36 MonoToStereo_Planar

```
helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar ) > sound4::impact::dyn::CDynLib↩
::MonoToStereo_Planar
```

Definition at line 272 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.37 NewClient

```
helper::DynFuncHolder< decltype(impact_NewClient ) > sound4::impact::dyn::CDynLib::NewClient
```

Definition at line 257 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.38 NewParameters

```
helper::DynFuncHolder< decltype(impact_NewParameters ) > sound4::impact::dyn::CDynLib::New↩
Parameters
```

Definition at line 226 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.39 PresetManager_InformChange

```
helper::DynFuncHolder< decltype(impact_PresetManager_InformChange ) > sound4::impact::dyn::↩
CDynLib::PresetManager_InformChange
```

Definition at line 281 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.40 ProcessAudio

```
helper::DynFuncHolder< decltype(impact_ProcessAudio ) > sound4::impact::dyn::CDynLib::Process↩
Audio
```

Definition at line 238 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.41 ProcessAudio_Planar

```
helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar ) > sound4::impact::dyn::CDynLib↩
::ProcessAudio_Planar
```

Definition at line 239 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.42 ProcessJson

```
helper::DynFuncHolder< decltype(impact_ProcessJson ) > sound4::impact::dyn::CDynLib::Process↩
Json
```

Definition at line 259 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.43 SaveState

```
helper::DynFuncHolder< decltype(impact_SaveState ) > sound4::impact::dyn::CDynLib::SaveState
```

Definition at line 261 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.44 SetLoggerCallback

```
helper::DynFuncHolder< decltype(impact_SetLoggerCallback ) > sound4::impact::dyn::CDynLib::↩
SetLoggerCallback
```

Definition at line 224 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.45 SetLogSeverity

```
helper::DynFuncHolder< decltype(impact_SetLogSeverity ) > sound4::impact::dyn::CDynLib::Set↩
LogSeverity
```

Definition at line 225 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.46 SetMetadata

```
helper::DynFuncHolder< decltype(impact_SetMetadata ) > sound4::impact::dyn::CDynLib::Set↩
Metadata
```

Definition at line 275 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.47 SetMetadataMulti

```
helper::DynFuncHolder< decltype(impact_SetMetadataMulti ) > sound4::impact::dyn::CDynLib::↩
SetMetadataMulti
```

Definition at line 296 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.48 SetParameter

```
helper::DynFuncHolder< decltype(impact_SetParameter ) > sound4::impact::dyn::CDynLib::Set↩
Parameter
```

Definition at line 228 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.49 SetPresetManager

```
helper::DynFuncHolder< decltype(impact_SetPresetManager ) > sound4::impact::dyn::CDynLib::↩
SetPresetManager
```

Definition at line 280 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.50 StartUpdateThread

```
helper::DynFuncHolder< decltype(impact_StartUpdateThread ) > sound4::impact::dyn::CDynLib::↩
StartUpdateThread
```

Definition at line 235 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.51 StereoToMono

```
helper::DynFuncHolder< decltype(impact_StereoToMono ) > sound4::impact::dyn::CDynLib::Stereo↩
ToMono
```

Definition at line 253 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.52 StereoToMono_Planar

```
helper::DynFuncHolder< decltype(impact_StereoToMono_Planar ) > sound4::impact::dyn::CDynLib↩
::StereoToMono_Planar
```

Definition at line 271 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.53 StopUpdateThread

```
helper::DynFuncHolder< decltype(impact_StopUpdateThread ) > sound4::impact::dyn::CDynLib::↩
StopUpdateThread
```

Definition at line 236 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.54 TerminateProcess

```
helper::DynFuncHolder< decltype(impact_TerminateProcess ) > sound4::impact::dyn::CDynLib::↩
TerminateProcess
```

Definition at line 233 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.55 WaitUpdateThreadReady

```
helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady ) > sound4::impact::dyn::CDyn↩
Lib::WaitUpdateThreadReady
```

Definition at line 237 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.56 Webserver

```
helper::DynFuncHolder< decltype(impact_Webserver ) > sound4::impact::dyn::CDynLib::Webserver
```

Definition at line 265 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.57 Webserver_Status

```
helper::DynFuncHolder< decltype(impact_Webserver_Status ) > sound4::impact::dyn::CDynLib::↩
Webserver_Status
```

Definition at line 267 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.58 Webserver_Stop

```
helper::DynFuncHolder< decltype(impact_Webserver_Stop ) > sound4::impact::dyn::CDynLib::↩
Webserver_Stop
```

Definition at line 266 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.59 Webserver_tcp

```
helper::DynFuncHolder< decltype(impact_Webserver_tcp ) > sound4::impact::dyn::CDynLib::↩
Webserver_tcp
```

Definition at line 263 of file sound4.impact.cl_dyn.hpp.

#### 10.5.4.60 Webserver_tcp2

```
helper::DynFuncHolder< decltype(impact_Webserver_tcp2 ) > sound4::impact::dyn::CDynLib::↩
Webserver_tcp2
```

Definition at line 264 of file sound4.impact.cl_dyn.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.6 sound4::helper::CDynLoader Class Reference

Helper to load dynamic library.

```
#include <sound4cl.hpp>
```

### Public Member Functions

- CDynLoader ()=default
- CDynLoader (CDynLoader &&)=default
- CDynLoader (const CDynLoader &)=delete
- CDynLoader & operator= (CDynLoader const &)=delete
- operator bool () const
- bool IsOk () const
- bool Load (const std::filesystem::path &dynlib, uint32_t loadflags=LOAD_LIBRARY_SEARCH_DLL_LOAD↩ _DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_↩ LIBRARY_SEARCH_SYSTEM32)
- void Close ()
- template<typename T > DynFuncHolder< T > GetSymbol (const std::string &name)
- template<typename T > DynFuncHolder< T > GetSymbol_safe (const std::string &name)
- virtual ~CDynLoader ()

### Static Public Member Functions

- static std::filesystem::path GetThisLibraryPath (void)

### 10.6.1 Detailed Description

Definition at line 130 of file sound4cl.hpp.

### 10.6.2 Constructor & Destructor Documentation

#### 10.6.2.1 CDynLoader() [1/3]

```
sound4::helper::CDynLoader::CDynLoader ( ) [default]
```

#### 10.6.2.2 CDynLoader() [2/3]

```
sound4::helper::CDynLoader::CDynLoader (
            CDynLoader && ) [default]
```

#### 10.6.2.3 CDynLoader() [3/3]

```
sound4::helper::CDynLoader::CDynLoader (
            const CDynLoader & ) [delete]
```

#### 10.6.2.4 ~CDynLoader()

```
virtual sound4::helper::CDynLoader::~CDynLoader ( ) [inline], [virtual]
```

Definition at line 227 of file sound4cl.hpp.

### 10.6.3 Member Function Documentation

#### 10.6.3.1 Close()

```
void sound4::helper::CDynLoader::Close ( ) [inline]
```

Definition at line 188 of file sound4cl.hpp.

#### 10.6.3.2 GetSymbol()

```
template<typename T >
DynFuncHolder<T> sound4::helper::CDynLoader::GetSymbol (
            const std::string & name ) [inline]
```

Definition at line 201 of file sound4cl.hpp.

#### 10.6.3.3 GetSymbol_safe()

```
template<typename T >
DynFuncHolder<T> sound4::helper::CDynLoader::GetSymbol_safe (
            const std::string & name )  [inline]
```

Definition at line 209 of file sound4cl.hpp.

#### 10.6.3.4 GetThisLibraryPath()

```
static std::filesystem::path sound4::helper::CDynLoader::GetThisLibraryPath (
            void  )  [inline], [static]
```

Definition at line 213 of file sound4cl.hpp.

#### 10.6.3.5 IsOk()

```
bool sound4::helper::CDynLoader::IsOk ( ) const  [inline]
```

Definition at line 142 of file sound4cl.hpp.

#### 10.6.3.6 Load()

```
bool sound4::helper::CDynLoader::Load (
            const std::filesystem::path & dynlib,
            uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION↩
_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_SYSTEM32 )  [inline]
```

Definition at line 182 of file sound4cl.hpp.

#### 10.6.3.7 operator bool()

```
sound4::helper::CDynLoader::operator bool ( ) const  [inline]
```

Definition at line 139 of file sound4cl.hpp.

#### 10.6.3.8 operator=()

```
CDynLoader& sound4::helper::CDynLoader::operator= (
            CDynLoader const & ) [delete]
```

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.7 sound4::impact::dyn::helper::CDynLoader Class Reference

Helper to load dynamic library.

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Member Functions

- CDynLoader ()=default
- CDynLoader (CDynLoader &&)=default
- CDynLoader (const CDynLoader &)=delete
- CDynLoader & operator= (CDynLoader const &)=delete
- operator bool () const
- bool IsOk () const
- bool Load (const std::filesystem::path &dynlib, uint32_t loadflags=LOAD_LIBRARY_SEARCH_DLL_LOAD↩ _DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_↩ LIBRARY_SEARCH_SYSTEM32)
- void Close ()
- template<typename T > DynFuncHolder< T > GetSymbol (const std::string &name)
- template<typename T > DynFuncHolder< T > GetSymbol_safe (const std::string &name)
- virtual ~CDynLoader ()

### Static Public Member Functions

- static std::filesystem::path GetThisLibraryPath (void)

### 10.7.1 Detailed Description

Definition at line 101 of file sound4.impact.cl_dyn.hpp.

### 10.7.2 Constructor & Destructor Documentation

#### 10.7.2.1 CDynLoader() [1/3]

```
sound4::impact::dyn::helper::CDynLoader::CDynLoader ( ) [default]
```

#### 10.7.2.2 CDynLoader() [2/3]

```
sound4::impact::dyn::helper::CDynLoader::CDynLoader (
            CDynLoader && ) [default]
```

#### 10.7.2.3 CDynLoader() [3/3]

```
sound4::impact::dyn::helper::CDynLoader::CDynLoader (
            const CDynLoader & ) [delete]
```

#### 10.7.2.4 ~CDynLoader()

```
virtual sound4::impact::dyn::helper::CDynLoader::~CDynLoader ( ) [inline], [virtual]
```

Definition at line 198 of file sound4.impact.cl_dyn.hpp.

### 10.7.3 Member Function Documentation

#### 10.7.3.1 Close()

```
void sound4::impact::dyn::helper::CDynLoader::Close ( ) [inline]
```

Definition at line 159 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.2 GetSymbol()

```
template<typename T >
DynFuncHolder<T> sound4::impact::dyn::helper::CDynLoader::GetSymbol (
            const std::string & name ) [inline]
```

Definition at line 172 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.3 GetSymbol_safe()

```
template<typename T >
DynFuncHolder<T> sound4::impact::dyn::helper::CDynLoader::GetSymbol_safe (
            const std::string & name )  [inline]
```

Definition at line 180 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.4 GetThisLibraryPath()

```
static std::filesystem::path sound4::impact::dyn::helper::CDynLoader::GetThisLibraryPath (
            void  )  [inline], [static]
```

Definition at line 184 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.5 IsOk()

```
bool sound4::impact::dyn::helper::CDynLoader::IsOk ( ) const  [inline]
```

Definition at line 113 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.6 Load()

```
bool sound4::impact::dyn::helper::CDynLoader::Load (
            const std::filesystem::path & dynlib,
            uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION↩
_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_SYSTEM32 )  [inline]
```

Definition at line 153 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.7 operator bool()

```
sound4::impact::dyn::helper::CDynLoader::operator bool ( ) const  [inline]
```

Definition at line 110 of file sound4.impact.cl_dyn.hpp.

#### 10.7.3.8 operator=()

```
CDynLoader& sound4::impact::dyn::helper::CDynLoader::operator= (
            CDynLoader const & ) [delete]
```

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.8 sound4::CInstance Class Reference

Instance handling class.

```
#include <sound4cl.hpp>
```

### Classes

- class CClient

### Public Member Functions

- CInstance (CProcessor &dynlib)
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetBus (const CBus &bus)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &Access_Key_ID, const std::string &Access_Key_Secret, const std::filesystem::path &save_path, int json_port=0, unsigned int frames_per_chunk=0)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- void SetMetadataMulti (const std::unordered_map< std::string, const char * > &list)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- float * GetBufferIn ()
- float * GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

## Protected Member Functions

- template<typename T >
  unsigned int AddAudio (const T *payload, unsigned int nFrame)
- template<typename T >
  unsigned int GetAudio (T *payload, unsigned int max_nFrame)

## 10.8.1 Detailed Description

Holds an instance of processing

Definition at line 735 of file sound4cl.hpp.

## 10.8.2 Constructor & Destructor Documentation

### 10.8.2.1 CInstance() [1/2]

```
sound4::CInstance::CInstance (
            CProcessor & dynlib )  [inline]
```

Creates the instance, with default parameters

Definition at line 740 of file sound4cl.hpp.

### 10.8.2.2 CInstance() [2/2]

```
sound4::CInstance::CInstance (
            const CInstance &  )  [delete]
```

### 10.8.2.3 ~CInstance()

```
sound4::CInstance::~CInstance ( )  [inline]
```

Destructor, stopping any running tasks

Definition at line 752 of file sound4cl.hpp.

## 10.8.3 Member Function Documentation

#### 10.8.3.1 AddAudio()

```
template<typename T >
unsigned int sound4::CInstance::AddAudio (
            const T * payload,
            unsigned int nFrame )  [inline], [protected]
```

Definition at line 1234 of file sound4cl.hpp.

#### 10.8.3.2 Create()

```
bool sound4::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & Access_Key_ID,
            const std::string & Access_Key_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 0 )  [inline]
```

Creates and initialize the instance, and starts its update thread

**See also**

sound4cl_InitProcess2

**Parameters**

| | |
|---|---|
| *LoginKey* | SOUND4 Login key |
| *RadioName* | SOUND4 RadioName |
| *Access_Key_ID* | SOUND4 AWS access ID |
| *Access_Key_Secret* | SOUND4 AWS access Secret |
| *save_path* | Path where we can load/save the configuration |
| *json_port* | Optional json listening port |
| *frames_per_chunk* | The number of audio frames in each chunk of processing. Must be in the list returned by sound4cl_GetPossibleChunkSizeInFrames, or 0 to set default |

**Returns**

false if failed

Definition at line 839 of file sound4cl.hpp.

#### 10.8.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::CInstance::GetAudio (
```

```
T * payload,
unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 1240 of file sound4cl.hpp.

#### 10.8.3.4 GetBufferIn()

```
float* sound4::CInstance::GetBufferIn ( )  [inline]
```

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 1089 of file sound4cl.hpp.

#### 10.8.3.5 GetBufferOut()

```
float* sound4::CInstance::GetBufferOut ( )  [inline]
```

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 1105 of file sound4cl.hpp.

#### 10.8.3.6 GetChunkFrames()

```
unsigned int sound4::CInstance::GetChunkFrames ( )  [inline]
```

Get the number of audio frames in each chunk of processing.

The value reflects the frames_per_chunk parameter passed to Create

**Returns**

the number of audio frames in each chunk of processing

Definition at line 885 of file sound4cl.hpp.

#### 10.8.3.7 GetEstimatedDelay()

```
unsigned int sound4::CInstance::GetEstimatedDelay ( ) [inline]
```

Gets an estimated delay of current processing

**See also**

sound4cl_GetEstimatedDelay

Definition at line 1073 of file sound4cl.hpp.

#### 10.8.3.8 GetMetadataInfos()

```
std::vector< std::tuple<std::string,std::string> > sound4::CInstance::GetMetadataInfos ( )
[inline]
```

Get information on known metadata keys.

**Returns**

a list of {key,description} of known metadata used in process and extensions.

Definition at line 974 of file sound4cl.hpp.

#### 10.8.3.9 GetParam()

```
std::string sound4::CInstance::GetParam (
            const std::string & name ) [inline]
```

Gets a parameter value

**See also**

sound4cl_GetParameter

Definition at line 778 of file sound4cl.hpp.

#### 10.8.3.10 IsOk()

```
bool sound4::CInstance::IsOk ( ) const [inline]
```

Returns true if an instance has been created and not stopped.

Definition at line 760 of file sound4cl.hpp.

#### 10.8.3.11 NewClient()

```
std::shared_ptr<CClient> sound4::CInstance::NewClient ( ) [inline]
```

Creates a low-level JSON client

**See also**

JSON syntax

**Returns**

a new client to communicate with the processing

Definition at line 1226 of file sound4cl.hpp.

#### 10.8.3.12 OnUpdateThreadStart()

```
virtual void sound4::CInstance::OnUpdateThreadStart ( ) [inline], [virtual]
```

Definition at line 996 of file sound4cl.hpp.

#### 10.8.3.13 OnUpdateThreadStop()

```
virtual void sound4::CInstance::OnUpdateThreadStop ( ) [inline], [virtual]
```

Definition at line 998 of file sound4cl.hpp.

#### 10.8.3.14 operator=()

```
CInstance& sound4::CInstance::operator= (
            const CInstance & ) [delete]
```

#### 10.8.3.15 PresetManager_InformChange()

```
void sound4::CInstance::PresetManager_InformChange (
            const std::filesystem::path & relative_path,
            PresetChange_Kind change_kind ) [inline]
```

Inform that a change happened in the storage. This must be used only in conjonction with SetPresetManager

**Parameters**

| | |
|---|---|
| *relative_path* | The path which changed, relative to the root storage |
| *change_kind* | What happened to this path |

Definition at line 899 of file sound4cl.hpp.

#### 10.8.3.16 ProcessAnyAudio()

```
template<typename T >
std::vector<T> sound4::CInstance::ProcessAnyAudio (
            const std::vector< T > input )  [inline]
```

Process audio (different formats, less optimized)

Supports for int16_t, int32_t and float (32) in native endianness

**Attention**

the return count is independent from the input count

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

**Parameters**

| | |
|---|---|
| *input* | audio to process |

**Returns**

processed audio

Definition at line 1152 of file sound4cl.hpp.

#### 10.8.3.17 ProcessAudio()

```
void sound4::CInstance::ProcessAudio ( )  [inline]
```

Process one audio chunk, from GetBufferIn() to GetBufferOut()

**See also**

sound4cl_ProcessAudio

Definition at line 1115 of file sound4cl.hpp.

#### 10.8.3.18 ProcessAudio_Planar()

```
void sound4::CInstance::ProcessAudio_Planar (
            float const *const * input,
            float *const * output )  [inline]
```

Process one audio chunk, but with planar audio

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are NOT interlaced, so the buffers are arrays of planar audio : [ [L0,L1,...,Ln] , [R0,R1,...,Rn] ] (L=Left, R=Right)

In case of multiple audio input/output, buffer are [ [aL...], [aR...], [bL...], [bR...], ... ]

**See also**

sound4cl_ProcessAudio_Planar

**Parameters**

| | |
|---|---|
| *input* | has GetAudioInputCount() ∗ GetChannelCount() channels |
| *output* | has GetAudioOutputCount() ∗ GetChannelCount() channels |

Definition at line 1134 of file sound4cl.hpp.

#### 10.8.3.19 SetBus()

```
void sound4::CInstance::SetBus (
            const CBus & bus )  [inline]
```

Set the instance bus

Definition at line 790 of file sound4cl.hpp.

#### 10.8.3.20 SetMetadata()

```
void sound4::CInstance::SetMetadata (
            const std::string & key,
            const char * value )  [inline]
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible. Setting a null value erase the metadata, which is different from setting an empty string.

**Parameters**

| | |
|---|---|
| *key* | the metadata key If key contains a dot (like wm.KEY), it will be sent only to the matching plugin (ie wm here) without the prefix (ie KEY here) Use empty prefix for routing to main only. |
| *value* | the metadata value or nullptr to erase it. |

Definition at line 917 of file sound4cl.hpp.

#### 10.8.3.21 SetMetadataMulti()

```
void sound4::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

**See also**

CInstance::SetMetadata

**Parameters**

| | |
|---|---|
| *list* | a map of key/value pair. |

Definition at line 938 of file sound4cl.hpp.

#### 10.8.3.22 SetParam()

```
void sound4::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

**See also**

sound4cl_SetParameter

Definition at line 768 of file sound4cl.hpp.

#### 10.8.3.23 SetPresetManager()

```
void sound4::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

Sets a custom preset manager Optional, this overrides the `save_path` in Create

**See also**

sound4cl_SetPresetManager

Must be done before Create

**Parameters**

| | |
|---|---|
| *preset_manager* | The custom preset manager |

Definition at line 806 of file sound4cl.hpp.

#### 10.8.3.24 StartWebServer()

```
bool sound4::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

**See also**

sound4cl_Webserver

Definition at line 1005 of file sound4cl.hpp.

#### 10.8.3.25 Stop()

```
void sound4::CInstance::Stop ( )  [inline]
```

Stops the instance and its web server and its update thread.

**See also**

sound4cl_ExitProcess,

sound4cl_StopUpdateThread and

sound4cl_Webserver_Stop

Definition at line 1056 of file sound4cl.hpp.

#### 10.8.3.26 StopWebServer()

```
void sound4::CInstance::StopWebServer ( ) [inline]
```

Stops the Web server

**See also**

sound4cl_Webserver_Stop

Definition at line 1030 of file sound4cl.hpp.

#### 10.8.3.27 Terminate()

```
int sound4::CInstance::Terminate ( ) [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

**See also**

sound4cl_TerminateProcess

Definition at line 1045 of file sound4cl.hpp.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.9 sound4::impact::CInstance Class Reference

```
#include <sound4.impact.cl.hpp>
```

### Classes

- class CClient

## Public Member Functions

- CInstance ()
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &AWS_Access_ID, const std::string &AWS_Access_Secret, const std::filesystem::path &save_path, int json_port=0, unsigned int frames_per_chunk=64)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- void SetMetadataMulti (const std::unordered_map< std::string, const char * > &list)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

## Protected Member Functions

- template<typename T >
  unsigned int AddAudio (const T *payload, unsigned int nFrame)
- template<typename T >
  unsigned int GetAudio (T *payload, unsigned int max_nFrame)

### 10.9.1 Detailed Description

Instance handling class

Definition at line 385 of file sound4.impact.cl.hpp.

### 10.9.2 Constructor & Destructor Documentation

#### 10.9.2.1 CInstance() [1/2]

```
sound4::impact::CInstance::CInstance ( ) [inline]
```

Creates the instance, with default parameters

Definition at line 390 of file sound4.impact.cl.hpp.

#### 10.9.2.2 CInstance() [2/2]

```
sound4::impact::CInstance::CInstance (
            const CInstance & ) [delete]
```

#### 10.9.2.3 ~CInstance()

```
sound4::impact::CInstance::~CInstance ( ) [inline]
```

Destructor, stopping any running tasks

Definition at line 401 of file sound4.impact.cl.hpp.

### 10.9.3 Member Function Documentation

#### 10.9.3.1 AddAudio()

```
template<typename T >
unsigned int sound4::impact::CInstance::AddAudio (
            const T * payload,
            unsigned int nFrame ) [inline], [protected]
```

Definition at line 860 of file sound4.impact.cl.hpp.

#### 10.9.3.2 Create()

```
bool sound4::impact::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & AWS_Access_ID,
            const std::string & AWS_Access_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 ) [inline]
```

Creates and initialize the instance, and starts its update thread

**See also**

impact_InitProcess2

**Parameters**

| | |
|---|---|
| *LoginKey* | SOUND4 Login key |
| *RadioName* | SOUND4 RadioName |
| *AWS_Access_ID* | SOUND4 AWS access ID |
| *AWS_Access_Secret* | SOUND4 AWS access Secret |
| *save_path* | Path where we can load/save the configuration |
| *json_port* | Optional json listening port |
| *frames_per_chunk* | The number of audio frames in each chunk of processing. Must be in the list returned by impact_GetPossibleChunkSizeInFrames, or 0 to set default |

**Returns**

false if failed

Definition at line 487 of file sound4.impact.cl.hpp.

#### 10.9.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::impact::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 866 of file sound4.impact.cl.hpp.

#### 10.9.3.4 GetBufferIn()

```
std::array<float,InputSampleSize>& sound4::impact::CInstance::GetBufferIn ( )  [inline]
```

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 715 of file sound4.impact.cl.hpp.

#### 10.9.3.5 GetBufferOut()

```
std::array<float,OutputSampleSize>& sound4::impact::CInstance::GetBufferOut ( ) [inline]
```

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 732 of file sound4.impact.cl.hpp.

#### 10.9.3.6 GetChunkFrames()

```
unsigned int sound4::impact::CInstance::GetChunkFrames ( ) [inline]
```

Get the number of audio frames in each chunk of processing.

The value reflects the frames_per_chunk parameter passed to Create

**Returns**

the number of audio frames in each chunk of processing

Definition at line 524 of file sound4.impact.cl.hpp.

#### 10.9.3.7 GetEstimatedDelay()

```
unsigned int sound4::impact::CInstance::GetEstimatedDelay ( ) [inline]
```

Gets an estimated delay of current processing

**See also**

impact_GetEstimatedDelay

Definition at line 699 of file sound4.impact.cl.hpp.

#### 10.9.3.8 GetMetadataInfos()

```
std::vector< std::tuple<std::string,std::string> > sound4::impact::CInstance::GetMetadata↩
Infos ( ) [inline]
```

Get information on known metadata keys.

**Returns**

a list of {key,description} of known metadata used in process and extensions.

Definition at line 602 of file sound4.impact.cl.hpp.

#### 10.9.3.9 GetParam()

```
std::string sound4::impact::CInstance::GetParam (
            const std::string & name ) [inline]
```

Gets a parameter value

**See also**

impact_GetParameter

Definition at line 427 of file sound4.impact.cl.hpp.

#### 10.9.3.10 IsOk()

```
bool sound4::impact::CInstance::IsOk ( ) const [inline]
```

Returns true if an instance has been created and not stopped.

Definition at line 409 of file sound4.impact.cl.hpp.

#### 10.9.3.11 NewClient()

```
std::shared_ptr<CClient> sound4::impact::CInstance::NewClient ( ) [inline]
```

Creates a low-level JSON client

**See also**

JSON syntax

**Returns**

a new client to communicate with the processing

Definition at line 852 of file sound4.impact.cl.hpp.

#### 10.9.3.12 OnUpdateThreadStart()

```
virtual void sound4::impact::CInstance::OnUpdateThreadStart ( ) [inline], [virtual]
```

Definition at line 622 of file sound4.impact.cl.hpp.

#### 10.9.3.13 OnUpdateThreadStop()

```
virtual void sound4::impact::CInstance::OnUpdateThreadStop ( ) [inline], [virtual]
```

Definition at line 624 of file sound4.impact.cl.hpp.

#### 10.9.3.14 operator=()

```
CInstance& sound4::impact::CInstance::operator= (
            const CInstance & ) [delete]
```

#### 10.9.3.15 PresetManager_InformChange()

```
void sound4::impact::CInstance::PresetManager_InformChange (
            const std::filesystem::path & relative_path,
            PresetChange_Kind change_kind ) [inline]
```

Inform that a change happened in the storage. This must be used only in conjonction with SetPresetManager

**Parameters**

| | |
|---|---|
| *relative_path* | The path which changed, relative to the root storage |
| *change_kind* | What happened to this path |

Definition at line 538 of file sound4.impact.cl.hpp.

#### 10.9.3.16 ProcessAnyAudio()

```
template<typename T >
std::vector<T> sound4::impact::CInstance::ProcessAnyAudio (
            const std::vector< T > input ) [inline]
```

Process audio (different formats, less optimized)

Supports for int16_t, int32_t and float (32) in native endianness

**Attention**

the return count is independent from the input count due to IMPACT_AUDIOFRAME_COUNT

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

**Parameters**

| | |
|---|---|
| *input* | audio to process |

**Returns**

processed audio

Definition at line 780 of file sound4.impact.cl.hpp.

#### 10.9.3.17 ProcessAudio()

```
void sound4::impact::CInstance::ProcessAudio ( ) [inline]
```

Process one audio chunk, from GetBufferIn() to GetBufferOut()

**See also**

impact_ProcessAudio

Definition at line 743 of file sound4.impact.cl.hpp.

#### 10.9.3.18 ProcessAudio_Planar()

```
void sound4::impact::CInstance::ProcessAudio_Planar (
            float const *const * input,
            float *const * output ) [inline]
```

Process one audio chunk, but with planar audio

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are NOT interlaced, so the buffers are arrays of planar audio : [ [L0,L1,...,Ln] , [R0,R1,...,Rn] ] (L=Left, R=Right)

In case of multiple audio input/output, buffer are [ [aL...], [aR...], [bL...], [bR...], ... ]

**See also**

impact_ProcessAudio_Planar

**Parameters**

| | |
|---|---|
| *input* | has GetAudioInputCount() ∗ GetChannelCount() channels |
| *output* | has GetAudioOutputCount() ∗ GetChannelCount() channels |

Definition at line 762 of file sound4.impact.cl.hpp.

#### 10.9.3.19 SetMetadata()

```
void sound4::impact::CInstance::SetMetadata (
            const std::string & key,
            const char * value )  [inline]
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible. Setting a null value erase the metadata, which is different from setting an empty string.

**Parameters**

| | |
|---|---|
| *key* | the metadata key If key contains a dot (like wm.KEY), it will be sent only to the matching plugin (ie wm here) without the prefix (ie KEY here) Use empty prefix for routing to main only. |
| *value* | the metadata value or nullptr to erase it. |

Definition at line 554 of file sound4.impact.cl.hpp.

#### 10.9.3.20 SetMetadataMulti()

```
void sound4::impact::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

**See also**

CInstance::SetMetadata

**Parameters**

| | |
|---|---|
| *list* | a map of key/value pair. |

Definition at line 572 of file sound4.impact.cl.hpp.

#### 10.9.3.21 SetParam()

```
void sound4::impact::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

**See also**

impact_SetParameter

Definition at line 417 of file sound4.impact.cl.hpp.

#### 10.9.3.22 SetPresetManager()

```
void sound4::impact::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

Sets a custom preset manager Optional, this overrides the `save_path` in Create

**See also**

impact_SetPresetManager

Must be done before Create

**Parameters**

| | |
|---|---|
| *preset_manager* | The custom preset manager |

Definition at line 445 of file sound4.impact.cl.hpp.

#### 10.9.3.23 StartWebServer()

```
bool sound4::impact::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

**See also**

impact_Webserver

Definition at line 632 of file sound4.impact.cl.hpp.

#### 10.9.3.24 Stop()

```
void sound4::impact::CInstance::Stop ( ) [inline]
```

Stops the instance and its web server and its update thread.

**See also**

impact_ExitProcess,
impact_StopUpdateThread and
impact_Webserver_Stop

Definition at line 680 of file sound4.impact.cl.hpp.

#### 10.9.3.25 StopWebServer()

```
void sound4::impact::CInstance::StopWebServer ( ) [inline]
```

Stops the Web server

**See also**

impact_Webserver_Stop

Definition at line 654 of file sound4.impact.cl.hpp.

#### 10.9.3.26 Terminate()

```
int sound4::impact::CInstance::Terminate ( ) [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

**See also**

impact_TerminateProcess

Definition at line 669 of file sound4.impact.cl.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl.hpp

## 10.10 sound4::impact::dyn::CInstance Class Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

## Classes

- class CClient

## Public Member Functions

- CInstance (CDynLib &dynlib)
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &Access_Key_ID, const std::string &Access_Key_Secret, const std::filesystem::path &save_path, int json_port=0, unsigned int frames_per_chunk=64)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- void SetMetadataMulti (const std::unordered_map< std::string, const char * > &list)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

## Protected Member Functions

- template<typename T >
  unsigned int AddAudio (const T *payload, unsigned int nFrame)
- template<typename T >
  unsigned int GetAudio (T *payload, unsigned int max_nFrame)

### 10.10.1 Detailed Description

Instance handling class

Definition at line 790 of file sound4.impact.cl_dyn.hpp.

## 10.10.2 Constructor & Destructor Documentation

### 10.10.2.1 CInstance() [1/2]

```
sound4::impact::dyn::CInstance::CInstance (
            CDynLib & dynlib )  [inline]
```

Creates the instance, with default parameters

Definition at line 795 of file sound4.impact.cl_dyn.hpp.

### 10.10.2.2 CInstance() [2/2]

```
sound4::impact::dyn::CInstance::CInstance (
            const CInstance &  )  [delete]
```

### 10.10.2.3 ~CInstance()

```
sound4::impact::dyn::CInstance::~CInstance ( )  [inline]
```

Destructor, stopping any running tasks

Definition at line 807 of file sound4.impact.cl_dyn.hpp.

## 10.10.3 Member Function Documentation

### 10.10.3.1 AddAudio()

```
template<typename T >
unsigned int sound4::impact::dyn::CInstance::AddAudio (
            const T * payload,
            unsigned int nFrame )  [inline], [protected]
```

Definition at line 1289 of file sound4.impact.cl_dyn.hpp.

### 10.10.3.2 Create()

```
bool sound4::impact::dyn::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & Access_Key_ID,
            const std::string & Access_Key_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 )  [inline]
```

Creates and initialize the instance, and starts its update thread

**See also**

impact_InitProcess2

**Parameters**

| | |
|---|---|
| *LoginKey* | SOUND4 Login key |
| *RadioName* | SOUND4 RadioName |
| *Access_Key_ID* | SOUND4 AWS access ID |
| *Access_Key_Secret* | SOUND4 AWS access Secret |
| *save_path* | Path where we can load/save the configuration |
| *json_port* | Optional json listening port |
| *frames_per_chunk* | The number of audio frames in each chunk of processing. Must be in the list returned by impact_GetPossibleChunkSizeInFrames, or 0 to set default |

**Returns**

false if failed

Definition at line 895 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::impact::dyn::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 1295 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.4 GetBufferIn()

```
std::array<float,InputSampleSize>& sound4::impact::dyn::CInstance::GetBufferIn ( )  [inline]
```

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 1142 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.5 GetBufferOut()

```
std::array<float,OutputSampleSize>& sound4::impact::dyn::CInstance::GetBufferOut ( ) [inline]
```

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() ∗ GetChannelCount() channels

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are interlaced, so the buffers are [L0,R0,L1,R1,...,Ln,Rn] (L=Left, R=Right)

In case of multiple audio input/output, buffer is [aL0,aR0,bL0,bR0,...,aLn,aRn,bLn,bRn]

Definition at line 1159 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.6 GetChunkFrames()

```
unsigned int sound4::impact::dyn::CInstance::GetChunkFrames ( ) [inline]
```

Get the number of audio frames in each chunk of processing.

The value reflects the frames_per_chunk parameter passed to Create

**Returns**

the number of audio frames in each chunk of processing

Definition at line 938 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.7 GetEstimatedDelay()

```
unsigned int sound4::impact::dyn::CInstance::GetEstimatedDelay ( ) [inline]
```

Gets an estimated delay of current processing

**See also**

impact_GetEstimatedDelay

Definition at line 1126 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.8 GetMetadataInfos()

```
std::vector< std::tuple<std::string,std::string> > sound4::impact::dyn::CInstance::Get↩
MetadataInfos ( ) [inline]
```

Get information on known metadata keys.

**Returns**

a list of {key,description} of known metadata used in process and extensions.

Definition at line 1027 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.9 GetParam()

```
std::string sound4::impact::dyn::CInstance::GetParam (
            const std::string & name ) [inline]
```

Gets a parameter value

**See also**

impact_GetParameter

Definition at line 833 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.10 IsOk()

```
bool sound4::impact::dyn::CInstance::IsOk ( ) const [inline]
```

Returns true if an instance has been created and not stopped.

Definition at line 815 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.11 NewClient()

```
std::shared_ptr<CClient> sound4::impact::dyn::CInstance::NewClient ( ) [inline]
```

Creates a low-level JSON client

**See also**

JSON syntax

**Returns**

a new client to communicate with the processing

Definition at line 1281 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.12 OnUpdateThreadStart()

```
virtual void sound4::impact::dyn::CInstance::OnUpdateThreadStart ( ) [inline], [virtual]
```

Definition at line 1049 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.13 OnUpdateThreadStop()

```
virtual void sound4::impact::dyn::CInstance::OnUpdateThreadStop ( ) [inline], [virtual]
```

Definition at line 1051 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.14 operator=()

```
CInstance& sound4::impact::dyn::CInstance::operator= (
            const CInstance & ) [delete]
```

#### 10.10.3.15 PresetManager_InformChange()

```
void sound4::impact::dyn::CInstance::PresetManager_InformChange (
            const std::filesystem::path & relative_path,
            PresetChange_Kind change_kind ) [inline]
```

Inform that a change happened in the storage. This must be used only in conjonction with SetPresetManager

**Parameters**

| | |
|---|---|
| *relative_path* | The path which changed, relative to the root storage |
| *change_kind* | What happened to this path |

Definition at line 952 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.16 ProcessAnyAudio()

```
template<typename T >
std::vector<T> sound4::impact::dyn::CInstance::ProcessAnyAudio (
            const std::vector< T > input ) [inline]
```

Process audio (different formats, less optimized)

Supports for int16_t, int32_t and float (32) in native endianness

**Attention**

the return count is independent from the input count due to IMPACT_AUDIOFRAME_COUNT

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

**Parameters**

| | |
|---|---|
| *input* | audio to process |

**Returns**

processed audio

Definition at line 1207 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.17 ProcessAudio()

```
void sound4::impact::dyn::CInstance::ProcessAudio ( ) [inline]
```

Process one audio chunk, from GetBufferIn() to GetBufferOut()

**See also**

impact_ProcessAudio

Definition at line 1170 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.18 ProcessAudio_Planar()

```
void sound4::impact::dyn::CInstance::ProcessAudio_Planar (
            float const *const * input,
            float *const * output ) [inline]
```

Process one audio chunk, but with planar audio

Each sample is a native 32-bit floating-point number where 1.0 is full scale. Channels are NOT interlaced, so the buffers are arrays of planar audio : [ [L0,L1,...,Ln] , [R0,R1,...,Rn] ] (L=Left, R=Right)

In case of multiple audio input/output, buffer are [ [aL...], [aR...], [bL...], [bR...], ... ]

**See also**

impact_ProcessAudio_Planar

**Parameters**

| | |
|---|---|
| *input* | has GetAudioInputCount() ∗ GetChannelCount() channels |
| *output* | has GetAudioOutputCount() ∗ GetChannelCount() channels |

Definition at line 1189 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.19 SetMetadata()

```
void sound4::impact::dyn::CInstance::SetMetadata (
            const std::string & key,
            const char * value )  [inline]
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible. Setting a null value erase the metadata, which is different from setting an empty string.

**Parameters**

| | |
|---|---|
| *key* | the metadata key If key contains a dot (like wm.KEY), it will be sent only to the matching plugin (ie wm here) without the prefix (ie KEY here) Use empty prefix for routing to main only. |
| *value* | the metadata value or nullptr to erase it. |

Definition at line 970 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.20 SetMetadataMulti()

```
void sound4::impact::dyn::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

**See also**

CInstance::SetMetadata

**Parameters**

| | |
|---|---|
| *list* | a map of key/value pair. |

Definition at line 991 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.21 SetParam()

```
void sound4::impact::dyn::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

**See also**

impact_SetParameter

Definition at line 823 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.22 SetPresetManager()

```
void sound4::impact::dyn::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

Sets a custom preset manager Optional, this overrides the `save_path` in Create

**See also**

impact_SetPresetManager

Must be done before Create

**Parameters**

| | |
|---|---|
| *preset_manager* | The custom preset manager |

Definition at line 862 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.23 StartWebServer()

```
bool sound4::impact::dyn::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

**See also**

impact_Webserver

Definition at line 1059 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.24 Stop()

```
void sound4::impact::dyn::CInstance::Stop ( ) [inline]
```

Stops the instance and its web server and its update thread.

**See also**

impact_ExitProcess,
impact_StopUpdateThread and
impact_Webserver_Stop

Definition at line 1107 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.25 StopWebServer()

```
void sound4::impact::dyn::CInstance::StopWebServer ( ) [inline]
```

Stops the Web server

**See also**

impact_Webserver_Stop

Definition at line 1081 of file sound4.impact.cl_dyn.hpp.

#### 10.10.3.26 Terminate()

```
int sound4::impact::dyn::CInstance::Terminate ( ) [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

**See also**

impact_TerminateProcess

Definition at line 1096 of file sound4.impact.cl_dyn.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.11 sound4::CPresetLoader Class Reference

Custom preset handler helper.

```
#include <sound4cl.hpp>
```

Inherited by MyPresetManager.

### Public Member Functions

- CPresetLoader ()=default
- virtual ~CPresetLoader ()=default
- virtual bool IsReadOnly ()=0
- virtual bool Exists (const std::filesystem::path &name)=0
- virtual bool Remove (const std::filesystem::path &name)=0
- virtual bool Rename (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > GetAll ()=0
- virtual std::string Read (const std::filesystem::path &filename)=0
- virtual bool Write (const std::filesystem::path &filename, const std::string &content)=0

### 10.11.1 Detailed Description

Derive from this class and call CInstance::SetPresetManager

Definition at line 715 of file sound4cl.hpp.

### 10.11.2 Constructor & Destructor Documentation

#### 10.11.2.1 CPresetLoader()

```
sound4::CPresetLoader::CPresetLoader ( ) [default]
```

#### 10.11.2.2 ~CPresetLoader()

```
virtual sound4::CPresetLoader::~CPresetLoader ( ) [virtual], [default]
```

### 10.11.3 Member Function Documentation

#### 10.11.3.1 Exists()

```
virtual bool sound4::CPresetLoader::Exists (
            const std::filesystem::path & name )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.2 GetAll()

```
virtual std::vector<std::filesystem::path> sound4::CPresetLoader::GetAll ( )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.3 IsReadOnly()

```
virtual bool sound4::CPresetLoader::IsReadOnly ( )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.4 Read()

```
virtual std::string sound4::CPresetLoader::Read (
            const std::filesystem::path & filename )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.5 Remove()

```
virtual bool sound4::CPresetLoader::Remove (
            const std::filesystem::path & name )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.6 Rename()

```
virtual bool sound4::CPresetLoader::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.11.3.7 Write()

```
virtual bool sound4::CPresetLoader::Write (
            const std::filesystem::path & filename,
            const std::string & content )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.12 sound4::impact::CPresetLoader Class Reference

```
#include <sound4.impact.cl.hpp>
```

### Public Member Functions

- CPresetLoader ()=default
- virtual ~CPresetLoader ()=default
- virtual bool IsReadOnly ()=0
- virtual bool Exists (const std::filesystem::path &name)=0
- virtual bool Remove (const std::filesystem::path &name)=0
- virtual bool Rename (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > GetAll ()=0
- virtual std::string Read (const std::filesystem::path &filename)=0
- virtual bool Write (const std::filesystem::path &filename, const std::string &content)=0

### 10.12.1 Detailed Description

Custom preset handler helper Derive from this class and call CInstance::SetPresetManager

Definition at line 367 of file sound4.impact.cl.hpp.

### 10.12.2 Constructor & Destructor Documentation

#### 10.12.2.1 CPresetLoader()

```
sound4::impact::CPresetLoader::CPresetLoader ( )  [default]
```

#### 10.12.2.2 ~CPresetLoader()

```
virtual sound4::impact::CPresetLoader::~CPresetLoader ( ) [virtual], [default]
```

### 10.12.3 Member Function Documentation

#### 10.12.3.1 Exists()

```
virtual bool sound4::impact::CPresetLoader::Exists (
            const std::filesystem::path & name ) [pure virtual]
```

#### 10.12.3.2 GetAll()

```
virtual std::vector<std::filesystem::path> sound4::impact::CPresetLoader::GetAll ( ) [pure
virtual]
```

#### 10.12.3.3 IsReadOnly()

```
virtual bool sound4::impact::CPresetLoader::IsReadOnly ( ) [pure virtual]
```

#### 10.12.3.4 Read()

```
virtual std::string sound4::impact::CPresetLoader::Read (
            const std::filesystem::path & filename ) [pure virtual]
```

#### 10.12.3.5 Remove()

```
virtual bool sound4::impact::CPresetLoader::Remove (
            const std::filesystem::path & name ) [pure virtual]
```

#### 10.12.3.6 Rename()

```
virtual bool sound4::impact::CPresetLoader::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [pure virtual]
```

#### 10.12.3.7 Write()

```
virtual bool sound4::impact::CPresetLoader::Write (
            const std::filesystem::path & filename,
            const std::string & content )  [pure virtual]
```

The documentation for this class was generated from the following file:

- sound4.impact.cl.hpp

## 10.13 sound4::impact::dyn::CPresetLoader Class Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

Inherited by MyPresetManager.

### Public Member Functions

- CPresetLoader ()=default
- virtual ~CPresetLoader ()=default
- virtual bool IsReadOnly ()=0
- virtual bool Exists (const std::filesystem::path &name)=0
- virtual bool Remove (const std::filesystem::path &name)=0
- virtual bool Rename (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > GetAll ()=0
- virtual std::string Read (const std::filesystem::path &filename)=0
- virtual bool Write (const std::filesystem::path &filename, const std::string &content)=0

### 10.13.1 Detailed Description

Custom preset handler helper Derive from this class and call CInstance::SetPresetManager

Definition at line 772 of file sound4.impact.cl_dyn.hpp.

### 10.13.2 Constructor & Destructor Documentation

#### 10.13.2.1 CPresetLoader()

```
sound4::impact::dyn::CPresetLoader::CPresetLoader ( ) [default]
```

#### 10.13.2.2 ~CPresetLoader()

```
virtual sound4::impact::dyn::CPresetLoader::~CPresetLoader ( ) [virtual], [default]
```

### 10.13.3 Member Function Documentation

#### 10.13.3.1 Exists()

```
virtual bool sound4::impact::dyn::CPresetLoader::Exists (
            const std::filesystem::path & name ) [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.2 GetAll()

```
virtual std::vector<std::filesystem::path> sound4::impact::dyn::CPresetLoader::GetAll ( )
[pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.3 IsReadOnly()

```
virtual bool sound4::impact::dyn::CPresetLoader::IsReadOnly ( ) [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.4 Read()

```
virtual std::string sound4::impact::dyn::CPresetLoader::Read (
            const std::filesystem::path & filename ) [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.5 Remove()

```
virtual bool sound4::impact::dyn::CPresetLoader::Remove (
            const std::filesystem::path & name )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.6 Rename()

```
virtual bool sound4::impact::dyn::CPresetLoader::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

#### 10.13.3.7 Write()

```
virtual bool sound4::impact::dyn::CPresetLoader::Write (
            const std::filesystem::path & filename,
            const std::string & content )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.14 sound4::CProcessor Class Reference

Dynamic library interface.

```
#include <sound4cl.hpp>
```

## Public Member Functions

- CProcessor ()=default
- bool Load (const std::filesystem::path &filepath)

  *Loads the library.*
- bool IsOk () const

  *Check if the library was loaded correctly.*
- void AudioConvertFrom (const uint8_t ∗payload, float ∗output, size_t nSpl, sound4cl_SampleFormat fmt)
- void AudioConvertTo (const float ∗input, uint8_t ∗payload, size_t nSpl, sound4cl_SampleFormat fmt)
- void StereoToMono (const float ∗input, float ∗output, size_t nSpl)
- void MonoToStereo (const float ∗input, float ∗output, size_t nSpl)
- void StereoToMono_Planar (const float ∗inputL, const float ∗inputR, float ∗output, size_t nSpl)
- void MonoToStereo_Planar (const float ∗input, float ∗outputL, float ∗outputR, size_t nSpl)
- void AudioMonoFromLiveStereo (const uint8_t ∗payload, float ∗output)
- void AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)
- std::string GetVersion ()
- unsigned int GetChunkSizeInFrames ()
- std::vector< unsigned int > GetPossibleChunkSizeInFrames ()
- unsigned int GetChannelCount ()
- unsigned int GetAudioInputCount ()
- unsigned int GetAudioOutputCount ()
- unsigned int GetSampleRate ()
- std::string GetFormatName (const sound4cl_SampleFormat fmt)
- sound4cl_SampleFormat GetFormatFromName (const std::string &name)
- unsigned int GetBytesFromFormat (const sound4cl_SampleFormat fmt)
- void SetLogSeverity (LogSeverity severity)
- void SetLoggerCallback (log_cb_t cb)

## Protected Member Functions

- template<typename T >
  helper::DynFuncHolder< T > GetPrefixSymbol (const std::string &name)

## Protected Attributes

- helper::DynFuncHolder< decltype(sound4cl_GetVersion) > fnGetVersion
- helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > fnGetChunkSizeInFrames
- helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > fnGetChannelCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > fnGetAudioInputCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > fnGetAudioOutputCount
- helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > fnGetSampleRate
- helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > fnSetLoggerCallback
- helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > fnSetLogSeverity
- helper::DynFuncHolder< decltype(sound4cl_NewParameters) > fnNewParameters
- helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > fnFreeParameters
- helper::DynFuncHolder< decltype(sound4cl_SetParameter) > fnSetParameter
- helper::DynFuncHolder< decltype(sound4cl_GetParameter) > fnGetParameter
- helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > fnFreeParameterValue
- helper::DynFuncHolder< decltype(sound4cl_InitProcess) > fnInitProcess
- helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > fnInitProcess2
- helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > fnTerminateProcess
- helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > fnExitProcess

- helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > fnStartUpdateThread
- helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > fnStopUpdateThread
- helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > fnWaitUpdateThreadReady
- helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) > fnProcessAudio
- helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) > fnProcessAudio_Planar
- helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) > fnGetBufferIn
- helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) > fnGetBufferOut
- helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > fnGetEstimatedDelay
- helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > fnGetFormatName
- helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > fnGetFormatFromName
- helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > fnGetBytesFromFormat
- helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > fnGetMaxPacketFrame
- helper::DynFuncHolder< decltype(sound4cl_AddAudio) > fnAddAudio
- helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > fnAddPadAudio
- helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > fnGetOutputCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudio) > fnGetAudio
- helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > fnAudioConvertFrom
- helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > fnAudioConvertTo
- helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > fnStereoToMono
- helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > fnMonoToStereo
- helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > fnAudioMonoFromLiveStereo
- helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > fnAudioMonoToLiveStereo
- helper::DynFuncHolder< decltype(sound4cl_NewClient) > fnNewClient
- helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > fnDeleteClient
- helper::DynFuncHolder< decltype(sound4cl_ProcessJson) > fnProcessJson
- helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) > fnFreeJsonAnswer
- helper::DynFuncHolder< decltype(sound4cl_SaveState) > fnSaveState
- helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) > fnWebserver_tcp
- helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) > fnWebserver_tcp2
- helper::DynFuncHolder< decltype(sound4cl_Webserver) > fnWebserver
- helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > fnWebserver_Stop
- helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > fnWebserver_Status
- helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > fnStereoToMono_Planar
- helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > fnMonoToStereo_Planar
- helper::DynFuncHolder< decltype(sound4cl_SetMetadata) > fnSetMetadata
- helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > fnGetMetadataInfos
- helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) > fnFreeMetadataInfos
- helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > fnSetPresetManager
- helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) > fnPresetManager_InformChange
- helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > fnGetPossibleChunkSizeInFrames
- helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > fnGetProcessChunkFrames
- helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > fnInitProcess3
- helper::DynFuncHolder< decltype(sound4cl_NewBus) > fnNewBus
- helper::DynFuncHolder< decltype(sound4cl_FreeBus) > fnFreeBus
- helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) > fnSetInstanceBus
- helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) > fnSetMetadataMulti

## Friends

- class CBus
- class CInstance

### 10.14.1 Detailed Description

Once you loaded the library with Load(), if it succeed you can use all the standard C functions from this library.

Definition at line 270 of file sound4cl.hpp.

### 10.14.2 Constructor & Destructor Documentation

#### 10.14.2.1 CProcessor()

```
sound4::CProcessor::CProcessor ( ) [default]
```

### 10.14.3 Member Function Documentation

#### 10.14.3.1 AudioConvertFrom()

```
void sound4::CProcessor::AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            sound4cl_SampleFormat fmt ) [inline]
```

Conversion helpers

Definition at line 548 of file sound4cl.hpp.

#### 10.14.3.2 AudioConvertTo()

```
void sound4::CProcessor::AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            sound4cl_SampleFormat fmt ) [inline]
```

Definition at line 550 of file sound4cl.hpp.

#### 10.14.3.3 AudioMonoFromLiveStereo()

```
void sound4::CProcessor::AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )  [inline]
```

Definition at line 560 of file sound4cl.hpp.

#### 10.14.3.4 AudioMonoToLiveStereo()

```
void sound4::CProcessor::AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )  [inline]
```

Definition at line 562 of file sound4cl.hpp.

#### 10.14.3.5 GetAudioInputCount()

```
unsigned int sound4::CProcessor::GetAudioInputCount ( )  [inline]
```

Get the number of inputs for this process

**Remarks**

All inputs have sound4cl_GetChannelCount() channels

**Returns**

The process input count

Definition at line 620 of file sound4cl.hpp.

#### 10.14.3.6 GetAudioOutputCount()

```
unsigned int sound4::CProcessor::GetAudioOutputCount ( )  [inline]
```

Get the number of outputs for this process

**Remarks**

All outputs have sound4cl_GetChannelCount() channels

**Returns**

The process outputs count

Definition at line 630 of file sound4cl.hpp.

#### 10.14.3.7 GetBytesFromFormat()

```
unsigned int sound4::CProcessor::GetBytesFromFormat (
            const sound4cl_SampleFormat fmt )  [inline]
```

Get the format byte size

**Parameters**

| | |
|---|---|
| *fmt* | the format |

**Returns**

the number of bytes used by this format for a single channel audio sample

Definition at line 667 of file sound4cl.hpp.

#### 10.14.3.8 GetChannelCount()

```
unsigned int sound4::CProcessor::GetChannelCount ( ) [inline]
```

Get the number of channels needed by this process

**Returns**

The process channels count

Definition at line 610 of file sound4cl.hpp.

#### 10.14.3.9 GetChunkSizeInFrames()

```
unsigned int sound4::CProcessor::GetChunkSizeInFrames ( ) [inline]
```

Get the number of frames in a process chunk.

The output will be available only by chunks of this size.

**Note**

A Frame is a set of sound4cl_GetChannelCount samples

**Returns**

The process chunk size in frame

Definition at line 581 of file sound4cl.hpp.

#### 10.14.3.10 GetFormatFromName()

```
sound4cl_SampleFormat sound4::CProcessor::GetFormatFromName (
            const std::string & name ) [inline]
```

Convert string to format

**Parameters**

| | |
|---|---|
| *name* | the format string |

**Returns**

the matching format, or INVALID_FORMAT if invalid.

Definition at line 658 of file sound4cl.hpp.

#### 10.14.3.11 GetFormatName()

```
std::string sound4::CProcessor::GetFormatName (
            const sound4cl_SampleFormat fmt )  [inline]
```

Convert format to string

**Parameters**

| | |
|---|---|
| *fmt* | the format |

**Returns**

the string name of the format

Definition at line 649 of file sound4cl.hpp.

#### 10.14.3.12 GetPossibleChunkSizeInFrames()

```
std::vector<unsigned int> sound4::CProcessor::GetPossibleChunkSizeInFrames ( )  [inline]
```

Get the possibles number of frames in a process chunk.

The output will be available only by chunks of this size.

**Note**

A Frame is a set of sound4cl_GetChannelCount samples

**Returns**

An vector of possible process chunk size in frame

Definition at line 592 of file sound4cl.hpp.

#### 10.14.3.13 GetPrefixSymbol()

```
template<typename T >
helper::DynFuncHolder<T> sound4::CProcessor::GetPrefixSymbol (
            const std::string & name )  [inline], [protected]
```

Definition at line 359 of file sound4cl.hpp.

#### 10.14.3.14 GetSampleRate()

```
unsigned int sound4::CProcessor::GetSampleRate ( )  [inline]
```

Get the processing sample-rate

**Remarks**

For now, always return 48000, but should be checked for backward compatibility

**Returns**

The process sample-rate in Hz

Definition at line 640 of file sound4cl.hpp.

#### 10.14.3.15 GetVersion()

```
std::string sound4::CProcessor::GetVersion ( )  [inline]
```

Get the process version string

**Returns**

version string

Definition at line 570 of file sound4cl.hpp.

#### 10.14.3.16 IsOk()

```
bool sound4::CProcessor::IsOk ( ) const  [inline]
```

**Returns**

true if OK.

false if not OK.

Definition at line 540 of file sound4cl.hpp.

#### 10.14.3.17 Load()

```
bool sound4::CProcessor::Load (
            const std::filesystem::path & filepath )  [inline]
```

**Parameters**

| | |
|---|---|
| *filepath* | the path of the library to load. |

**Returns**

true if library was loaded correctly.

false if library load failed. Do not use in this case !

Definition at line 374 of file sound4cl.hpp.

#### 10.14.3.18 MonoToStereo()

```
void sound4::CProcessor::MonoToStereo (
            const float * input,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 554 of file sound4cl.hpp.

#### 10.14.3.19 MonoToStereo_Planar()

```
void sound4::CProcessor::MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nSpl )  [inline]
```

Definition at line 558 of file sound4cl.hpp.

#### 10.14.3.20 SetLoggerCallback()

```
void sound4::CProcessor::SetLoggerCallback (
            log_cb_t cb )  [inline]
```

Sets the log callback. If not set, console output will be used.

Definition at line 679 of file sound4cl.hpp.

#### 10.14.3.21 SetLogSeverity()

```
void sound4::CProcessor::SetLogSeverity (
            LogSeverity severity )  [inline]
```

Sets the log severity (global)

Definition at line 673 of file sound4cl.hpp.

#### 10.14.3.22 StereoToMono()

```
void sound4::CProcessor::StereoToMono (
            const float * input,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 552 of file sound4cl.hpp.

#### 10.14.3.23 StereoToMono_Planar()

```
void sound4::CProcessor::StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 556 of file sound4cl.hpp.

### 10.14.4 Friends And Related Function Documentation

#### 10.14.4.1 CBus

```
friend class CBus  [friend]
```

Definition at line 276 of file sound4cl.hpp.

#### 10.14.4.2 CInstance

```
friend class CInstance  [friend]
```

Definition at line 277 of file sound4cl.hpp.

## 10.14.5 Member Data Documentation

### 10.14.5.1 fnAddAudio

```
helper::DynFuncHolder< decltype(sound4cl_AddAudio ) > sound4::CProcessor::fnAddAudio [protected]
```

Definition at line 309 of file sound4cl.hpp.

### 10.14.5.2 fnAddPadAudio

```
helper::DynFuncHolder< decltype(sound4cl_AddPadAudio ) > sound4::CProcessor::fnAddPadAudio
[protected]
```

Definition at line 310 of file sound4cl.hpp.

### 10.14.5.3 fnAudioConvertFrom

```
helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom ) > sound4::CProcessor::fnAudio↩
ConvertFrom [protected]
```

Definition at line 313 of file sound4cl.hpp.

### 10.14.5.4 fnAudioConvertTo

```
helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo ) > sound4::CProcessor::fnAudio↩
ConvertTo [protected]
```

Definition at line 314 of file sound4cl.hpp.

### 10.14.5.5 fnAudioMonoFromLiveStereo

```
helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > sound4::CProcessor::fn↩
AudioMonoFromLiveStereo [protected]
```

Definition at line 317 of file sound4cl.hpp.

#### 10.14.5.6 fnAudioMonoToLiveStereo

```
helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo ) > sound4::CProcessor::fn↩
AudioMonoToLiveStereo [protected]
```

Definition at line 318 of file sound4cl.hpp.

#### 10.14.5.7 fnDeleteClient

```
helper::DynFuncHolder< decltype(sound4cl_DeleteClient ) > sound4::CProcessor::fnDeleteClient
[protected]
```

Definition at line 320 of file sound4cl.hpp.

#### 10.14.5.8 fnExitProcess

```
helper::DynFuncHolder< decltype(sound4cl_ExitProcess ) > sound4::CProcessor::fnExitProcess
[protected]
```

Definition at line 296 of file sound4cl.hpp.

#### 10.14.5.9 fnFreeBus

```
helper::DynFuncHolder< decltype(sound4cl_FreeBus ) > sound4::CProcessor::fnFreeBus [protected]
```

Definition at line 352 of file sound4cl.hpp.

#### 10.14.5.10 fnFreeJsonAnswer

```
helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer ) > sound4::CProcessor::fnFreeJson↩
Answer [protected]
```

Definition at line 322 of file sound4cl.hpp.

#### 10.14.5.11 fnFreeMetadataInfos

```
helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos ) > sound4::CProcessor::fnFree↩
MetadataInfos [protected]
```

Definition at line 339 of file sound4cl.hpp.

#### 10.14.5.12 fnFreeParameters

```
helper::DynFuncHolder< decltype(sound4cl_FreeParameters ) > sound4::CProcessor::fnFreeParameters
[protected]
```

Definition at line 289 of file sound4cl.hpp.

#### 10.14.5.13 fnFreeParameterValue

```
helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue ) > sound4::CProcessor::fnFree↩
ParameterValue [protected]
```

Definition at line 292 of file sound4cl.hpp.

#### 10.14.5.14 fnGetAudio

```
helper::DynFuncHolder< decltype(sound4cl_GetAudio ) > sound4::CProcessor::fnGetAudio [protected]
```

Definition at line 312 of file sound4cl.hpp.

#### 10.14.5.15 fnGetAudioInputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount ) > sound4::CProcessor::fnGet↩
AudioInputCount [protected]
```

Definition at line 283 of file sound4cl.hpp.

#### 10.14.5.16 fnGetAudioOutputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount ) > sound4::CProcessor::fnGet↩
AudioOutputCount [protected]
```

Definition at line 284 of file sound4cl.hpp.

#### 10.14.5.17 fnGetBufferIn

```
helper::DynFuncHolder< decltype(sound4cl_GetBufferIn ) > sound4::CProcessor::fnGetBufferIn
[protected]
```

Definition at line 302 of file sound4cl.hpp.

#### 10.14.5.18 fnGetBufferOut

```
helper::DynFuncHolder< decltype(sound4cl_GetBufferOut ) > sound4::CProcessor::fnGetBufferOut
[protected]
```

Definition at line 303 of file sound4cl.hpp.

#### 10.14.5.19 fnGetBytesFromFormat

```
helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat ) > sound4::CProcessor::fnGet↩
BytesFromFormat [protected]
```

Definition at line 307 of file sound4cl.hpp.

#### 10.14.5.20 fnGetChannelCount

```
helper::DynFuncHolder< decltype(sound4cl_GetChannelCount ) > sound4::CProcessor::fnGetChannel↩
Count [protected]
```

Definition at line 282 of file sound4cl.hpp.

#### 10.14.5.21 fnGetChunkSizeInFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames ) > sound4::CProcessor::fnGet↩
ChunkSizeInFrames [protected]
```

Definition at line 281 of file sound4cl.hpp.

#### 10.14.5.22 fnGetEstimatedDelay

```
helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay ) > sound4::CProcessor::fnGet↩
EstimatedDelay [protected]
```

Definition at line 304 of file sound4cl.hpp.

#### 10.14.5.23 fnGetFormatFromName

```
helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName ) > sound4::CProcessor::fnGet↩
FormatFromName [protected]
```

Definition at line 306 of file sound4cl.hpp.

#### 10.14.5.24 fnGetFormatName

```
helper::DynFuncHolder< decltype(sound4cl_GetFormatName ) > sound4::CProcessor::fnGetFormatName
[protected]
```

Definition at line 305 of file sound4cl.hpp.

#### 10.14.5.25 fnGetMaxPacketFrame

```
helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame ) > sound4::CProcessor::fnGetMax↩
PacketFrame [protected]
```

Definition at line 308 of file sound4cl.hpp.

#### 10.14.5.26 fnGetMetadataInfos

```
helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos ) > sound4::CProcessor::fnGet↩
MetadataInfos [protected]
```

Definition at line 338 of file sound4cl.hpp.

#### 10.14.5.27 fnGetOutputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetOutputCount ) > sound4::CProcessor::fnGetOutput↩
Count [protected]
```

Definition at line 311 of file sound4cl.hpp.

#### 10.14.5.28 fnGetParameter

```
helper::DynFuncHolder< decltype(sound4cl_GetParameter ) > sound4::CProcessor::fnGetParameter
[protected]
```

Definition at line 291 of file sound4cl.hpp.

#### 10.14.5.29 fnGetPossibleChunkSizeInFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames ) > sound4::CProcessor↩
::fnGetPossibleChunkSizeInFrames [protected]
```

Definition at line 346 of file sound4cl.hpp.

#### 10.14.5.30 fnGetProcessChunkFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames ) > sound4::CProcessor::fnGet↩
ProcessChunkFrames [protected]
```

Definition at line 347 of file sound4cl.hpp.

#### 10.14.5.31 fnGetSampleRate

```
helper::DynFuncHolder< decltype(sound4cl_GetSampleRate ) > sound4::CProcessor::fnGetSampleRate
[protected]
```

Definition at line 285 of file sound4cl.hpp.

#### 10.14.5.32 fnGetVersion

```
helper::DynFuncHolder< decltype(sound4cl_GetVersion ) > sound4::CProcessor::fnGetVersion
[protected]
```

Definition at line 280 of file sound4cl.hpp.

#### 10.14.5.33 fnInitProcess

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess ) > sound4::CProcessor::fnInitProcess
[protected]
```

Definition at line 293 of file sound4cl.hpp.

#### 10.14.5.34 fnInitProcess2

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess2 ) > sound4::CProcessor::fnInitProcess2
[protected]
```

Definition at line 294 of file sound4cl.hpp.

#### 10.14.5.35 fnInitProcess3

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess3 ) > sound4::CProcessor::fnInitProcess3
[protected]
```

Definition at line 348 of file sound4cl.hpp.

#### 10.14.5.36 fnMonoToStereo

```
helper::DynFuncHolder< decltype(sound4cl_MonoToStereo ) > sound4::CProcessor::fnMonoToStereo
[protected]
```

Definition at line 316 of file sound4cl.hpp.

#### 10.14.5.37 fnMonoToStereo_Planar

```
helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar ) > sound4::CProcessor::fnMono↩
ToStereo_Planar [protected]
```

Definition at line 334 of file sound4cl.hpp.

#### 10.14.5.38 fnNewBus

```
helper::DynFuncHolder< decltype(sound4cl_NewBus ) > sound4::CProcessor::fnNewBus [protected]
```

Definition at line 351 of file sound4cl.hpp.

#### 10.14.5.39 fnNewClient

```
helper::DynFuncHolder< decltype(sound4cl_NewClient ) > sound4::CProcessor::fnNewClient [protected]
```

Definition at line 319 of file sound4cl.hpp.

#### 10.14.5.40 fnNewParameters

```
helper::DynFuncHolder< decltype(sound4cl_NewParameters ) > sound4::CProcessor::fnNewParameters
[protected]
```

Definition at line 288 of file sound4cl.hpp.

#### 10.14.5.41 fnPresetManager_InformChange

```
helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange ) > sound4::CProcessor↩
::fnPresetManager_InformChange [protected]
```

Definition at line 343 of file sound4cl.hpp.

#### 10.14.5.42 fnProcessAudio

```
helper::DynFuncHolder< decltype(sound4cl_ProcessAudio ) > sound4::CProcessor::fnProcessAudio
[protected]
```

Definition at line 300 of file sound4cl.hpp.

#### 10.14.5.43 fnProcessAudio_Planar

```
helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar ) > sound4::CProcessor::fn↩
ProcessAudio_Planar [protected]
```

Definition at line 301 of file sound4cl.hpp.

#### 10.14.5.44 fnProcessJson

```
helper::DynFuncHolder< decltype(sound4cl_ProcessJson ) > sound4::CProcessor::fnProcessJson
[protected]
```

Definition at line 321 of file sound4cl.hpp.

#### 10.14.5.45 fnSaveState

```
helper::DynFuncHolder< decltype(sound4cl_SaveState ) > sound4::CProcessor::fnSaveState [protected]
```

Definition at line 323 of file sound4cl.hpp.

#### 10.14.5.46 fnSetInstanceBus

```
helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus ) > sound4::CProcessor::fnSetInstance↩
Bus [protected]
```

Definition at line 353 of file sound4cl.hpp.

#### 10.14.5.47 fnSetLoggerCallback

```
helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback ) > sound4::CProcessor::fnSet↩
LoggerCallback [protected]
```

Definition at line 286 of file sound4cl.hpp.

#### 10.14.5.48 fnSetLogSeverity

```
helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity ) > sound4::CProcessor::fnSetLog↩
Severity [protected]
```

Definition at line 287 of file sound4cl.hpp.

#### 10.14.5.49 fnSetMetadata

```
helper::DynFuncHolder< decltype(sound4cl_SetMetadata ) > sound4::CProcessor::fnSetMetadata
[protected]
```

Definition at line 337 of file sound4cl.hpp.

#### 10.14.5.50 fnSetMetadataMulti

```
helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti ) > sound4::CProcessor::fnSet↩
MetadataMulti [protected]
```

Definition at line 356 of file sound4cl.hpp.

#### 10.14.5.51 fnSetParameter

```
helper::DynFuncHolder< decltype(sound4cl_SetParameter ) > sound4::CProcessor::fnSetParameter
[protected]
```

Definition at line 290 of file sound4cl.hpp.

#### 10.14.5.52 fnSetPresetManager

```
helper::DynFuncHolder< decltype(sound4cl_SetPresetManager ) > sound4::CProcessor::fnSet↩
PresetManager [protected]
```

Definition at line 342 of file sound4cl.hpp.

#### 10.14.5.53 fnStartUpdateThread

```
helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread ) > sound4::CProcessor::fnStart↩
UpdateThread [protected]
```

Definition at line 297 of file sound4cl.hpp.

#### 10.14.5.54 fnStereoToMono

```
helper::DynFuncHolder< decltype(sound4cl_StereoToMono ) > sound4::CProcessor::fnStereoToMono
[protected]
```

Definition at line 315 of file sound4cl.hpp.

#### 10.14.5.55 fnStereoToMono_Planar

```
helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar ) > sound4::CProcessor::fn↩
StereoToMono_Planar [protected]
```

Definition at line 333 of file sound4cl.hpp.

#### 10.14.5.56 fnStopUpdateThread

```
helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread ) > sound4::CProcessor::fnStop↩
UpdateThread [protected]
```

Definition at line 298 of file sound4cl.hpp.

#### 10.14.5.57 fnTerminateProcess

```
helper::DynFuncHolder< decltype(sound4cl_TerminateProcess ) > sound4::CProcessor::fnTerminate↩
Process [protected]
```

Definition at line 295 of file sound4cl.hpp.

#### 10.14.5.58 fnWaitUpdateThreadReady

```
helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady ) > sound4::CProcessor::fn↩
WaitUpdateThreadReady [protected]
```

Definition at line 299 of file sound4cl.hpp.

#### 10.14.5.59 fnWebserver

```
helper::DynFuncHolder< decltype(sound4cl_Webserver ) > sound4::CProcessor::fnWebserver [protected]
```

Definition at line 328 of file sound4cl.hpp.

#### 10.14.5.60 fnWebserver_Status

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_Status ) > sound4::CProcessor::fnWebserver↩
_Status [protected]
```

Definition at line 330 of file sound4cl.hpp.

#### 10.14.5.61 fnWebserver_Stop

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop ) > sound4::CProcessor::fnWebserver_↩
Stop [protected]
```

Definition at line 329 of file sound4cl.hpp.

#### 10.14.5.62 fnWebserver_tcp

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp ) > sound4::CProcessor::fnWebserver_tcp
[protected]
```

Definition at line 326 of file sound4cl.hpp.

#### 10.14.5.63 fnWebserver_tcp2

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2 ) > sound4::CProcessor::fnWebserver_↩
tcp2 [protected]
```

Definition at line 327 of file sound4cl.hpp.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.15 sound4::helper::DynFuncHolder< T > Class Template Reference

Helper to wrap a function pointer.

```
#include <sound4cl.hpp>
```

### Public Member Functions

- DynFuncHolder ()=default
- DynFuncHolder (DynFunc_t a_ptr)
- operator bool () const
- bool IsOk () const
- operator T* () const

### 10.15.1 Detailed Description

**template<typename T>**
**class sound4::helper::DynFuncHolder< T >**

**Parameters**

| | |
|---|---|
| *T* | is the function type, you may use decltype(fn) |

Definition at line 115 of file sound4cl.hpp.

## 10.15.2 Constructor & Destructor Documentation

### 10.15.2.1 DynFuncHolder() [1/2]

```
template<typename T >
sound4::helper::DynFuncHolder< T >::DynFuncHolder ( ) [default]
```

### 10.15.2.2 DynFuncHolder() [2/2]

```
template<typename T >
sound4::helper::DynFuncHolder< T >::DynFuncHolder (
            DynFunc_t a_ptr ) [inline]
```

Definition at line 120 of file sound4cl.hpp.

## 10.15.3 Member Function Documentation

### 10.15.3.1 IsOk()

```
template<typename T >
bool sound4::helper::DynFuncHolder< T >::IsOk ( ) const [inline]
```

Definition at line 122 of file sound4cl.hpp.

### 10.15.3.2 operator bool()

```
template<typename T >
sound4::helper::DynFuncHolder< T >::operator bool ( ) const [inline]
```

Definition at line 121 of file sound4cl.hpp.

#### 10.15.3.3 operator T∗()

```
template<typename T >
sound4::helper::DynFuncHolder< T >::operator T* ( ) const  [inline]
```

Definition at line 123 of file sound4cl.hpp.

The documentation for this class was generated from the following file:

- sound4cl.hpp

## 10.16 sound4::impact::dyn::helper::DynFuncHolder< T > Class Template Reference

Helper to wrap a function pointer.

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Member Functions

- DynFuncHolder ()=default
- DynFuncHolder (DynFunc_t a_ptr)
- operator bool () const
- bool IsOk () const
- operator T∗ () const

### 10.16.1 Detailed Description

**template<typename T>**
**class sound4::impact::dyn::helper::DynFuncHolder< T >**

**Parameters**

| | |
|---|---|
| *T* | is the function type, you may use decltype(fn) |

Definition at line 86 of file sound4.impact.cl_dyn.hpp.

### 10.16.2 Constructor & Destructor Documentation

#### 10.16.2.1 DynFuncHolder() [1/2]

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::DynFuncHolder ( )  [default]
```

#### 10.16.2.2 DynFuncHolder() [2/2]

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::DynFuncHolder (
            DynFunc_t a_ptr )  [inline]
```

Definition at line 91 of file sound4.impact.cl_dyn.hpp.

### 10.16.3 Member Function Documentation

#### 10.16.3.1 IsOk()

```
template<typename T >
bool sound4::impact::dyn::helper::DynFuncHolder< T >::IsOk ( ) const  [inline]
```

Definition at line 93 of file sound4.impact.cl_dyn.hpp.

#### 10.16.3.2 operator bool()

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::operator bool ( ) const  [inline]
```

Definition at line 92 of file sound4.impact.cl_dyn.hpp.

#### 10.16.3.3 operator T*()

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::operator T* ( ) const  [inline]
```

Definition at line 94 of file sound4.impact.cl_dyn.hpp.

The documentation for this class was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.17 impact_CClientInstance Struct Reference

```
#include <sound4.impact.cl.h>
```

### 10.17.1 Detailed Description

Abstract pointer for a client instance

The documentation for this struct was generated from the following file:

- sound4.impact.cl.h

## 10.18 impact_CInstance Struct Reference

```
#include <sound4.impact.cl.h>
```

### 10.18.1 Detailed Description

Abstract holder for a process instance

The documentation for this struct was generated from the following file:

- sound4.impact.cl.h

## 10.19 impact_CParameters Struct Reference

```
#include <sound4.impact.cl.h>
```

### 10.19.1 Detailed Description

Abstract holder for a process instance parameters

Definition at line 331 of file sound4.impact.cl.h.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.h

## 10.20 MyPresetManager Class Reference

[Custom preset manager]

Inherits sound4::impact::dyn::CPresetLoader, and sound4::CPresetLoader.

## Public Member Functions

- MyPresetManager ()=default
- virtual ~MyPresetManager ()=default
- bool SetupPath (const std::filesystem::path &a_preset_path)
- virtual bool IsReadOnly () override
- virtual bool Exists (const std::filesystem::path &name) override
- virtual bool Remove (const std::filesystem::path &name) override
- virtual bool Rename (const std::filesystem::path &from, const std::filesystem::path &to) override
- virtual std::vector< std::filesystem::path > GetAll () override
- virtual std::string Read (const std::filesystem::path &filename) override
- virtual bool Write (const std::filesystem::path &filename, const std::string &content) override
- MyPresetManager ()=default
- virtual ~MyPresetManager ()=default
- bool SetupPath (const std::filesystem::path &a_preset_path)
- virtual bool IsReadOnly () override
- virtual bool Exists (const std::filesystem::path &name) override
- virtual bool Remove (const std::filesystem::path &name) override
- virtual bool Rename (const std::filesystem::path &from, const std::filesystem::path &to) override
- virtual std::vector< std::filesystem::path > GetAll () override
- virtual std::string Read (const std::filesystem::path &filename) override
- virtual bool Write (const std::filesystem::path &filename, const std::string &content) override

### 10.20.1 Detailed Description

Definition at line 119 of file sound4.impact.cl-pipe-dyn.cpp.

### 10.20.2 Constructor & Destructor Documentation

#### 10.20.2.1 MyPresetManager() [1/2]

```
MyPresetManager::MyPresetManager ( )  [default]
```

#### 10.20.2.2 ~MyPresetManager() [1/2]

```
virtual MyPresetManager::~MyPresetManager ( )  [virtual], [default]
```

#### 10.20.2.3 MyPresetManager() [2/2]

```
MyPresetManager::MyPresetManager ( )  [default]
```

#### 10.20.2.4 ~MyPresetManager() [2/2]

```
virtual MyPresetManager::~MyPresetManager ( ) [virtual], [default]
```

### 10.20.3 Member Function Documentation

#### 10.20.3.1 Exists() [1/2]

```
virtual bool MyPresetManager::Exists (
            const std::filesystem::path & name ) [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 137 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.2 Exists() [2/2]

```
virtual bool MyPresetManager::Exists (
            const std::filesystem::path & name ) [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 121 of file sound4cl-pipe.cpp.

#### 10.20.3.3 GetAll() [1/2]

```
virtual std::vector<std::filesystem::path> MyPresetManager::GetAll ( ) [inline], [override],
[virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 153 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.4 GetAll() [2/2]

```
virtual std::vector<std::filesystem::path> MyPresetManager::GetAll ( ) [inline], [override],
[virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 137 of file sound4cl-pipe.cpp.

#### 10.20.3.5 IsReadOnly() [1/2]

```
virtual bool MyPresetManager::IsReadOnly ( )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 136 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.6 IsReadOnly() [2/2]

```
virtual bool MyPresetManager::IsReadOnly ( )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 120 of file sound4cl-pipe.cpp.

#### 10.20.3.7 Read() [1/2]

```
virtual std::string MyPresetManager::Read (
            const std::filesystem::path & filename )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 163 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.8 Read() [2/2]

```
virtual std::string MyPresetManager::Read (
            const std::filesystem::path & filename )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 147 of file sound4cl-pipe.cpp.

#### 10.20.3.9 Remove() [1/2]

```
virtual bool MyPresetManager::Remove (
            const std::filesystem::path & name )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 142 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.10 Remove() [2/2]

```
virtual bool MyPresetManager::Remove (
            const std::filesystem::path & name )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 126 of file sound4cl-pipe.cpp.

#### 10.20.3.11 Rename() [1/2]

```
virtual bool MyPresetManager::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 147 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.12 Rename() [2/2]

```
virtual bool MyPresetManager::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 131 of file sound4cl-pipe.cpp.

#### 10.20.3.13 SetupPath() [1/2]

```
bool MyPresetManager::SetupPath (
            const std::filesystem::path & a_preset_path )  [inline]
```

Definition at line 124 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.14 SetupPath() [2/2]

```
bool MyPresetManager::SetupPath (
            const std::filesystem::path & a_preset_path )  [inline]
```

Definition at line 108 of file sound4cl-pipe.cpp.

#### 10.20.3.15 Write() [1/2]

```
virtual bool MyPresetManager::Write (
            const std::filesystem::path & filename,
            const std::string & content )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 173 of file sound4.impact.cl-pipe-dyn.cpp.

#### 10.20.3.16 Write() [2/2]

```
virtual bool MyPresetManager::Write (
            const std::filesystem::path & filename,
            const std::string & content )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 157 of file sound4cl-pipe.cpp.

The documentation for this class was generated from the following files:

- sound4.impact.cl-pipe-dyn.cpp
- sound4cl-pipe.cpp

## 10.21 sound4::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

```
#include <sound4cl.hpp>
```

### 10.21.1 Detailed Description

**template<typename T>**
**struct sound4::helper::SampleFormat< T >**

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

Definition at line 85 of file sound4cl.hpp.

The documentation for this struct was generated from the following file:

- sound4cl.hpp

## 10.22 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

```
#include <sound4.impact.cl_dyn.hpp>
```

### 10.22.1 Detailed Description

**template<typename T>**
**struct sound4::impact::dyn::helper::SampleFormat< T >**

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

Definition at line 509 of file sound4.impact.cl_dyn.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.23 sound4::impact::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

```
#include <sound4.impact.cl.hpp>
```

### 10.23.1 Detailed Description

**template<typename T>**
**struct sound4::impact::helper::SampleFormat< T >**

**Template Parameters**

| | |
|---|---|
| *T* | the C type for one audio sample |

Definition at line 116 of file sound4.impact.cl.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.hpp

## 10.24 sound4::helper::SampleFormat< float > Struct Reference

```
#include <sound4cl.hpp>
```

### Public Attributes

- const sound4cl_SampleFormat format = F32_NATIVE

### 10.24.1 Detailed Description

Definition at line 96 of file sound4cl.hpp.

### 10.24.2 Member Data Documentation

#### 10.24.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 97 of file sound4cl.hpp.

The documentation for this struct was generated from the following file:

- sound4cl.hpp

## 10.25 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Attributes

- const impact_SampleFormat format = F32_NATIVE

### 10.25.1 Detailed Description

Definition at line 520 of file sound4.impact.cl_dyn.hpp.

### 10.25.2 Member Data Documentation

#### 10.25.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 521 of file sound4.impact.cl_dyn.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.26 sound4::impact::helper::SampleFormat< float > Struct Reference

```
#include <sound4.impact.cl.hpp>
```

### Public Attributes

- const impact_SampleFormat format = F32_NATIVE

### 10.26.1 Detailed Description

Definition at line 127 of file sound4.impact.cl.hpp.

### 10.26.2 Member Data Documentation

#### 10.26.2.1 format

```
const impact_SampleFormat sound4::impact::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 128 of file sound4.impact.cl.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.hpp

## 10.27 sound4::helper::SampleFormat< int16_t > Struct Reference

```
#include <sound4cl.hpp>
```

### Public Attributes

- const sound4cl_SampleFormat format = S16_NATIVE

### 10.27.1 Detailed Description

Definition at line 88 of file sound4cl.hpp.

### 10.27.2 Member Data Documentation

#### 10.27.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< int16_t >::format = S16_NATIVE
```

Definition at line 89 of file sound4cl.hpp.

The documentation for this struct was generated from the following file:

- sound4cl.hpp

## 10.28 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Attributes

- const impact_SampleFormat format = S16_NATIVE

### 10.28.1 Detailed Description

Definition at line 512 of file sound4.impact.cl_dyn.hpp.

### 10.28.2 Member Data Documentation

#### 10.28.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< int16_t >::format =
S16_NATIVE
```

Definition at line 513 of file sound4.impact.cl_dyn.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.29 sound4::impact::helper::SampleFormat< int16_t > Struct Reference

```
#include <sound4.impact.cl.hpp>
```

### Public Attributes

- const impact_SampleFormat format = S16_NATIVE

### 10.29.1 Detailed Description

Definition at line 119 of file sound4.impact.cl.hpp.

### 10.29.2 Member Data Documentation

#### 10.29.2.1 format

```
const impact_SampleFormat sound4::impact::helper::SampleFormat< int16_t >::format = S16_NATIVE
```

Definition at line 120 of file sound4.impact.cl.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.hpp

## 10.30 sound4::helper::SampleFormat< int32_t > Struct Reference

```
#include <sound4cl.hpp>
```

### Public Attributes

- const sound4cl_SampleFormat format = S32_NATIVE

### 10.30.1 Detailed Description

Definition at line 92 of file sound4cl.hpp.

### 10.30.2 Member Data Documentation

#### 10.30.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 93 of file sound4cl.hpp.

The documentation for this struct was generated from the following file:

- sound4cl.hpp

## 10.31 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference

```
#include <sound4.impact.cl_dyn.hpp>
```

### Public Attributes

- const impact_SampleFormat format = S32_NATIVE

### 10.31.1 Detailed Description

Definition at line 516 of file sound4.impact.cl_dyn.hpp.

### 10.31.2 Member Data Documentation

#### 10.31.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< int32_t >::format =
S32_NATIVE
```

Definition at line 517 of file sound4.impact.cl_dyn.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl_dyn.hpp

## 10.32 sound4::impact::helper::SampleFormat< int32_t > Struct Reference

```
#include <sound4.impact.cl.hpp>
```

### Public Attributes

- const impact_SampleFormat format = S32_NATIVE

### 10.32.1 Detailed Description

Definition at line 123 of file sound4.impact.cl.hpp.

### 10.32.2 Member Data Documentation

#### 10.32.2.1 format

```
const impact_SampleFormat sound4::impact::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 124 of file sound4.impact.cl.hpp.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.hpp

## 10.33 SOUND4_ProcessInfo Struct Reference

```
#include <sound4.impact.cl.h>
```

### Public Attributes

- size_t struct_size

  *The size of total this structure, for compatibility checking.*
- const char * name

  *The name of the processor.*
- const char * prefix

  *The prefix of this processor functions.*
- const char * shortname

  *A short name of the processor, single word.*
- const char * version

  *Library version.*

### 10.33.1 Detailed Description

Information on this processor.

**See also**

SOUND4_GetProcessInfo

Definition at line 187 of file sound4.impact.cl.h.

## 10.33.2 Member Data Documentation

### 10.33.2.1 name

```
const char* SOUND4_ProcessInfo::name
```

Definition at line 189 of file sound4.impact.cl.h.

### 10.33.2.2 prefix

```
const char* SOUND4_ProcessInfo::prefix
```

Definition at line 190 of file sound4.impact.cl.h.

### 10.33.2.3 shortname

```
const char* SOUND4_ProcessInfo::shortname
```

Definition at line 191 of file sound4.impact.cl.h.

### 10.33.2.4 struct_size

```
size_t SOUND4_ProcessInfo::struct_size
```

Definition at line 188 of file sound4.impact.cl.h.

### 10.33.2.5 version

```
const char* SOUND4_ProcessInfo::version
```

Definition at line 192 of file sound4.impact.cl.h.

The documentation for this struct was generated from the following file:

- sound4.impact.cl.h

## 10.34 sound4cl_SOUND4_ProcessInfo Struct Reference

```
#include <sound4cl_cdef.h>
```

### Public Attributes

- size_t struct_size

  *The size of total this structure, for compatibility checking.*
- const char * name

  *The name of the processor.*
- const char * prefix

  *The prefix of this processor functions.*
- const char * shortname

  *A short name of the processor, single word.*
- const char * version

  *Library version.*

### 10.34.1 Detailed Description

Definition at line 34 of file sound4cl_cdef.h.

### 10.34.2 Member Data Documentation

#### 10.34.2.1 name

```
const char* sound4cl_SOUND4_ProcessInfo::name
```

Definition at line 36 of file sound4cl_cdef.h.

#### 10.34.2.2 prefix

```
const char* sound4cl_SOUND4_ProcessInfo::prefix
```

Definition at line 37 of file sound4cl_cdef.h.

#### 10.34.2.3 shortname

```
const char* sound4cl_SOUND4_ProcessInfo::shortname
```

Definition at line 38 of file sound4cl_cdef.h.

#### 10.34.2.4 struct_size

```
size_t sound4cl_SOUND4_ProcessInfo::struct_size
```

Definition at line 35 of file sound4cl_cdef.h.

#### 10.34.2.5 version

```
const char* sound4cl_SOUND4_ProcessInfo::version
```

Definition at line 39 of file sound4cl_cdef.h.

The documentation for this struct was generated from the following file:

- sound4cl_cdef.h

# Chapter 11

# File Documentation

## 11.1 sound4.impact.cl-pipe-cxx.cpp File Reference

C++ sample using C++ interface wrapper.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <io.h>
#include <fcntl.h>
#include "sound4.impact.cl.hpp"
```

### Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::impact::CInstance &instance)
- void Usage (const char *progname)
- void MyLogger (sound4::impact::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

### 11.1.1 Function Documentation

#### 11.1.1.1 GetEnv()

```
std::string GetEnv (
            const std::string & name )
```

Definition at line 39 of file sound4.impact.cl-pipe-cxx.cpp.

#### 11.1.1.2 main()

```
int main (
            int argc,
            const char ** argv )
```

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 104 of file sound4.impact.cl-pipe-cxx.cpp.

#### 11.1.1.3 MyLogger()

```
void MyLogger (
            sound4::impact::LogSeverity severity,
            const std::string & msg )
```

Definition at line 99 of file sound4.impact.cl-pipe-cxx.cpp.

#### 11.1.1.4 ProcessLoop()

```
void ProcessLoop (
            sound4::impact::CInstance & instance )
```

[Processing Loop]

[Processing Loop]

Definition at line 48 of file sound4.impact.cl-pipe-cxx.cpp.

#### 11.1.1.5 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 79 of file sound4.impact.cl-pipe-cxx.cpp.

## 11.2 sound4.impact.cl-pipe-cxx.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #if defined(_WIN32) || defined(WIN32)
00010     #include <io.h>
00011     #include <fcntl.h>
00012 #endif
00013
00014 #include "sound4.impact.cl.hpp"
00015
00016 // ------------------------------------------------------------
00039 std::string GetEnv(const std::string& name)
00040 {
00041     char *data=getenv(name.c_str());
00042     if (!data) return {};
00043     return std::string(data);
00044 }
00045
00046
00047 // Reads from stdin, write to stdout
00048 void ProcessLoop(sound4::impact::CInstance &instance)
00049 {
00050 bool stopthread=false;
00051
00052     #if defined(_WIN32) || defined(WIN32)
00053         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00054         _setmode(_fileno(stdin), _O_BINARY);
00055         _setmode(_fileno(stdout), _O_BINARY);
00056     #endif
00057
00059     while (!stopthread)
00060     {
00061         // Read a chunk from stdin
00062         if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::impact::InputSampleSize,
       stdin) != sound4::impact::InputSampleSize) {
00063             std::cerr « "ProcessLoop: input stopped"« std::endl;
00064             stopthread=true;
00065             break;
00066         }
00067         if (stopthread) break;
00068         // Process it
00069         instance.ProcessAudio();
00070         // Write it to stdout
00071         if (fwrite(instance.GetBufferOut().data(), sizeof(float), sound4::impact::OutputSampleSize,
       stdout) != sound4::impact::OutputSampleSize) {
00072             std::cerr « "Failed to write audio output" « std::endl;
00073             stopthread=true;
00074         }
00075     }
00077 }
00078
00079 void Usage(const char *progname)
00080 {
00081     std::cerr « "Usage: " « progname « std::endl;;
00082     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00083     std::cerr « "Environment:"« std::endl;;
00084     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
       std::endl;;
00085     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00086     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00087     std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00088     std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
00089     std::cerr « std::endl;
00090     std::cerr « "Example:"« std::endl;
00091     std::cerr « "ffmpeg -re -i file.wav -ar "«sound4::impact::GetSampleRate()«" -ac
       "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00092     #if defined(_WIN32) || defined(WIN32)
00093         std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«sound4::impact::GetSampleRate()«"
       -ac "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00094     #else
00095         std::cerr « "ffmpeg -f pulse -i default -ar "«sound4::impact::GetSampleRate()«" -ac
       "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00096     #endif
00097 }
00098
00099 void MyLogger(sound4::impact::LogSeverity severity,const std::string& msg)
00100 {
00101     std::cerr « msg « std::endl;
00102 }
```

```
int main(int argc, const char **argv)
{
#if (IMPACT_HAS_WEBSERVER)
    int listen_port = 8080;
#endif // (IMPACT_HAS_WEBSERVER)
    auto store_path_str=GetEnv("S4STORE_PATH");

    // Get credentials from environment
    auto LoginKey=GetEnv("S4LOGINKEY");
    if (LoginKey.empty()) {
        LoginKey=GetEnv("S4USERID");
    }
    auto RadioName=GetEnv("RADIO_NAME");
    auto KeyId=GetEnv("ACCESS_KEY_ID");
    auto AccessKey=GetEnv("ACCESS_KEY_SECRET");

    const char *progname=argv[0];

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname);
            return 0;
        }
    }
    if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname);
        return 1;
    }
    std::filesystem::path store_path;
    if (!store_path_str.empty()) {
        store_path = std::filesystem::path(store_path_str);
        if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
            std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
std::endl;
            Usage(progname);
            return 1;
        }
    }

    sound4::impact::SetLogSeverity(verbose5);
    sound4::impact::SetLoggerCallback(MyLogger);

    sound4::impact::CInstance instance;
    instance.SetParam("ADMIN_USER","admin");
    instance.SetParam("ADMIN_SECRET","admin");

    std::cerr « "Creating" « std::endl;
    if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
        std::cerr « "Failed to create the processing instance" « std::endl;
        return 1;
    }

#if (IMPACT_HAS_WEBSERVER)
    std::cerr « "Starting web server" « std::endl;
    if (!instance.StartWebServer(listen_port,0)) {
        std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
        return 1;
    }
#endif // (IMPACT_HAS_WEBSERVER)

    {
        auto client = instance.NewClient();
        auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
        std::cerr « "Request for processor name returned "«answer«std::endl;
    }

    ProcessLoop(instance);

    std::cerr « "Stopping now" « std::endl;
    instance.Stop();

    std::cerr « "End" « std::endl;
    return 0;
}
```

## 11.3 sound4.impact.cl-pipe-dyn.cpp File Reference

C++ sample using C++ interface wrapper with dynamic library loading.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <io.h>
#include <fcntl.h>
#include "sound4.impact.cl_dyn.hpp"
```

## Classes

- class MyPresetManager

  *[Custom preset manager]*

## Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::impact::dyn::CInstance &instance)
- void Usage (const char *progname, sound4::impact::dyn::CDynLib &dynlib)
- void MyLogger (sound4::impact::dyn::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

  *[Custom preset manager]*

### 11.3.1 Function Documentation

#### 11.3.1.1 GetEnv()

```
std::string GetEnv (
            const std::string & name )
```

Definition at line 51 of file sound4.impact.cl-pipe-dyn.cpp.

#### 11.3.1.2 main()

```
int main (
            int argc,
            const char ** argv )
```

[Dynamic loading]

[Dynamic loading]

[Setup and sanity checks]

[Setup and sanity checks]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 194 of file sound4.impact.cl-pipe-dyn.cpp.

#### 11.3.1.3 MyLogger()

```
void MyLogger (
            sound4::impact::dyn::LogSeverity severity,
            const std::string & msg )
```

Definition at line 113 of file sound4.impact.cl-pipe-dyn.cpp.

#### 11.3.1.4 ProcessLoop()

```
void ProcessLoop (
            sound4::impact::dyn::CInstance & instance )
```

[Processing Loop]

[Processing Loop]

Definition at line 60 of file sound4.impact.cl-pipe-dyn.cpp.

#### 11.3.1.5 Usage()

```
void Usage (
            const char * progname,
            sound4::impact::dyn::CDynLib & dynlib )
```

Definition at line 91 of file sound4.impact.cl-pipe-dyn.cpp.

## 11.4 sound4.impact.cl-pipe-dyn.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #include <fstream>
00010 #include <sstream>
00011 #if defined(_WIN32) || defined(WIN32)
00012     #include <io.h>
00013     #include <fcntl.h>
00014 #endif
00015
00016 #include "sound4.impact.cl_dyn.hpp"
00017
00018 // ------------------------------------------------------------
00051 std::string GetEnv(const std::string& name)
00052 {
00053     char *data=getenv(name.c_str());
00054     if (!data) return {};
00055     return std::string(data);
00056 }
00057
00058
00059 // Reads from stdin, write to stdout
00060 void ProcessLoop(sound4::impact::dyn::CInstance &instance)
00061 {
00062 bool stopthread=false;
00063
00064     #if defined(_WIN32) || defined(WIN32)
00065         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00066         _setmode(_fileno(stdin), _O_BINARY);
00067         _setmode(_fileno(stdout), _O_BINARY);
00068     #endif
00069
00071     while (!stopthread)
00072     {
00073         // Read a chunk from stdin
00074         if (std::fread(instance.GetBufferIn().data(), sizeof(float),
       sound4::impact::dyn::InputSampleSize, stdin) != sound4::impact::dyn::InputSampleSize) {
00075             std::cerr « "ProcessLoop: input stopped"« std::endl;
00076             stopthread=true;
00077             break;
00078         }
00079         if (stopthread) break;
00080         // Process it
00081         instance.ProcessAudio();
00082         // Write it to stdout
00083         if (fwrite(instance.GetBufferOut().data(), sizeof(float),
       sound4::impact::dyn::OutputSampleSize, stdout) != sound4::impact::dyn::OutputSampleSize) {
00084             std::cerr « "Failed to write audio output" « std::endl;
00085             stopthread=true;
00086         }
00087     }
00089 }
00090
00091 void Usage(const char *progname, sound4::impact::dyn::CDynLib &dynlib)
00092 {
00093     std::cerr « "Usage: " « progname « std::endl;;
00094     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00095     std::cerr « "Environment:"« std::endl;;
00096     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
       std::endl;;
00097     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00098     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00099     std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00100     std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
00101     if (dynlib.IsOk()) {
00102         std::cerr « std::endl;
```

```
        std::cerr « "Example:"« std::endl;
        std::cerr « "ffmpeg -re -i file.wav -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #if defined(_WIN32) || defined(WIN32)
            std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #else
            std::cerr « "ffmpeg -f pulse -i default -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #endif
    }
}

void MyLogger(sound4::impact::dyn::LogSeverity severity,const std::string& msg)
{
    std::cerr « msg « std::endl;
}

class MyPresetManager : public sound4::impact::dyn::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;

    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(), std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(entry.path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
            if ( f.fail() ) {
                return false;
            }
            return true;
```

```
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};

int main(int argc, const char **argv)
{
#if (IMPACT_HAS_WEBSERVER)
    int listen_port = 8080;
#endif // (IMPACT_HAS_WEBSERVER)
    auto store_path_str=GetEnv("S4STORE_PATH");

    // Get credentials from environment
    auto LoginKey=GetEnv("S4LOGINKEY");
    if (LoginKey.empty()) {
        LoginKey=GetEnv("S4USERID");
    }
    auto RadioName=GetEnv("RADIO_NAME");
    auto KeyId=GetEnv("ACCESS_KEY_ID");
    auto AccessKey=GetEnv("ACCESS_KEY_SECRET");

    const char *progname=argv[0];


    sound4::impact::dyn::CDynLib dynlib;
    if (!dynlib.Load()) {
        std::cerr « "Failed to load processing library" « std::endl;
        return 1;
    }

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname,dynlib);
            return 0;
        }
    }
    if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname,dynlib);
        return 1;
    }
    std::filesystem::path store_path;
    if (!store_path_str.empty()) {
        store_path = std::filesystem::path(store_path_str);
        if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
            std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
std::endl;
            Usage(progname,dynlib);
            return 1;
        }
    }

    sound4::impact::dyn::SetLogSeverity(dynlib, verbose5);
    sound4::impact::dyn::SetLoggerCallback(dynlib, MyLogger);

    sound4::impact::dyn::CInstance instance(dynlib);
    instance.SetParam("ADMIN_USER","admin");
    instance.SetParam("ADMIN_SECRET","admin");

    MyPresetManager preset_manager;
    if (!store_path.empty()) {
        if (!preset_manager.SetupPath(store_path)) {
            std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
std::endl;
            return 1;
        }
        instance.SetPresetManager(&preset_manager);
    }

    std::cerr « "Creating" « std::endl;
    if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
        std::cerr « "Failed to create the processing instance" « std::endl;
        return 1;
    }

#if (IMPACT_HAS_WEBSERVER)
    std::cerr « "Starting web server" « std::endl;
    if (!instance.StartWebServer(listen_port,0)) {
        std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
        return 1;
    }
#endif // (IMPACT_HAS_WEBSERVER)

```

```
    {
        auto client = instance.NewClient();
        auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
        std::cerr « "Request for processor name returned "«answer«std::endl;
    }

    ProcessLoop(instance);

    std::cerr « "Stopping now" « std::endl;
    instance.Stop();

    std::cerr « "End" « std::endl;
    return 0;
}
```

## 11.5 sound4.impact.cl-pipe.cpp File Reference

C++ sample using only C interface.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <thread>
#include <experimental/filesystem>
#include <io.h>
#include <fcntl.h>
#include <stringapiset.h>
#include <sound4.impact.cl.h>
```

### Namespaces

- std

### Macros

- #define SAMPLE_RATE 48000
- #define CHUNK_SIZE 64
- #define CHANNEL_COUNT 2

### Functions

- void ProcessLoop (struct impact_CInstance ∗pInstance)
- void Usage (const char ∗progname)
- void MyLogger (impact_LogSeverity severe[[maybe_unused]], const char ∗str)
- int main (int argc, char ∗const argv[ ])

### 11.5.1 Macro Definition Documentation

#### 11.5.1.1 CHANNEL_COUNT

```
#define CHANNEL_COUNT 2
```

Definition at line 28 of file sound4.impact.cl-pipe.cpp.

#### 11.5.1.2 CHUNK_SIZE

```
#define CHUNK_SIZE 64
```

Definition at line 27 of file sound4.impact.cl-pipe.cpp.

#### 11.5.1.3 SAMPLE_RATE

```
#define SAMPLE_RATE 48000
```

Definition at line 26 of file sound4.impact.cl-pipe.cpp.

### 11.5.2 Function Documentation

#### 11.5.2.1 main()

```
int main (
            int argc,
            char *const argv[] )
```

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Stopping the processing instance]

[Stopping the processing instance]

Definition at line 90 of file sound4.impact.cl-pipe.cpp.

#### 11.5.2.2 MyLogger()

```
void MyLogger (
            impact_LogSeverity severe] [[maybe_unused],
            const char * str )
```

Definition at line 84 of file sound4.impact.cl-pipe.cpp.

#### 11.5.2.3 ProcessLoop()

```
void ProcessLoop (
            struct impact_CInstance * pInstance )
```

[Processing Loop]

[Processing Loop]

Definition at line 31 of file sound4.impact.cl-pipe.cpp.

#### 11.5.2.4 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 64 of file sound4.impact.cl-pipe.cpp.

## 11.6 sound4.impact.cl-pipe.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <thread>           // NOTE: needs pthread link
00009 // Choose good filesystem version depending on compiler.
00010 #if __has_include(<filesystem>)
00011     #include <filesystem>
00012 #else //if __cpp_lib_experimental_filesystem
00013     // We need the alias from std::experimental::filesystem to std::filesystem
00014     #include <experimental/filesystem>
00015     namespace std {
00016         namespace filesystem = experimental::filesystem;
00017     }
00018 #endif
00019 #if defined(_WIN32) || defined(WIN32)
00020     #include <io.h>
00021     #include <fcntl.h>
00022     #include <stringapiset.h>   // UTF-8 conversion
00023 #endif
00024 #include <sound4.impact.cl.h>
00025
00026 #define SAMPLE_RATE       48000
00027 #define CHUNK_SIZE        64
00028 #define CHANNEL_COUNT     2
00029
00030 // Reads from stdin, write to stdout
00031 void ProcessLoop(struct impact_CInstance *pInstance)
00032 {
00034 bool stopthread=false;
```

```
alignas(64) float audio_in[CHUNK_SIZE*CHANNEL_COUNT];
alignas(64) float audio_out[CHUNK_SIZE*CHANNEL_COUNT];

    #if defined(_WIN32) || defined(WIN32)
        // Windows: The stdin, stdout, and stderr streams always open in text mode by default
        _setmode(_fileno(stdin), _O_BINARY);
        _setmode(_fileno(stdout), _O_BINARY);
    #endif

    while (!stopthread)
    {
        // Read a chunk from stdin
        if (std::fread(&audio_in[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdin) != CHUNK_SIZE*CHANNEL_COUNT) {
            std::cerr « "ProcessLoop: input stopped"« std::endl;
            stopthread=true;
            break;
        }
        if (stopthread) break;
        // Process it
        impact_ProcessAudio(pInstance, audio_in, audio_out);
        // Write it to stdout
        if (fwrite(&audio_out[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdout) != CHUNK_SIZE*CHANNEL_COUNT) {
            std::cerr « "Failed to write audio output" « std::endl;
            stopthread=true;
        }
    }
}

void Usage(const char *progname)
{
    std::cerr « "Usage: " « progname « std::endl;;
    std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
    std::cerr « "Environment:"« std::endl;;
    std::cerr « " - S4STORE_PATH            : path to store config [default: ./state]"« std::endl;;
    std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
    std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
    std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
    std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
    std::cerr « std::endl;
    std::cerr « "Example:"« std::endl;
    std::cerr « "ffmpeg -re -i file.wav -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #if defined(_WIN32) || defined(WIN32)
        std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #else
        std::cerr « "ffmpeg -f pulse -i default -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #endif
}

void MyLogger(impact_LogSeverity severe [[maybe_unused]], const char *str)
{
    std::cerr « str « std::endl;
}


int main(int argc, char *const argv[])
{
    struct impact_CInstance *pInstance = NULL;
    const char *store_path_str=getenv("S4STORE_PATH");
    // Get credentials from environment
    char *pLoginKey=getenv("S4LOGINKEY");
    if (!pLoginKey) {
        pLoginKey=getenv("S4USERID");
    }
    char *pRadioName=getenv("RADIO_NAME");
    char *pKeyId=getenv("ACCESS_KEY_ID");
    char *pAccessKey=getenv("ACCESS_KEY_SECRET");

    const char *progname=argv[0];

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname);
            return 0;
        }
    }
    if (!pLoginKey || !pRadioName || !pKeyId || !pAccessKey) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname);
        return 1;
    }
    std::filesystem::path fs_store_path;
```

```
    if (!store_path_str || store_path_str[0]==0) {
        fs_store_path=std::filesystem::current_path() / "state";
    } else {
        fs_store_path = std::filesystem::path(store_path_str);
    }
    if (!std::filesystem::exists(fs_store_path) || !std::filesystem::is_directory(fs_store_path)) {
        std::cerr « "Storage folder '"«fs_store_path.c_str()«"' is not a valid directory" « std::endl;
        Usage(progname);
        return 1;
    }
    std::string store_path;
    #ifdef _WIN32
    {
        // Convert path to UTF-8
        std::wstring wstr(fs_store_path.wstring());
        if (wstr.empty()) return {};
        int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
        std::string str(size, 0);
        WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
        store_path=str;
    }
    #else // !_WIN32
        store_path=fs_store_path;
    #endif // !_WIN32

    // Optional: Use our own logging, instead of default color output to stderr.
    //impact_SetLoggerCallback(&MyLogger);

    //impact_SetLogSeverity(impact_LogSeverity::info);
    impact_SetLogSeverity(impact_LogSeverity::verbose5);

    std::cerr « "Process version: " « impact_GetVersion() « std::endl;
    // Do some sanity checks
    if (impact_GetSampleRate()!=SAMPLE_RATE) {
      std::cerr « "Bad library sampling rate, expecting '"«SAMPLE_RATE«"' but has "«
impact_GetSampleRate() « std::endl;
      return 1;
    }
    if (impact_GetChunkSizeInFrames()!=CHUNK_SIZE) {
      std::cerr « "Bad library frame size, expecting '"«CHUNK_SIZE«"' but has "«
impact_GetChunkSizeInFrames() « std::endl;
      return 1;
    }
    if (impact_GetChannelCount()!=CHANNEL_COUNT) {
      std::cerr « "Bad library channel count, expecting '"«CHANNEL_COUNT«"' but has "«
impact_GetChannelCount() « std::endl;
      return 1;
    }

    // Setup parameters
    struct impact_CParameters *params = impact_NewParameters();
    // Optional: activate web authentication, set admin password
    // This creates the authentication only for a new storage path
    impact_SetParameter(params,"ADMIN_USER", "admin");
    impact_SetParameter(params,"ADMIN_SECRET", "admin");

    // Create the processing instance
    pInstance=impact_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
    impact_FreeParameters(params);
    if (!pInstance) {
      std::cerr « "Failed to create the process instance"« std::endl;
      return 1;
    }

    // Start the update thread, without JSON server (low priority)
    std::thread updater([pInstance](){
       // will returned when stopped or destroyed
       std::cerr « "impact_StartUpdateThread (no json port)" « std::endl;
       impact_StartUpdateThread(pInstance, 0);
       std::cerr « "impact_StartUpdateThread END" « std::endl;
     });

    // This is not mandatory, but better be sure we are ready right now
    std::cerr « "Waiting update thread" « std::endl;
    if (impact_WaitUpdateThreadReady(pInstance, 5000)<0) {
        std::cerr « "Timeout to create the process instance"« std::endl;
    }
    std::cerr « "Waiting update thread DONE" « std::endl;

#if (IMPACT_HAS_WEBSERVER)
    // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
    std::cerr « "Starting web server" « std::endl;
    int listenport= 8080;
    int listenport_secure= 8443;
    uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
    if (listenport || listenport_secure) {
        webserver = impact_Webserver(listenport,listenport_secure,pInstance);
```

```
        if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
            std::cerr « "Starting web server FAILED" « std::endl;
        } else {
            int web_status=impact_Webserver_Status(webserver);
            if (listenport) {
                if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
                    std::cerr « "Web server listening on port " « listenport;
                } else {
                    std::cerr « "Web server failed to listen on port " « listenport;
                }
            }
            if (listenport_secure) {
                if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
                    std::cerr « "Web server listening on secure port " « listenport_secure;
                } else {
                    std::cerr « "Web server failed to listen on secure port " « listenport_secure;
                }
            }
            std::cerr « "Starting web server DONE" « std::endl;
        }
    }
#endif // (IMPACT_HAS_WEBSERVER)

     // Force load a factory preset now
     // Could be used instead of setting a store_path
     /*{
         impact_CClientInstance *pClient=impact_NewClient(pInstance);
         const char *answer=impact_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }", NULL);
         std::cerr « "Loaded factory preset answer: " « answer« std::endl;
         impact_FreeJsonAnswer(answer);
         impact_DeleteClient(pClient);
     }*/

    std::cerr « "Estimated delay " « ( impact_GetEstimatedDelay(pInstance)*1000.0/SAMPLE_RATE ) « " ms" « std::endl;

    ProcessLoop(pInstance);

#if (IMPACT_HAS_WEBSERVER)
    // Stopping the webserver
    std::cerr « "Stopping web server" « std::endl;
    impact_Webserver_Stop(webserver, 1000);
    std::cerr « "Stopping web server DONE" « std::endl;
#endif // (IMPACT_HAS_WEBSERVER)

    // Stopping the processing instance
    std::cerr « "Stopping Update thread" « std::endl;
    impact_StopUpdateThread(pInstance); // this will exit the update thread
    std::cerr « "Stopping Update thread DONE" « std::endl;

    updater.join();

    impact_ExitProcess(pInstance);

    return 0;
}
```

## 11.7 sound4.impact.cl.h File Reference

C interface for library.

```
#include <stdint.h>
```

### Classes

- struct SOUND4_ProcessInfo

## Macros

- #define IMPACT_DECL __declspec(dllimport)
- #define IMPACT_HAS_WEBSERVER 1
- #define IMPACT_HAS_CLOUDBUS 0
- #define IMPACT_AUDIOFRAME_COUNT 64
- #define IMPACT_CHANNEL_COUNT 2
- #define IMPACT_AUDIO_INPUT_COUNT 1
- #define IMPACT_AUDIO_OUTPUT_COUNT 1
- #define IMPACT_SAMPLE_RATE 48000
- #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
- #define SOUND4_WEBSERVER_HTTP_OK (1<<0)
- #define SOUND4_WEBSERVER_HTTPS_OK (1<<1)

## Typedefs

- typedef void(* impact_loggerfn) (enum impact_LogSeverity, const char *)
- typedef wchar_t fs_char
- typedef char *(* impact_storage_reader) (const fs_char *filename, void *userdata)
- typedef void(* impact_storage_reader_free) (char *content, void *userdata)
- typedef int(* impact_storage_writer) (const fs_char *filename, const char *content, void *userdata)
- typedef int(* impact_storage_exists) (const fs_char *filename, void *userdata)
- typedef fs_char **(* impact_storage_getall) (void *userdata)
- typedef void(* impact_storage_getall_free) (fs_char **all, void *userdata)
- typedef int(* impact_storage_remove) (const fs_char *filename, void *userdata)
- typedef int(* impact_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)

## Enumerations

- enum impact_LogSeverity {
  none = 0 , fatal = 1 , error = 2 , warning = 3 ,
  info = 4 , verbose = 5 , verbose2 = 6 , verbose3 = 7 ,
  verbose4 = 8 , verbose5 = 9 }
- enum impact_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }
- enum impact_SampleFormat {
  INVALID_FORMAT =0 , S16_LE , S16_BE , S24_LE ,
  S24_BE , S32_LE , S32_BE , F32_LE ,
  F32_BE , S16_NATIVE =S16_LE , S24_NATIVE =S24_LE , S32_NATIVE =S32_LE ,
  F32_NATIVE =F32_LE }

## Functions

- const struct SOUND4_ProcessInfo * SOUND4_GetProcessInfo ()
- const char * impact_GetVersion ()
- unsigned int impact_GetChunkSizeInFrames ()
- unsigned int * impact_GetPossibleChunkSizeInFrames ()
- unsigned int impact_GetChannelCount ()
- unsigned int impact_GetAudioInputCount ()
- unsigned int impact_GetAudioOutputCount ()
- unsigned int impact_GetSampleRate ()
- void impact_SetLoggerCallback (impact_loggerfn logger)

- void impact_SetLogSeverity (enum impact_LogSeverity severity)
- struct impact_CParameters * impact_NewParameters ()
- void impact_FreeParameters (struct impact_CParameters *params)
- void impact_SetParameter (struct impact_CParameters *params, const char *name, const char *value)
- const char * impact_GetParameter (struct impact_CParameters *params, const char *name)
- void impact_FreeParameterValue (const char *value)
- struct impact_CInstance * impact_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct impact_CInstance * impact_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct impact_CParameters *parameters)
- struct impact_CInstance * impact_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct impact_CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int impact_GetProcessChunkFrames (struct impact_CInstance *instance)
- int impact_TerminateProcess (struct impact_CInstance *instance)
- void impact_ExitProcess (struct impact_CInstance *instance)
- void impact_SetPresetManager (struct impact_CParameters *params, impact_storage_reader reader, impact_storage_reader_free, impact_storage_writer writer, impact_storage_exists exists, impact_storage_getall getall, impact_storage_getall_free getall_free, impact_storage_remove remove, impact_storage_rename rename, int IsReadOnly, void *userdata)
- void impact_PresetManager_InformChange (struct impact_CInstance *instance, const fs_char *relative_path, enum impact_PresetChange_Kind change_kind)
- void impact_SetMetadata (struct impact_CInstance *instance, const char *key, const char *value)
- void impact_SetMetadataMulti (struct impact_CInstance *instance, const char **keyvalue)
- const char ** impact_GetMetadataInfos (struct impact_CInstance *instance)
- void impact_FreeMetadataInfos (struct impact_CInstance *instance, const char **infos)
- void impact_StartUpdateThread (struct impact_CInstance *instance, unsigned int port)
- void impact_StopUpdateThread (struct impact_CInstance *instance)
- int impact_WaitUpdateThreadReady (struct impact_CInstance *instance, int milliseconds)
- void impact_ProcessAudio (struct impact_CInstance *instance, const float *input, float *output)
- void impact_ProcessAudio_Planar (struct impact_CInstance *instance, float const *const *input, float *const *output)
- float * impact_GetBufferIn (struct impact_CInstance *instance)
- float * impact_GetBufferOut (struct impact_CInstance *instance)
- unsigned int impact_GetEstimatedDelay (struct impact_CInstance *instance)
- const char * impact_GetFormatName (const enum impact_SampleFormat fmt)
- enum impact_SampleFormat impact_GetFormatFromName (const char *name)
- unsigned int impact_GetBytesFromFormat (const enum impact_SampleFormat fmt)
- unsigned int impact_GetMaxPacketFrame (struct impact_CInstance *instance)
- unsigned int impact_AddAudio (struct impact_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum impact_SampleFormat fmt)
- unsigned int impact_AddPadAudio (struct impact_CInstance *instance)
- unsigned int impact_GetOutputCount (struct impact_CInstance *instance)
- unsigned int impact_GetAudio (struct impact_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum impact_SampleFormat fmt)
- void impact_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_StereoToMono (const float *input, float *output, size_t nFrame)
- void impact_MonoToStereo (const float *input, float *output, size_t nFrame)
- void impact_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void impact_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void impact_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)

- void impact_AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)
- struct impact_CClientInstance ∗ impact_NewClient (struct impact_CInstance ∗instance)
- void impact_DeleteClient (struct impact_CClientInstance ∗client)
- const char ∗ impact_ProcessJson (struct impact_CClientInstance ∗client, const char ∗json_str, int ∗need_save)
- void impact_FreeJsonAnswer (const char ∗json_str)
- int impact_SaveState (struct impact_CInstance ∗instance)
- uint64_t impact_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char ∗socket_ip, unsigned int socket_port)
- uint64_t impact_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char ∗socket_ip, unsigned int socket_port, const struct impact_CParameters ∗parameters)
- uint64_t impact_Webserver (unsigned int listenport, unsigned int listenport_secure, struct impact_CInstance ∗instance)
- int impact_Webserver_Stop (uint64_t id, int timeout_ms)
- int impact_Webserver_Status (uint64_t id)

## 11.7.1 Macro Definition Documentation

### 11.7.1.1 IMPACT_DECL

```
#define IMPACT_DECL __declspec(dllimport)
```

Definition at line 14 of file sound4.impact.cl.h.

### 11.7.1.2 IMPACT_HAS_CLOUDBUS

```
#define IMPACT_HAS_CLOUDBUS 0
```

Definition at line 19 of file sound4.impact.cl.h.

### 11.7.1.3 IMPACT_HAS_WEBSERVER

```
#define IMPACT_HAS_WEBSERVER 1
```

Definition at line 18 of file sound4.impact.cl.h.

## 11.8 sound4.impact.cl.h

```
#pragma once

#include <stdint.h>
#ifdef __unix__
    #include <sys/types.h>
    #define IMPACT_DECL     __attribute__ ((visibility ("default")))
#elif defined(_WIN32) || defined(WIN32)
    #ifdef BUILDING_IMPACT_DLL
        #define IMPACT_DECL __declspec(dllexport)
    #else
        #define IMPACT_DECL __declspec(dllimport)
    #endif
#endif

#define IMPACT_HAS_WEBSERVER 1
#define IMPACT_HAS_CLOUDBUS 0

#ifdef __cplusplus
extern "C" {
#endif // __cplusplus

// ------------------------------------------------------------
// ------------------------------------------------------------
// ------------------------------------------------------------

// ------------------------------------------------------------
#define IMPACT_AUDIOFRAME_COUNT      64
#define IMPACT_CHANNEL_COUNT          2
#define IMPACT_AUDIO_INPUT_COUNT      1
#define IMPACT_AUDIO_OUTPUT_COUNT     1
#define IMPACT_SAMPLE_RATE            48000

struct SOUND4_ProcessInfo {
    size_t struct_size;
    const char *name;
    const char *prefix;
    const char *shortname;
    const char *version;
};

IMPACT_DECL const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo();

IMPACT_DECL const char *impact_GetVersion();

IMPACT_DECL unsigned int impact_GetChunkSizeInFrames();

IMPACT_DECL unsigned int* impact_GetPossibleChunkSizeInFrames();

IMPACT_DECL unsigned int impact_GetChannelCount();

IMPACT_DECL unsigned int impact_GetAudioInputCount();

IMPACT_DECL unsigned int impact_GetAudioOutputCount();

IMPACT_DECL unsigned int impact_GetSampleRate();

enum impact_LogSeverity {
    none = 0,
    fatal = 1,
    error = 2,
    warning = 3,
    info = 4,
    verbose = 5,
    verbose2 = 6,
    verbose3 = 7,
    verbose4 = 8,
    verbose5 = 9,
};

typedef void (*impact_loggerfn) (enum impact_LogSeverity, const char *);

IMPACT_DECL void impact_SetLoggerCallback( impact_loggerfn logger );

IMPACT_DECL void impact_SetLogSeverity( enum impact_LogSeverity severity );

// ------------------------------------------------------------
// ------------------------------------------------------------
struct impact_CInstance;

struct impact_CParameters;

IMPACT_DECL struct impact_CParameters *impact_NewParameters();

IMPACT_DECL void impact_FreeParameters(struct impact_CParameters *params);
```

```
IMPACT_DECL void impact_SetParameter(struct impact_CParameters *params, const char *name, const char *value);

IMPACT_DECL const char *impact_GetParameter(struct impact_CParameters *params, const char *name);

IMPACT_DECL void impact_FreeParameterValue(const char *value);

IMPACT_DECL struct impact_CInstance *impact_InitProcess(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path
    );

IMPACT_DECL struct impact_CInstance *impact_InitProcess2(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path,
        const struct impact_CParameters *parameters
    );

IMPACT_DECL struct impact_CInstance *impact_InitProcess3(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path,
        const struct impact_CParameters *parameters,
        unsigned int frames_per_chunk
    );

IMPACT_DECL unsigned int impact_GetProcessChunkFrames(struct impact_CInstance *instance);

IMPACT_DECL int impact_TerminateProcess(struct impact_CInstance *instance);

IMPACT_DECL void impact_ExitProcess(struct impact_CInstance *instance);

#if IMPACT_HAS_CLOUDBUS
// ---------------------------------------------------------------
struct impact_CBus;

IMPACT_DECL struct impact_CBus* impact_NewBus();

IMPACT_DECL void impact_FreeBus(struct impact_CBus *bus);

IMPACT_DECL void impact_SetInstanceBus(struct impact_CParameters *params, struct impact_CBus *bus);


#endif // IMPACT_HAS_CLOUDBUS

// ---------------------------------------------------------------
// ---------------------------------------------------------------

// OS dependent filesystem path char type
#ifdef _WIN32
    typedef wchar_t fs_char;
#else // !_WIN32
    typedef char fs_char;
#endif // !_WIN32

typedef char * (*impact_storage_reader) (const fs_char *filename, void* userdata);

typedef void (*impact_storage_reader_free) (char *content, void* userdata);

typedef int (*impact_storage_writer) (const fs_char *filename, const char *content, void* userdata);

typedef int (*impact_storage_exists) (const fs_char *filename, void* userdata);

typedef fs_char** (*impact_storage_getall) (void* userdata);

typedef void (*impact_storage_getall_free) (fs_char**all, void* userdata);

typedef int (*impact_storage_remove) (const fs_char *filename, void* userdata);

typedef int (*impact_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);

IMPACT_DECL void impact_SetPresetManager(struct impact_CParameters *params
    , impact_storage_reader reader, impact_storage_reader_free
    , impact_storage_writer writer
    , impact_storage_exists exists
    , impact_storage_getall getall, impact_storage_getall_free getall_free
    , impact_storage_remove remove
    , impact_storage_rename rename
```

```
    , int IsReadOnly
    , void* userdata
    );

enum impact_PresetChange_Kind {
    change_kind_created = 1,
    change_kind_modified   ,
    change_kind_deleted    ,
};

IMPACT_DECL void impact_PresetManager_InformChange(struct impact_CInstance *instance, const fs_char *relative_path, enum impact_PresetChange_Kind change_kind);

// ------------------------------------------------------------
// ------------------------------------------------------------

IMPACT_DECL void impact_SetMetadata(struct impact_CInstance *instance, const char* key, const char* value);

IMPACT_DECL void impact_SetMetadataMulti(struct impact_CInstance *instance, const char** keyvalue);

IMPACT_DECL const char** impact_GetMetadataInfos(struct impact_CInstance *instance);

IMPACT_DECL void impact_FreeMetadataInfos(struct impact_CInstance *instance, const char** infos);

// ------------------------------------------------------------
// ------------------------------------------------------------

IMPACT_DECL void impact_StartUpdateThread(struct impact_CInstance *instance, unsigned int port);

IMPACT_DECL void impact_StopUpdateThread(struct impact_CInstance *instance);

IMPACT_DECL int impact_WaitUpdateThreadReady(struct impact_CInstance *instance, int milliseconds);

// ------------------------------------------------------------
// ------------------------------------------------------------
IMPACT_DECL void impact_ProcessAudio(struct impact_CInstance *instance, const float *input, float *output );

IMPACT_DECL void impact_ProcessAudio_Planar(struct impact_CInstance *instance
    , float const * const *input
    , float * const *output);

IMPACT_DECL float *impact_GetBufferIn(struct impact_CInstance *instance);

IMPACT_DECL float *impact_GetBufferOut(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetEstimatedDelay(struct impact_CInstance *instance);

// ------------------------------------------------------------
// ------------------------------------------------------------
    #ifdef _MSC_VER
        #define __BYTE_ORDER__  REG_DWORD
        #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
        #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
    #endif

enum impact_SampleFormat {
    INVALID_FORMAT=0,
    S16_LE,
    S16_BE,
    S24_LE,
    S24_BE,
    S32_LE,
    S32_BE,
    F32_LE,
    F32_BE,

    // Native definitions, only links to the physical ones
    #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
        S16_NATIVE=S16_LE,
        S24_NATIVE=S24_LE,
        S32_NATIVE=S32_LE,
        F32_NATIVE=F32_LE,
    #else
        S16_NATIVE=S16_BE,
        S24_NATIVE=S24_BE,
        S32_NATIVE=S32_BE,
        F32_NATIVE=F32_BE,
    #endif
};

IMPACT_DECL const char *impact_GetFormatName(const enum impact_SampleFormat fmt);

IMPACT_DECL enum impact_SampleFormat impact_GetFormatFromName(const char *name);

IMPACT_DECL unsigned int impact_GetBytesFromFormat(const enum impact_SampleFormat fmt);
```

```
IMPACT_DECL unsigned int impact_GetMaxPacketFrame(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_AddAudio(struct impact_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum impact_SampleFormat fmt );

IMPACT_DECL unsigned int impact_AddPadAudio(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetOutputCount(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetAudio(struct impact_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum impact_SampleFormat fmt );

// ------------------------------------------------------------
// ------------------------------------------------------------
IMPACT_DECL void impact_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum impact_SampleFormat fmt);

IMPACT_DECL void impact_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum impact_SampleFormat fmt);

IMPACT_DECL void impact_StereoToMono(const float *input, float *output, size_t nFrame);

IMPACT_DECL void impact_MonoToStereo(const float *input, float *output, size_t nFrame);

IMPACT_DECL void impact_StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nFrame);

IMPACT_DECL void impact_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nFrame);

IMPACT_DECL void impact_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);

IMPACT_DECL void impact_AudioMonoToLiveStereo(const float *input, uint8_t *payload);

// ------------------------------------------------------------
// ------------------------------------------------------------
struct impact_CClientInstance;

IMPACT_DECL struct impact_CClientInstance *impact_NewClient(struct impact_CInstance *instance);

IMPACT_DECL void impact_DeleteClient(struct impact_CClientInstance *client);

IMPACT_DECL const char *impact_ProcessJson(struct impact_CClientInstance *client, const char *json_str, int *need_save);

IMPACT_DECL void impact_FreeJsonAnswer(const char *json_str);

IMPACT_DECL int impact_SaveState(struct impact_CInstance *instance);

#if IMPACT_HAS_WEBSERVER
// ------------------------------------------------------------
// ------------------------------------------------------------
IMPACT_DECL uint64_t impact_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port);

// ------------------------------------------------------------
IMPACT_DECL uint64_t impact_Webserver_tcp2(unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct impact_CParameters *parameters);

IMPACT_DECL uint64_t impact_Webserver(unsigned int listenport, unsigned int listenport_secure, struct impact_CInstance *instance);

IMPACT_DECL int impact_Webserver_Stop(uint64_t id, int timeout_ms);

IMPACT_DECL int impact_Webserver_Status(uint64_t id);

#define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)

#define SOUND4_WEBSERVER_HTTP_OK               (1«0)
#define SOUND4_WEBSERVER_HTTPS_OK              (1«1)

#endif // IMPACT_HAS_WEBSERVER

#ifdef __cplusplus
} // extern "C"
#endif // __cplusplus
```

## 11.9 sound4.impact.cl.hpp File Reference

C++ wrapper for library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include "sound4.impact.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

## Classes

- struct sound4::impact::helper::SampleFormat< T >

  *Helper for Sample format types.*
- struct sound4::impact::helper::SampleFormat< int16_t >
- struct sound4::impact::helper::SampleFormat< int32_t >
- struct sound4::impact::helper::SampleFormat< float >
- class sound4::impact::CPresetLoader
- class sound4::impact::CInstance
- class sound4::impact::CInstance::CClient

## Namespaces

- sound4
- sound4::impact
- sound4::impact::helper

## Macros

- #define fs_strdup wcsdup

  *helper for OS dependant file name copy*

## Typedefs

- using sound4::impact::SampleFormat = impact_SampleFormat
- using sound4::impact::LogSeverity = impact_LogSeverity
- using sound4::impact::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::impact::PresetChange_Kind = impact_PresetChange_Kind

## Variables

- const size_t sound4::impact::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t sound4::impact::InputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_INPUT_COUNT
- const size_t sound4::impact::OutputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_OUTPUT_COUNT
- const size_t sound4::impact::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::impact::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::impact::SampleRate =IMPACT_SAMPLE_RATE

### 11.9.1 Macro Definition Documentation

#### 11.9.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 25 of file sound4.impact.cl.hpp.

## 11.10 sound4.impact.cl.hpp

```
00001
00004 #pragma once
00005
00006 #include <string>
00007 #include <exception>
00008 #include <thread>
00009 #include <functional>
00010 #include <memory>
00011 #include <array>
00012 #include <assert.h>
00013 #include <stdexcept>
00014 #include <filesystem>
00015 #include <string.h>
00016
00017 #include "sound4.impact.cl.h"
00018
00019
00020 // Filesystem path char type
00021 #ifdef _WIN32
00022     #include <stringapiset.h>
00023     #include <wchar.h>
00025     #define fs_strdup wcsdup
00026 #else // !_WIN32
00028     #define fs_strdup strdup
00029 #endif // !_WIN32
00030
00031 // bridge C callbacks to CPresetLoader
00032 extern "C" {
00033     static char *impact_custom_reader(const fs_char *filename, void* userdata);
00034     static void impact_custom_reader_free(char *content, void* userdata);
00035     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata);
00036     static int impact_custom_exists(const fs_char *filename, void* userdata);
00037     static fs_char** impact_custom_getall(void* userdata);
00038     static void impact_custom_getall_free(fs_char** all, void* userdata);
00039     static int impact_custom_remove(const fs_char *filename, void* userdata);
00040     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041 };
00042
00046 namespace sound4 {
00050 namespace impact {
00051     static constexpr const char *process_name = "SOUND4 IMPACT.CL";
00052     static constexpr const char *process_shortname = "impact";
00053     using SampleFormat = impact_SampleFormat;
00057     const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
00061     const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
00065     const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
00069     const size_t InputSampleSize=ChunkFrames*InputChannels;
00073     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00077     const size_t SampleRate=IMPACT_SAMPLE_RATE;
00078
00082     namespace helper {
00083         #ifdef _WIN32
00089         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00090             if (wstr.empty()) return {};
00091             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
       NULL);
00092             std::string str(size, 0);
00093             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00094             return str;
00095         }
00101         static inline std::wstring UTF8ToWString(const std::string& str) {
00102             if (str.empty()) return std::wstring();
00103             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00104             std::wstring wstr(size, 0);
```

```
            MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
            return wstr;
        }
        #endif // _WIN32

            template<typename T>
        struct SampleFormat {
        };
            template<>
        struct SampleFormat<int16_t> {
            const impact_SampleFormat format = S16_NATIVE;
        };
            template<>
        struct SampleFormat<int32_t> {
            const impact_SampleFormat format = S32_NATIVE;
        };
            template<>
        struct SampleFormat<float> {
            const impact_SampleFormat format = F32_NATIVE;
        };


        static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, impact_SampleFormat fmt) {
            impact_AudioConvertFrom(payload, output, nSpl, fmt);
        }
        static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, impact_SampleFormat fmt) {
            impact_AudioConvertTo(input, payload, nSpl, fmt);
        }
        static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
            impact_StereoToMono(input, output, nSpl);
        }
        static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
            impact_MonoToStereo(input, output, nSpl);
        }
        static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl) {
            impact_StereoToMono_Planar(inputL, inputR, output, nSpl);
        }
        static inline void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl) {
            impact_MonoToStereo_Planar(input, outputL, outputR, nSpl);
        }
        static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
            impact_AudioMonoFromLiveStereo(payload, output);
        }
        static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
            impact_AudioMonoToLiveStereo(input, payload);
        }

    }; // namespace helper

    static inline std::string GetVersion()
        {
            return std::string(impact_GetVersion());
        }

    static inline unsigned int GetChunkSizeInFrames()
        {
            return impact_GetChunkSizeInFrames();
        }

    static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames()
        {
            std::vector<unsigned int> list;
            for (unsigned int* src=impact_GetPossibleChunkSizeInFrames();*src;src++) {
                list.push_back(*src);
            }
            return list;
        }

    static inline unsigned int GetChannelCount()
        {
            return impact_GetChannelCount();
        }

    static inline unsigned int GetAudioInputCount()
        {
            return impact_GetAudioInputCount();
        }

    static inline unsigned int GetAudioOutputCount()
        {
            return impact_GetAudioOutputCount();
        }

```

```
    static inline unsigned int GetSampleRate()
        {
            return impact_GetSampleRate();
        }

    static inline void SanityCheck(bool a_bCheckFrames = true)
    {
        // Do some sanity checks
        if (impact_GetSampleRate()!=IMPACT_SAMPLE_RATE) {
            throw std::runtime_error("Bad library sampling rate");
        }
        if (a_bCheckFrames && impact_GetChunkSizeInFrames()!=ChunkFrames) {
            throw std::runtime_error("Bad library frame size");
        }
        if (impact_GetChannelCount()!=IMPACT_CHANNEL_COUNT) {
            throw std::runtime_error("Bad library channel count");
        }
        if (impact_GetAudioInputCount()!=IMPACT_AUDIO_INPUT_COUNT) {
            throw std::runtime_error("Bad library input count");
        }
        if (impact_GetAudioOutputCount()!=IMPACT_AUDIO_OUTPUT_COUNT) {
            throw std::runtime_error("Bad library output count");
        }
    }

    static inline std::string GetFormatName(const SampleFormat fmt)
        {
            return std::string(impact_GetFormatName(fmt));
        }

    static inline SampleFormat GetFormatFromName(const std::string& name)
        {
            return impact_GetFormatFromName(name.c_str());
        }

    static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
        {
            return impact_GetBytesFromFormat(fmt);
        }

    using LogSeverity = impact_LogSeverity;
    using log_cb_t = std::function<void(LogSeverity,const std::string&)>;

    static log_cb_t _log_cb;
    extern "C" {
        static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
            _log_cb(severity, std::string(c_msg));
        }
    }
    static inline void SetLogSeverity(LogSeverity severity)
        {
            impact_SetLogSeverity(severity);
        }
    static inline void SetLoggerCallback(log_cb_t cb)
        {
            _log_cb=cb;
            impact_SetLoggerCallback(&_log_cb_c);
        }

#if IMPACT_HAS_CLOUDBUS
    class CBus {
        public:
            CBus()
                : m_bus(impact_NewBus()) {}
            ~CBus()
                {
                    impact_FreeBus(m_bus);
                    m_bus = nullptr;
                }
            impact_CBus *Get() const { return m_bus; }
        private:
            impact_CBus *m_bus;
    };
#endif // IMPACT_HAS_CLOUDBUS

    class CPresetLoader {
    public:
        CPresetLoader() = default;
        virtual ~CPresetLoader() = default;

        virtual bool IsReadOnly() = 0;
        virtual bool Exists(const std::filesystem::path &name) = 0;
        virtual bool Remove(const std::filesystem::path &name) = 0;
        virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
        virtual std::vector<std::filesystem::path> GetAll() = 0;
        virtual std::string Read(const std::filesystem::path &filename) = 0;
        virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
```

```
    };
    using PresetChange_Kind = impact_PresetChange_Kind;

    class CInstance {
    public:
        CInstance()
            {
                params = impact_NewParameters();
                if (!params) throw std::bad_alloc();
            }
        // non-copyable
        CInstance( const CInstance& ) = delete;
        CInstance& operator=( const CInstance& ) = delete;
        ~CInstance()
            {
                Stop();
                if (params) impact_FreeParameters(params);
            }
       bool IsOk() const { return instance!=nullptr; }
        void SetParam(const std::string& name, const std::string& value)
            {
                assert(params);
                impact_SetParameter(params, name.c_str(), value.c_str());
            }
        std::string GetParam(const std::string& name)
            {
                assert(params);
                auto c_value=impact_GetParameter(params, name.c_str());
                std::string ret(c_value);
                impact_FreeParameterValue(c_value);
                return ret;
            }

        void SetPresetManager(CPresetLoader *preset_manager)
            {
                if (!preset_manager) return;
                assert(params);
                impact_SetPresetManager(params,
                    impact_custom_reader,
                    impact_custom_reader_free,
                    impact_custom_writer,
                    impact_custom_exists,
                    impact_custom_getall,
                    impact_custom_getall_free,
                    impact_custom_remove,
                    impact_custom_rename,
                    preset_manager->IsReadOnly(),
                    preset_manager
                );
            }

#if IMPACT_HAS_CLOUDBUS
            void SetBus(const CBus& bus) {
                assert(params);
                impact_SetInstanceBus(params,bus.Get());
            }
#endif // IMPACT_HAS_CLOUDBUS

        bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
 AWS_Access_ID, const std::string& AWS_Access_Secret, const std::filesystem::path& save_path, int
 json_port = 0, unsigned int frames_per_chunk=64)
            {
                assert(!instance);
                std::string l_save_path;
                if (!save_path.empty()) {
                    l_save_path = save_path.u8string();
                }
                instance = impact_InitProcess3(LoginKey.c_str(), RadioName.c_str(),
 AWS_Access_ID.c_str(), AWS_Access_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
                if (!instance) return false;
                if (!init_metadata.empty()) {
                    for (auto&& [key,value]: init_metadata) {
                        impact_SetMetadata(instance, key.c_str(), value.c_str());
                    }
                    init_metadata.clear();
                }
                update_thread = std::thread([this, json_port](){
                    OnUpdateThreadStart();
                    impact_StartUpdateThread(instance, json_port);
                    OnUpdateThreadStop();
                });
                int timeout=5000;
                #ifdef DEBUG
                    timeout=60*1000; // For debugger pauses
                #endif
                if (impact_WaitUpdateThreadReady(instance, timeout)<0) {
                    return false;
```

```
                }
                return true;
            }

        unsigned int GetChunkFrames() {
            if (instance) {
                return impact_GetProcessChunkFrames(instance);
            }
            return 0;
        }

        void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind change_kind)
            {
                impact_PresetManager_InformChange(instance, relative_path.c_str(), (impact_PresetChange_Kind) change_kind);
            }

        void SetMetadata(const std::string& key, const char* value)
            {
                if (instance) {
                    impact_SetMetadata(instance, key.c_str(), value);
                } else {
                    init_metadata.push_back( {key, value} );
                }
            }

        void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
            {
                if (instance) {
                    typedef const char* pchar;
                    pchar* keyvalue=new pchar[2*list.size()+1];
                    size_t n=0;
                    for (auto&& [key,value]: list) {
                        keyvalue[2*n+0]=key.c_str();
                        keyvalue[2*n+1]=value;
                        n++;
                    }
                    keyvalue[2*n+0]=nullptr;

                    impact_SetMetadataMulti(instance, keyvalue);

                    delete[] keyvalue;
                } else {
                    for (auto&& [key,value]: list) {
                        if (value) {
                            init_metadata.push_back( {key, value} );
                        }
                    }
                }
            }

        std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
            {
                std::vector< std::tuple<std::string,std::string> > values;
                const char** c_values = impact_GetMetadataInfos(instance);
                if (c_values) {
                    for (const char** c_value=c_values; *c_value; ) {
                        std::string key(*c_value);
                        c_value++;
                        if (*c_value) {
                            std::string descr(*c_value);
                            c_value++;
                            values.push_back( {key,descr} );
                        }
                    }
                    impact_FreeMetadataInfos(instance, c_values);
                }
                return values;
            }

        // Can be overloaded, called at start of UpdateThread in the thread
        virtual void OnUpdateThreadStart() {}
        // Can be overloaded to call at end of UpdateThread in the thread
        virtual void OnUpdateThreadStop() {}

#if IMPACT_HAS_WEBSERVER
        bool StartWebServer(int http_port, int https_port=0)
            {
                assert(instance);
                assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
                webserver = impact_Webserver(http_port,https_port,instance);
                if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
                    return false;
                }
                int web_status=impact_Webserver_Status(webserver);
                if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
```

```
                    return false;
                }
                if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
                    return false;
                }
                return true;
            }
        void StopWebServer()
            {
                if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
                    impact_Webserver_Stop(webserver, 1000);
                    webserver = SOUND4_INVALID_WEBSERVER_ID;
                }
            }
#endif // IMPACT_HAS_WEBSERVER

        int Terminate()
            {
                if (!instance) return 0;
                return impact_TerminateProcess(instance);
            }

        void Stop()
            {
                if (!instance) return;
#if IMPACT_HAS_WEBSERVER
                StopWebServer();
#endif // IMPACT_HAS_WEBSERVER
                if (update_thread.joinable()) {
                    impact_StopUpdateThread(instance); // this will exit the update thread
                    update_thread.join();
                }
                impact_ExitProcess(instance);
                instance=nullptr;
            }

        unsigned int GetEstimatedDelay()
            {
                assert(instance);
                return impact_GetEstimatedDelay(instance);
            }

        std::array<float,InputSampleSize>& GetBufferIn()
            {
                assert(instance);
                float *buf=impact_GetBufferIn(instance);
                return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);

            }
        std::array<float,OutputSampleSize>& GetBufferOut()
            {
                assert(instance);
                float *buf= impact_GetBufferOut(instance);
                return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
            }
        void ProcessAudio()
            {
                assert(instance);
                impact_ProcessAudio(instance, impact_GetBufferIn(instance),
impact_GetBufferOut(instance));
            }
        // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
        void ProcessAudio_Planar(float const * const *input, float * const *output)
            {
                assert(instance);
                impact_ProcessAudio_Planar(instance, input, output);
            }

            template<typename T>
        std::vector<T> ProcessAnyAudio(const std::vector<T> input)
            {
                assert(instance);
                std::vector<T> output;
                unsigned int out_offset=0;
                unsigned int in_offset=0;
                unsigned int todo = input.size();
                while (todo>0) {
                    unsigned int left = AddAudio(&input[in_offset], todo);
                    unsigned int out_avail = impact_GetOutputCount(instance);
                    output.resize(out_offset + out_avail);
                    GetAudio(&output[out_offset], out_avail);
                    out_offset+=out_avail;
                    in_offset += todo-left;
                    todo=left;
                }
                return output;
```

```
            }

        class CClient {
            public:
                CClient(impact_CInstance* instance)
                    {
                        assert(instance);
                        client=impact_NewClient(instance);
                        if (!client) throw std::bad_alloc();
                    }
                // non-copyable
                CClient( const CInstance& ) = delete;
                CClient& operator=( const CInstance& ) = delete;
                ~CClient()
                    {
                        if (client) {
                            impact_DeleteClient(client);
                            client=nullptr;
                        }
                    }
                std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
                    {
                        assert(client);
                        int need_save=0;
                        const char *canswer = impact_ProcessJson(client, request.c_str(), &need_save);
                        if (!canswer) return {};
                        std::string answer(canswer);
                        impact_FreeJsonAnswer (canswer);
                        if (NeedSave) {
                            *NeedSave=(need_save!=0);
                        }
                        return answer;
                    }
            private:
                impact_CClientInstance *client = nullptr;
        };
        std::shared_ptr<CClient> NewClient()
            {
                assert(instance);
                return std::make_shared<CClient>(instance);
            }

    protected:
            template<typename T>
        unsigned int AddAudio(const T* payload, unsigned int nFrame)
            {
                assert(instance);
                return impact_AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame, helper::SampleFormat<T>::format);
            }
            template<typename T>
        unsigned int GetAudio(T* payload, unsigned int max_nFrame)
            {
                assert(instance);
                return impact_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame, helper::SampleFormat<T>::format);
            }

    private:
        impact_CParameters* params = nullptr;
        impact_CInstance* instance = nullptr;
        std::thread update_thread;
        std::vector< std::pair<std::string,std::string> > init_metadata;
#if IMPACT_HAS_WEBSERVER
        uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
#endif // IMPACT_HAS_WEBSERVER
    };

}; // namespace impact
}; // namespace sound4

// bridge C callbacks to CPresetLoader
extern "C" {
    static char *impact_custom_reader(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        std::string content=preset_loader->Read(filename);
        return strdup(content.c_str());
    }
    static void impact_custom_reader_free(char *content, void* userdata) {
        if (!content) return;
        free(content);
    }
    static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Write(filename,content);
```

```
        return res?0:-1;
    }
    static int impact_custom_exists(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Exists(filename);
        return res?0:-1;
    }
    static fs_char** impact_custom_getall(void* userdata) {
        sound4::impact::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->GetAll();
        fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
        for (size_t n=0;n<res.size();n++) {
            all[n]=fs_strdup(res[n].c_str());
        }
        all[res.size()]=nullptr;
        return all;
    }
    static void impact_custom_getall_free(fs_char** all, void* userdata) {
        if (!all) return;
        for (fs_char** one=all;*one;one++) {
            free(*one);
        }
        free(all);
    }
    static int impact_custom_remove(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Remove(filename);
        return res?0:-1;
    }
    static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
        sound4::impact::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Rename(from,to);
        return res?0:-1;
    }

};
```

## 11.11 sound4.impact.cl_dyn.hpp File Reference

C++ wrapper for dynamic loading library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include <libloaderapi.h>
#include "sound4.impact.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

### Classes

- class sound4::impact::dyn::helper::DynFuncHolder< T >

  *Helper to wrap a function pointer.*
- class sound4::impact::dyn::helper::CDynLoader

  *Helper to load dynamic library.*

- class sound4::impact::dyn::CDynLib

  *Dynamic library interface.*
- struct sound4::impact::dyn::helper::SampleFormat< T >

  *Helper for Sample format types.*
- struct sound4::impact::dyn::helper::SampleFormat< int16_t >
- struct sound4::impact::dyn::helper::SampleFormat< int32_t >
- struct sound4::impact::dyn::helper::SampleFormat< float >
- class sound4::impact::dyn::CPresetLoader
- class sound4::impact::dyn::CInstance
- class sound4::impact::dyn::CInstance::CClient

## Namespaces

- sound4
- sound4::impact
- dyn
- sound4::impact::dyn
- sound4::impact::dyn::helper

## Macros

- #define fs_strdup wcsdup

  *helper for OS dependant file name copy*

## Typedefs

- using sound4::impact::dyn::helper::DynLib_t = HMODULE
- using sound4::impact::dyn::helper::DynFunc_t = FARPROC
- using sound4::impact::dyn::SampleFormat = impact_SampleFormat
- using sound4::impact::dyn::LogSeverity = impact_LogSeverity
- using sound4::impact::dyn::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::impact::dyn::PresetChange_Kind = impact_PresetChange_Kind

## Variables

- const size_t sound4::impact::dyn::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t sound4::impact::dyn::InputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_INPUT_COUNT
- const size_t sound4::impact::dyn::OutputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_OUTPUT_COUNT
- const size_t sound4::impact::dyn::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::impact::dyn::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::impact::dyn::SampleRate =IMPACT_SAMPLE_RATE

### 11.11.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4.impact.cl_dyn.hpp.

### 11.11.2 Macro Definition Documentation

#### 11.11.2.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 36 of file sound4.impact.cl_dyn.hpp.

## 11.12 sound4.impact.cl_dyn.hpp

```
00001
00007 #pragma once
00008
00009 #include <string>
00010 #include <exception>
00011 #include <thread>
00012 #include <functional>
00013 #include <memory>
00014 #include <array>
00015 #include <assert.h>
00016 #include <stdexcept>
00017 #include <filesystem>
00018 #include <string.h>
00019
00020 #ifdef __unix__
00021     #include <dlfcn.h>
00022 #elif defined(_WIN32)
00023     #include <libloaderapi.h>
00024 #else
00025     #error "Unsupported OS"
00026 #endif
00027
00028 // Needed to have function declarations
00029 #include "sound4.impact.cl.h"
00030
00031
00032 #ifdef _WIN32
00033     #include <stringapiset.h>
00034     #include <wchar.h>
00036     #define fs_strdup wcsdup
00037 #else // !_WIN32
00039     #define fs_strdup strdup
00040 #endif // !_WIN32
00041
00042 // bridge C callbacks to CPresetLoader
00043 extern "C" {
00044     static char *impact_custom_reader(const fs_char *filename, void* userdata);
00045     static void impact_custom_reader_free(char *content, void* userdata);
00046     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata);
00047     static int impact_custom_exists(const fs_char *filename, void* userdata);
00048     static fs_char** impact_custom_getall(void* userdata);
00049     static void impact_custom_getall_free(fs_char** all, void* userdata);
00050     static int impact_custom_remove(const fs_char *filename, void* userdata);
00051     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00052 };
00053
00057 namespace sound4 {
00061 namespace impact {
00065 namespace dyn {
00066
00070     namespace helper {
00071         // Storage definition, depending on OS
00072         #ifdef __unix__
00073             using DynLib_t = void *;
00074             using DynFunc_t = void *;
00075         #else
00076             using DynLib_t = HMODULE;
00077             using DynFunc_t = FARPROC;
00078         #endif
00079
00085             template <typename T>
00086         class DynFuncHolder {
00087         private:
00088             DynFunc_t m_ptr = NULL;
```

```
        public:
            DynFuncHolder() = default;
            DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
            operator bool() const { return (m_ptr!=NULL); }
            bool IsOk() const { return m_ptr!=NULL; }
            operator T* () const { return reinterpret_cast<T*>(m_ptr); }
        };

        class CDynLoader {
            private:
                DynLib_t m_lib {};
            public:
                CDynLoader() = default;
                CDynLoader(CDynLoader&&) = default;
                // No copy allowed
                CDynLoader(const CDynLoader&) = delete;
                CDynLoader& operator=(CDynLoader const&) = delete;
                operator bool() const {
                    return (m_lib!=NULL);
                }
                bool IsOk() const {
                    return (m_lib!=NULL);
                }
                #ifdef __unix__
                    void Close() {
                        if (IsOk()) {
                            #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
                            dlclose(m_lib);
                            #endif // __SANITIZE_ADDRESS__
                            m_lib=NULL;
                        }
                    }
                    bool Load(const std::filesystem::path& dynlib)
                    {
                        Close();
                        m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
                        return IsOk();
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
                        }
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static Dl_info info;
                        if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
                        auto rp=realpath(info.dli_fname, NULL);
                        std::filesystem::path p(rp);
                        free(rp);
                        return p;
                    }
                #else
                    bool Load(const std::filesystem::path& dynlib, uint32_t loadflags =
LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_
                    {
                        Close();
                        m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
                        return IsOk();
                    }
                    void Close() {
                        if (IsOk()) {
                            // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
FreeLibraryAndExitThread but...
                            // So define SOUND4_CALL_FREELIBRARY before including this file if you
really want to unload the library dynamically
                            #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
                                FreeLibraryAndExitThread(m_lib, 0);
                            #elif defined(SOUND4_CALL_FREELIBRARY)
                                FreeLibrary(m_lib);
                            #endif // SOUND4_CALL_FREELIBRARY
                            m_lib=NULL;
                        }
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
                        }
```

```
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static wchar_t path[MAX_PATH]={};
                        HMODULE hm = NULL;
                        if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
                                GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
                                (LPCWSTR) &GetThisLibraryPath, &hm))
                        {
                            if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
                                return std::filesystem::path(path);
                            }
                        }
                        return {};
                    }
                #endif
                virtual ~CDynLoader() {
                    Close();
                }
        }; // class CDynLoader
    }; // namespace helper

    class CDynLib {
    private:
        helper::CDynLoader m_lib;
        bool m_bOK = false;
    public:
        CDynLib() = default;

        // All those wrapped functions have the same signature as there C impact_XXX
        helper::DynFuncHolder< decltype(impact_GetVersion              ) > GetVersion              ;
        helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames    ) > GetChunkSizeInFrames    ;
        helper::DynFuncHolder< decltype(impact_GetChannelCount         ) > GetChannelCount         ;
        helper::DynFuncHolder< decltype(impact_GetAudioInputCount      ) > GetAudioInputCount      ;
        helper::DynFuncHolder< decltype(impact_GetAudioOutputCount     ) > GetAudioOutputCount     ;
        helper::DynFuncHolder< decltype(impact_GetSampleRate           ) > GetSampleRate           ;
        helper::DynFuncHolder< decltype(impact_SetLoggerCallback       ) > SetLoggerCallback       ;
        helper::DynFuncHolder< decltype(impact_SetLogSeverity          ) > SetLogSeverity          ;
        helper::DynFuncHolder< decltype(impact_NewParameters           ) > NewParameters           ;
        helper::DynFuncHolder< decltype(impact_FreeParameters          ) > FreeParameters          ;
        helper::DynFuncHolder< decltype(impact_SetParameter            ) > SetParameter            ;
        helper::DynFuncHolder< decltype(impact_GetParameter            ) > GetParameter            ;
        helper::DynFuncHolder< decltype(impact_FreeParameterValue      ) > FreeParameterValue      ;
        helper::DynFuncHolder< decltype(impact_InitProcess             ) > InitProcess             ;
        helper::DynFuncHolder< decltype(impact_InitProcess2            ) > InitProcess2            ;
        helper::DynFuncHolder< decltype(impact_TerminateProcess        ) > TerminateProcess        ;
        helper::DynFuncHolder< decltype(impact_ExitProcess             ) > ExitProcess             ;
        helper::DynFuncHolder< decltype(impact_StartUpdateThread       ) > StartUpdateThread       ;
        helper::DynFuncHolder< decltype(impact_StopUpdateThread        ) > StopUpdateThread        ;
        helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady   ) > WaitUpdateThreadReady   ;
        helper::DynFuncHolder< decltype(impact_ProcessAudio            ) > ProcessAudio            ;
        helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar     ) > ProcessAudio_Planar     ;
        helper::DynFuncHolder< decltype(impact_GetBufferIn             ) > GetBufferIn             ;
        helper::DynFuncHolder< decltype(impact_GetBufferOut            ) > GetBufferOut            ;
        helper::DynFuncHolder< decltype(impact_GetEstimatedDelay       ) > GetEstimatedDelay       ;
        helper::DynFuncHolder< decltype(impact_GetFormatName           ) > GetFormatName           ;
        helper::DynFuncHolder< decltype(impact_GetFormatFromName       ) > GetFormatFromName       ;
        helper::DynFuncHolder< decltype(impact_GetBytesFromFormat      ) > GetBytesFromFormat      ;
        helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame       ) > GetMaxPacketFrame       ;
        helper::DynFuncHolder< decltype(impact_AddAudio                ) > AddAudio                ;
        helper::DynFuncHolder< decltype(impact_AddPadAudio             ) > AddPadAudio             ;
        helper::DynFuncHolder< decltype(impact_GetOutputCount          ) > GetOutputCount          ;
        helper::DynFuncHolder< decltype(impact_GetAudio                ) > GetAudio                ;
        helper::DynFuncHolder< decltype(impact_AudioConvertFrom        ) > AudioConvertFrom        ;
        helper::DynFuncHolder< decltype(impact_AudioConvertTo          ) > AudioConvertTo          ;
        helper::DynFuncHolder< decltype(impact_StereoToMono            ) > StereoToMono            ;
        helper::DynFuncHolder< decltype(impact_MonoToStereo            ) > MonoToStereo            ;
        helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo;
        helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo   ) > AudioMonoToLiveStereo   ;
        helper::DynFuncHolder< decltype(impact_NewClient               ) > NewClient               ;
        helper::DynFuncHolder< decltype(impact_DeleteClient            ) > DeleteClient            ;
        helper::DynFuncHolder< decltype(impact_ProcessJson             ) > ProcessJson             ;
        helper::DynFuncHolder< decltype(impact_FreeJsonAnswer          ) > FreeJsonAnswer          ;
        helper::DynFuncHolder< decltype(impact_SaveState               ) > SaveState               ;
#if IMPACT_HAS_WEBSERVER
        helper::DynFuncHolder< decltype(impact_Webserver_tcp           ) > Webserver_tcp           ;
        helper::DynFuncHolder< decltype(impact_Webserver_tcp2          ) > Webserver_tcp2          ;
        helper::DynFuncHolder< decltype(impact_Webserver               ) > Webserver               ;
        helper::DynFuncHolder< decltype(impact_Webserver_Stop          ) > Webserver_Stop          ;
        helper::DynFuncHolder< decltype(impact_Webserver_Status        ) > Webserver_Status        ;
#endif // IMPACT_HAS_WEBSERVER
```

```
        // added 2023-03-22
        helper::DynFuncHolder< decltype(impact_StereoToMono_Planar    ) > StereoToMono_Planar    ;
        helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar    ) > MonoToStereo_Planar    ;

        // added 2023-05-22
        helper::DynFuncHolder< decltype(impact_SetMetadata            ) > SetMetadata            ;
        helper::DynFuncHolder< decltype(impact_GetMetadataInfos       ) > GetMetadataInfos       ;
        helper::DynFuncHolder< decltype(impact_FreeMetadataInfos      ) > FreeMetadataInfos      ;

        // added 2023-06-20
        helper::DynFuncHolder< decltype(impact_SetPresetManager                ) > SetPresetManager ;
        helper::DynFuncHolder< decltype(impact_PresetManager_InformChange      ) > PresetManager_InformChange      ;

        // added 2023-09-21
        helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames    ) > GetPossibleChunkSizeInFrames   ;
        helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames           ) > GetProcessChunkFrames   ;
        helper::DynFuncHolder< decltype(impact_InitProcess3                    ) > InitProcess3 ;

#if IMPACT_HAS_CLOUDBUS
        // added 2023-06-26
        helper::DynFuncHolder< decltype(impact_NewBus                 ) > NewBus      ;
        helper::DynFuncHolder< decltype(impact_FreeBus                ) > FreeBus      ;
        helper::DynFuncHolder< decltype(impact_SetInstanceBus         ) > SetInstanceBus      ;
#endif // IMPACT_HAS_CLOUDBUS

        // added 2024-02-19
        helper::DynFuncHolder< decltype(impact_SetMetadataMulti       ) > SetMetadataMulti         ;

        bool Load(const std::filesystem::path& path = {}) {
            std::string filename;
            #ifdef __unix__
                filename="libsound4.impact.cl.so";
            #else
                filename="sound4.impact.cl.dll";
            #endif
            m_lib.Close();
            // If a path is given, use it directly and do not try other path
            if (!path.empty() && !m_lib.Load(path / filename)) {
                return false;
            } else if (!m_lib.IsOk()) {
                auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
                #ifndef _WIN32
                // Linux: if in a bin directory, try ../lib first
                if (!m_lib.IsOk() && thisdir.filename()=="bin") {
                    auto libdir = thisdir.parent_path() / "lib";
                    m_lib.Load(libdir / filename);
                }
                #endif // !_WIN32
                if (!m_lib.IsOk()) {
                    // Search in the same directory this code is in
                    m_lib.Load(thisdir / filename);
                }
                if (!m_lib.IsOk()) {
                    // Try current path
                    m_lib.Load(std::filesystem::current_path() / filename);
                }
                if (!m_lib.IsOk()) {
                    return false;
                }
            }
            // Load all C functions
            try {
                GetVersion              = m_lib.GetSymbol< decltype(impact_GetVersion              ) >("impact_GetVersion"             );
                GetChunkSizeInFrames    = m_lib.GetSymbol< decltype(impact_GetChunkSizeInFrames    ) >("impact_GetChunkSizeInFrames"   );
                GetChannelCount         = m_lib.GetSymbol< decltype(impact_GetChannelCount         ) >("impact_GetChannelCount"        );
                GetAudioInputCount      = m_lib.GetSymbol< decltype(impact_GetAudioInputCount      ) >("impact_GetAudioInputCount"     );
                GetAudioOutputCount     = m_lib.GetSymbol< decltype(impact_GetAudioOutputCount     ) >("impact_GetAudioOutputCount"    );
                GetSampleRate           = m_lib.GetSymbol< decltype(impact_GetSampleRate           ) >("impact_GetSampleRate"          );
                SetLoggerCallback       = m_lib.GetSymbol< decltype(impact_SetLoggerCallback       ) >("impact_SetLoggerCallback"      );
                SetLogSeverity          = m_lib.GetSymbol< decltype(impact_SetLogSeverity          ) >("impact_SetLogSeverity"         );
                NewParameters           = m_lib.GetSymbol< decltype(impact_NewParameters           ) >("impact_NewParameters"          );
                FreeParameters          = m_lib.GetSymbol< decltype(impact_FreeParameters          ) >("impact_FreeParameters"         );
```

```
                SetParameter            = m_lib.GetSymbol< decltype(impact_SetParameter           )
    >("impact_SetParameter"           );
                GetParameter            = m_lib.GetSymbol< decltype(impact_GetParameter           )
    >("impact_GetParameter"           );
                FreeParameterValue      = m_lib.GetSymbol< decltype(impact_FreeParameterValue     )
    >("impact_FreeParameterValue"     );
                InitProcess             = m_lib.GetSymbol< decltype(impact_InitProcess            )
    >("impact_InitProcess"            );
                InitProcess2            = m_lib.GetSymbol< decltype(impact_InitProcess2           )
    >("impact_InitProcess2"           );
                TerminateProcess        = m_lib.GetSymbol< decltype(impact_TerminateProcess       )
    >("impact_TerminateProcess"       );
                ExitProcess             = m_lib.GetSymbol< decltype(impact_ExitProcess            )
    >("impact_ExitProcess"            );
                StartUpdateThread       = m_lib.GetSymbol< decltype(impact_StartUpdateThread      )
    >("impact_StartUpdateThread"      );
                StopUpdateThread        = m_lib.GetSymbol< decltype(impact_StopUpdateThread       )
    >("impact_StopUpdateThread"       );
                WaitUpdateThreadReady   = m_lib.GetSymbol< decltype(impact_WaitUpdateThreadReady  )
    >("impact_WaitUpdateThreadReady"  );
                ProcessAudio            = m_lib.GetSymbol< decltype(impact_ProcessAudio           )
    >("impact_ProcessAudio"           );
                ProcessAudio_Planar     = m_lib.GetSymbol< decltype(impact_ProcessAudio_Planar    )
    >("impact_ProcessAudio_Planar"    );
                GetBufferIn             = m_lib.GetSymbol< decltype(impact_GetBufferIn            )
    >("impact_GetBufferIn"            );
                GetBufferOut            = m_lib.GetSymbol< decltype(impact_GetBufferOut           )
    >("impact_GetBufferOut"           );
                GetEstimatedDelay       = m_lib.GetSymbol< decltype(impact_GetEstimatedDelay      )
    >("impact_GetEstimatedDelay"      );
                GetFormatName           = m_lib.GetSymbol< decltype(impact_GetFormatName          )
    >("impact_GetFormatName"          );
                GetFormatFromName       = m_lib.GetSymbol< decltype(impact_GetFormatFromName      )
    >("impact_GetFormatFromName"      );
                GetBytesFromFormat      = m_lib.GetSymbol< decltype(impact_GetBytesFromFormat     )
    >("impact_GetBytesFromFormat"     );
                GetMaxPacketFrame       = m_lib.GetSymbol< decltype(impact_GetMaxPacketFrame      )
    >("impact_GetMaxPacketFrame"      );
                AddAudio                = m_lib.GetSymbol< decltype(impact_AddAudio               )
    >("impact_AddAudio"               );
                AddPadAudio             = m_lib.GetSymbol< decltype(impact_AddPadAudio            )
    >("impact_AddPadAudio"            );
                GetOutputCount          = m_lib.GetSymbol< decltype(impact_GetOutputCount         )
    >("impact_GetOutputCount"         );
                GetAudio                = m_lib.GetSymbol< decltype(impact_GetAudio               )
    >("impact_GetAudio"               );
                AudioConvertFrom        = m_lib.GetSymbol< decltype(impact_AudioConvertFrom       )
    >("impact_AudioConvertFrom"       );
                AudioConvertTo          = m_lib.GetSymbol< decltype(impact_AudioConvertTo         )
    >("impact_AudioConvertTo"         );
                StereoToMono            = m_lib.GetSymbol< decltype(impact_StereoToMono           )
    >("impact_StereoToMono"           );
                MonoToStereo            = m_lib.GetSymbol< decltype(impact_MonoToStereo           )
    >("impact_MonoToStereo"           );
                AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoFromLiveStereo)
    >("impact_AudioMonoFromLiveStereo");
                AudioMonoToLiveStereo   = m_lib.GetSymbol< decltype(impact_AudioMonoToLiveStereo  )
    >("impact_AudioMonoToLiveStereo"  );
                NewClient               = m_lib.GetSymbol< decltype(impact_NewClient              )
    >("impact_NewClient"              );
                DeleteClient            = m_lib.GetSymbol< decltype(impact_DeleteClient           )
    >("impact_DeleteClient"           );
                ProcessJson             = m_lib.GetSymbol< decltype(impact_ProcessJson            )
    >("impact_ProcessJson"            );
                FreeJsonAnswer          = m_lib.GetSymbol< decltype(impact_FreeJsonAnswer         )
    >("impact_FreeJsonAnswer"         );
                SaveState               = m_lib.GetSymbol< decltype(impact_SaveState              )
    >("impact_SaveState"              );
#if IMPACT_HAS_WEBSERVER
                Webserver_tcp           = m_lib.GetSymbol< decltype(impact_Webserver_tcp          )
    >("impact_Webserver_tcp"          );
                Webserver_tcp2          = m_lib.GetSymbol< decltype(impact_Webserver_tcp2         )
    >("impact_Webserver_tcp2"         );
                Webserver               = m_lib.GetSymbol< decltype(impact_Webserver              )
    >("impact_Webserver"              );
                Webserver_Stop          = m_lib.GetSymbol< decltype(impact_Webserver_Stop         )
    >("impact_Webserver_Stop"         );
                Webserver_Status        = m_lib.GetSymbol< decltype(impact_Webserver_Status       )
    >("impact_Webserver_Status"       );
#endif // IMPACT_HAS_WEBSERVER

                // added 2023-03-22
                StereoToMono_Planar     = m_lib.GetSymbol< decltype(impact_StereoToMono_Planar    )
    >("impact_StereoToMono_Planar"    );
                MonoToStereo_Planar     = m_lib.GetSymbol< decltype(impact_MonoToStereo_Planar    )
    >("impact_MonoToStereo_Planar"    );

```

```
            } catch (std::runtime_error& ) {
                return false;
            }
            // C functions allowed to be missing
            try {
                // added 2023-05-22
                SetMetadata             = m_lib.GetSymbol< decltype(impact_SetMetadata            ) >("impact_SetMetadata"            );
                GetMetadataInfos        = m_lib.GetSymbol< decltype(impact_GetMetadataInfos       ) >("impact_GetMetadataInfos"       );
                FreeMetadataInfos       = m_lib.GetSymbol< decltype(impact_FreeMetadataInfos      ) >("impact_FreeMetadataInfos"      );
                // added 2023-06-20
                SetPresetManager            = m_lib.GetSymbol< decltype(impact_SetPresetManager            ) >("impact_SetPresetManager"              );
                PresetManager_InformChange  = m_lib.GetSymbol< decltype(impact_PresetManager_InformChange     ) >("impact_PresetManager_InformChange"    );

                // added 2023-09-21
                GetPossibleChunkSizeInFrames = m_lib.GetSymbol< decltype(impact_GetPossibleChunkSizeInFrames  ) >("impact_GetPossibleChunkSizeInFrames"  );
                GetProcessChunkFrames        = m_lib.GetSymbol< decltype(impact_GetProcessChunkFrames          ) >("impact_GetProcessChunkFrames"         );
                InitProcess3                 = m_lib.GetSymbol< decltype(impact_InitProcess3                   ) >("impact_InitProcess3"                  );

#if IMPACT_HAS_CLOUDBUS
                // added 2023-06-26
                NewBus                  = m_lib.GetSymbol< decltype(impact_NewBus                 ) >("impact_NewBus"                 );
                FreeBus                 = m_lib.GetSymbol< decltype(impact_FreeBus                ) >("impact_FreeBus"                );
                SetInstanceBus          = m_lib.GetSymbol< decltype(impact_SetInstanceBus         ) >("impact_SetInstanceBus"         );
#endif // IMPACT_HAS_CLOUDBUS

                // added 2024-02-19
                SetMetadataMulti        = m_lib.GetSymbol< decltype(impact_SetMetadataMulti       ) >("impact_SetMetadataMulti"       );

            }  catch (std::runtime_error& ) {
                // Ignored, handler will take care
            }

            m_bOK=true;
            return true;
        }
        bool IsOk() const {
            return m_bOK;
        }

    };



    static constexpr const char *process_name = "SOUND4 IMPACT.CL";
    static constexpr const char *process_shortname = "impact";
    using SampleFormat = impact_SampleFormat;
    const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
    const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
    const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
    const size_t InputSampleSize=ChunkFrames*InputChannels;
    const size_t OutputSampleSize=ChunkFrames*OutputChannels;
    const size_t SampleRate=IMPACT_SAMPLE_RATE;

    namespace helper {
        #ifdef _WIN32
        static inline std::string WStringToUTF8(const std::wstring& wstr) {
            if (wstr.empty()) return {};
            int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
            std::string str(size, 0);
            WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
            return str;
        }
        static inline std::wstring UTF8ToWString(const std::string& str) {
            if (str.empty()) return std::wstring();
            int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
            std::wstring wstr(size, 0);
            MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
            return wstr;
        }
        #endif // _WIN32

            template<typename T>
        struct SampleFormat {
        };
```

```
            template<>
        struct SampleFormat<int16_t> {
            const impact_SampleFormat format = S16_NATIVE;
        };
            template<>
        struct SampleFormat<int32_t> {
            const impact_SampleFormat format = S32_NATIVE;
        };
            template<>
        struct SampleFormat<float> {
            const impact_SampleFormat format = F32_NATIVE;
        };


        static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output, size_t nSpl, impact_SampleFormat fmt) {
            dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
        }
        static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload, size_t nSpl, impact_SampleFormat fmt) {
            dynlib.AudioConvertTo(input, payload, nSpl, fmt);
        }
        static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t nSpl) {
            dynlib.StereoToMono(input, output, nSpl);
        }
        static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t nSpl) {
            dynlib.MonoToStereo(input, output, nSpl);
        }
        static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float *inputR, float *output, size_t nSpl) {
            dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
        }
        static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL, float *outputR, size_t nSpl) {
            dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
        }
        static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float *output) {
            dynlib.AudioMonoFromLiveStereo(payload, output);
        }
        static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t *payload) {
            dynlib.AudioMonoToLiveStereo(input, payload);
        }

    }; // namespace helper

    static inline std::string GetVersion(CDynLib& dynlib)
        {
            return std::string(dynlib.GetVersion());
        }

    static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
        {
            return dynlib.GetChunkSizeInFrames();
        }

    static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
        {
            std::vector<unsigned int> list;
            if (dynlib.GetPossibleChunkSizeInFrames) {
                for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
                    list.push_back(*src);
                }
            } else {
                list.push_back(64);
            }
            return list;
        }

    static inline unsigned int GetChannelCount(CDynLib& dynlib)
        {
            return dynlib.GetChannelCount();
        }

    static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
        {
            return dynlib.GetAudioInputCount();
        }

    static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
        {
            return dynlib.GetAudioOutputCount();
        }

```

```
    static inline unsigned int GetSampleRate(CDynLib& dynlib)
        {
            return dynlib.GetSampleRate();
        }

    static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
    {
        // Do some sanity checks
        if (dynlib.GetSampleRate()!=IMPACT_SAMPLE_RATE) {
            throw std::runtime_error("Bad library sampling rate");
        }
        if (a_bCheckFrames && dynlib.GetChunkSizeInFrames()!=ChunkFrames) {
            throw std::runtime_error("Bad library frame size");
        }
        if (dynlib.GetChannelCount()!=IMPACT_CHANNEL_COUNT) {
            throw std::runtime_error("Bad library channel count");
        }
        if (dynlib.GetAudioInputCount()!=IMPACT_AUDIO_INPUT_COUNT) {
            throw std::runtime_error("Bad library input count");
        }
        if (dynlib.GetAudioOutputCount()!=IMPACT_AUDIO_OUTPUT_COUNT) {
            throw std::runtime_error("Bad library output count");
        }
    }

    static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
        {
            return std::string(dynlib.GetFormatName(fmt));
        }

    static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
        {
            return dynlib.GetFormatFromName(name.c_str());
        }

    static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
        {
            return dynlib.GetBytesFromFormat(fmt);
        }

    using LogSeverity = impact_LogSeverity;
    using log_cb_t = std::function<void(LogSeverity,const std::string&)>;

    static log_cb_t _log_cb;
    extern "C" {
        static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
            _log_cb(severity, std::string(c_msg));
        }
    }
    static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
        {
            dynlib.SetLogSeverity(severity);
        }
    static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
        {
            _log_cb=cb;
            dynlib.SetLoggerCallback(&_log_cb_c);
        }

#if IMPACT_HAS_CLOUDBUS
    class CBus {
        public:
            CBus(CDynLib& dynlib)
                : m_dynlib(dynlib)
                {
                    if (m_dynlib.NewBus) {
                        m_bus=m_dynlib.NewBus();
                    }
                }
            ~CBus()
                {
                    if (m_bus && m_dynlib.FreeBus) {
                        m_dynlib.FreeBus(m_bus);
                    }
                    m_bus = nullptr;
                }
            impact_CBus *Get() const { return m_bus; }
        private:
            CDynLib& m_dynlib;
            impact_CBus *m_bus;
    };
#endif // IMPACT_HAS_CLOUDBUS

    class CPresetLoader {
    public:
        CPresetLoader() = default;
        virtual ~CPresetLoader() = default;
```

```
        virtual bool IsReadOnly() = 0;
        virtual bool Exists(const std::filesystem::path &name) = 0;
        virtual bool Remove(const std::filesystem::path &name) = 0;
        virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
        virtual std::vector<std::filesystem::path> GetAll() = 0;
        virtual std::string Read(const std::filesystem::path &filename) = 0;
        virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
    };
    using PresetChange_Kind = impact_PresetChange_Kind;

    class CInstance {
    public:
        CInstance(CDynLib& dynlib)
        : m_dynlib(dynlib)
            {
                params = m_dynlib.NewParameters();
                if (!params) throw std::bad_alloc();
            }
        // non-copyable
        CInstance( const CInstance& ) = delete;
        CInstance& operator=( const CInstance& ) = delete;
        ~CInstance()
            {
                Stop();
                if (params) m_dynlib.FreeParameters(params);
            }
       bool IsOk() const { return instance!=nullptr; }
        void SetParam(const std::string& name, const std::string& value)
            {
                assert(params);
                m_dynlib.SetParameter(params, name.c_str(), value.c_str());
            }
        std::string GetParam(const std::string& name)
            {
                assert(params);
                auto c_value=m_dynlib.GetParameter(params, name.c_str());
                std::string ret(c_value);
                m_dynlib.FreeParameterValue(c_value);
                return ret;
            }
#if IMPACT_HAS_CLOUDBUS
            void SetBus(const CBus& bus) {
                assert(params);
                if (m_dynlib.SetInstanceBus) {
                    m_dynlib.SetInstanceBus(params,bus.Get());
                }
            }
#endif // IMPACT_HAS_CLOUDBUS

        void SetPresetManager(CPresetLoader *preset_manager)
            {
                if (!preset_manager) return;
                if (!m_dynlib.SetPresetManager) return;
                assert(params);
                m_dynlib.SetPresetManager(params,
                    impact_custom_reader,
                    impact_custom_reader_free,
                    impact_custom_writer,
                    impact_custom_exists,
                    impact_custom_getall,
                    impact_custom_getall_free,
                    impact_custom_remove,
                    impact_custom_rename,
                    preset_manager->IsReadOnly(),
                    preset_manager
                );
            }

        bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
    Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
    json_port = 0, unsigned int frames_per_chunk=64)
            {
                assert(!instance);
                std::string l_save_path;
                if (!save_path.empty()) {
                    l_save_path = save_path.u8string();
                }
                if (m_dynlib.InitProcess3) {
                    instance = m_dynlib.InitProcess3(LoginKey.c_str(), RadioName.c_str(),
    Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
                } else if (frames_per_chunk!=64) {
                    return false;
                } else {
                    instance = m_dynlib.InitProcess2(LoginKey.c_str(), RadioName.c_str(),
    Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
                }
```

```
            if (!instance) return false;
            if (!init_metadata.empty()) {
                for (auto&& [key,value]: init_metadata) {
                    m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
                }
                init_metadata.clear();
            }
            update_thread = std::thread([this, json_port](){
                OnUpdateThreadStart();
                m_dynlib.StartUpdateThread(instance, json_port);
                OnUpdateThreadStop();
            });
            int timeout=5000;
            #ifdef DEBUG
                timeout=60*1000; // For debugger pauses
            #endif
            if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
                return false;
            }
            return true;
        }

    unsigned int GetChunkFrames() {
        if (instance && m_dynlib.GetProcessChunkFrames) {
            return m_dynlib.GetProcessChunkFrames(instance);
        }
        return 64;
    }

    void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind change_kind)
        {
            if (m_dynlib.PresetManager_InformChange) {
                m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(), (impact_PresetChange_Kind) change_kind);
            }
        }

    void SetMetadata(const std::string& key, const char* value)
        {
            if (m_dynlib.SetMetadata) {
                if (instance) {
                    m_dynlib.SetMetadata(instance, key.c_str(), value);
                } else {
                    init_metadata.push_back( {key, value} );
                }
            }
        }


    void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
        {
            if (instance) {
                if (m_dynlib.SetMetadataMulti) {
                    typedef const char* pchar;
                    pchar* keyvalue=new pchar[2*list.size()+1];
                    size_t n=0;
                    for (auto&& [key,value]: list) {
                        keyvalue[2*n+0]=key.c_str();
                        keyvalue[2*n+1]=value;
                        n++;
                    }
                    keyvalue[2*n+0]=nullptr;

                    impact_SetMetadataMulti(instance, keyvalue);

                    delete[] keyvalue;
                } else {
                    for (auto&& [key,value]: list) {
                        SetMetadata(key, value);
                    }
                }
            } else {
                for (auto&& [key,value]: list) {
                    if (value) {
                        init_metadata.push_back( {key, value} );
                    }
                }
            }
        }

    std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
        {
            std::vector< std::tuple<std::string,std::string> > values;
            if (m_dynlib.GetMetadataInfos) {
                const char** c_values = m_dynlib.GetMetadataInfos(instance);
                if (c_values) {
```

```
                        for (const char** c_value=c_values; *c_value; ) {
                            std::string key(*c_value);
                            c_value++;
                            if (*c_value) {
                                std::string descr(*c_value);
                                c_value++;
                                values.push_back( {key,descr} );
                            }
                        }
                        m_dynlib.FreeMetadataInfos(instance, c_values);
                    }
                }
                return values;
            }

        // Can be overloaded, called at start of UpdateThread in the thread
        virtual void OnUpdateThreadStart() {}
        // Can be overloaded to call at end of UpdateThread in the thread
        virtual void OnUpdateThreadStop() {}

#if IMPACT_HAS_WEBSERVER
        bool StartWebServer(int http_port, int https_port=0)
            {
                assert(instance);
                assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
                webserver = m_dynlib.Webserver(http_port,https_port,instance);
                if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
                    return false;
                }
                int web_status=m_dynlib.Webserver_Status(webserver);
                if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
                    return false;
                }
                if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
                    return false;
                }
                return true;
            }
        void StopWebServer()
            {
                if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
                    m_dynlib.Webserver_Stop(webserver, 1000);
                    webserver = SOUND4_INVALID_WEBSERVER_ID;
                }
            }
#endif // IMPACT_HAS_WEBSERVER

        int Terminate()
            {
                if (!instance) return 0;
                return m_dynlib.TerminateProcess(instance);
            }

        void Stop()
            {
                if (!instance) return;
#if IMPACT_HAS_WEBSERVER
                StopWebServer();
#endif // IMPACT_HAS_WEBSERVER
                if (update_thread.joinable()) {
                    m_dynlib.StopUpdateThread(instance); // this will exit the update thread
                    update_thread.join();
                }
                m_dynlib.ExitProcess(instance);
                instance=nullptr;
            }

        unsigned int GetEstimatedDelay()
            {
                assert(instance);
                return m_dynlib.GetEstimatedDelay(instance);
            }

        std::array<float,InputSampleSize>& GetBufferIn()
            {
                assert(instance);
                float *buf=m_dynlib.GetBufferIn(instance);
                return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);

            }
        std::array<float,OutputSampleSize>& GetBufferOut()
            {
                assert(instance);
                float *buf= m_dynlib.GetBufferOut(instance);
                return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
            }
        void ProcessAudio()
```

```
            {
                assert(instance);
                m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance), m_dynlib.GetBufferOut(instance));
            }
        // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>, InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
        void ProcessAudio_Planar(float const * const *input, float * const *output)
            {
                assert(instance);
                m_dynlib.ProcessAudio_Planar(instance, input, output);
            }

            template<typename T>
        std::vector<T> ProcessAnyAudio(const std::vector<T> input)
            {
                assert(instance);
                std::vector<T> output;
                unsigned int out_offset=0;
                unsigned int in_offset=0;
                unsigned int todo = input.size();
                while (todo>0) {
                    unsigned int left = AddAudio(&input[in_offset], todo);
                    unsigned int out_avail = m_dynlib.GetOutputCount(instance);
                    output.resize(out_offset + out_avail);
                    GetAudio(&output[out_offset], out_avail);
                    out_offset+=out_avail;
                    in_offset += todo-left;
                    todo=left;
                }
                return output;
            }

        class CClient {
            public:
                CClient(impact_CInstance* instance, CDynLib& dynlib)
                : m_dynlib(dynlib)
                    {
                        assert(instance);
                        client=m_dynlib.NewClient(instance);
                        if (!client) throw std::bad_alloc();
                    }
                // non-copyable
                CClient( const CInstance& ) = delete;
                CClient& operator=( const CInstance& ) = delete;
                ~CClient()
                    {
                        if (client) {
                            m_dynlib.DeleteClient(client);
                            client=nullptr;
                        }
                    }
                std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
                    {
                        assert(client);
                        int need_save=0;
                        const char *canswer = m_dynlib.ProcessJson(client, request.c_str(), &need_save);
                        if (!canswer) return {};
                        std::string answer(canswer);
                        m_dynlib.FreeJsonAnswer (canswer);
                        if (NeedSave) {
                            *NeedSave=(need_save!=0);
                        }
                        return answer;
                    }
            private:
                CDynLib& m_dynlib;
                impact_CClientInstance *client = nullptr;
        };
        std::shared_ptr<CClient> NewClient()
            {
                assert(instance);
                return std::make_shared<CClient>(instance, m_dynlib);
            }

    protected:
            template<typename T>
        unsigned int AddAudio(const T* payload, unsigned int nFrame)
            {
                assert(instance);
                return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame, helper::SampleFormat<T>::format);
            }
            template<typename T>
        unsigned int GetAudio(T* payload, unsigned int max_nFrame)
            {
```

```
                assert(instance);
                return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame, helper::SampleFormat<T>::format);
            }

    private:
        CDynLib& m_dynlib;
        impact_CParameters* params = nullptr;
        impact_CInstance* instance = nullptr;
        std::thread update_thread;
        std::vector< std::pair<std::string,std::string> > init_metadata;
#if IMPACT_HAS_WEBSERVER
        uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
#endif // IMPACT_HAS_WEBSERVER
    };

}; // namespace dyn
}; // namespace impact
}; // namespace sound4


// bridge C callbacks to CPresetLoader
extern "C" {
    static char *impact_custom_reader(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        std::string content=preset_loader->Read(filename);
        return strdup(content.c_str());
    }
    static void impact_custom_reader_free(char *content, void* userdata) {
        if (!content) return;
        free(content);
    }
    static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Write(filename,content);
        return res?0:-1;
    }
    static int impact_custom_exists(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Exists(filename);
        return res?0:-1;
    }
    static fs_char** impact_custom_getall(void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->GetAll();
        fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
        for (size_t n=0;n<res.size();n++) {
            all[n]=fs_strdup(res[n].c_str());
        }
        all[res.size()]=nullptr;
        return all;
    }
    static void impact_custom_getall_free(fs_char** all, void* userdata) {
        if (!all) return;
        for (fs_char** one=all;*one;one++) {
            free(*one);
        }
        free(all);
    }
    static int impact_custom_remove(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Remove(filename);
        return res?0:-1;
    }
    static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Rename(from,to);
        return res?0:-1;
    }

};
```

## 11.13 sound4cl-pipe.cpp File Reference

C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <io.h>
#include <fcntl.h>
#include "sound4cl.hpp"
```

## Classes

- class MyPresetManager

  *[Custom preset manager]*

## Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::CInstance &instance, sound4::CProcessor &processor)
- int main (int argc, const char **argv)

  *[Custom preset manager]*

## 11.13.1 Function Documentation

### 11.13.1.1 GetEnv()

```
std::string GetEnv (
            const std::string & name )
```

Definition at line 60 of file sound4cl-pipe.cpp.

### 11.13.1.2 main()

```
int main (
            int argc,
            const char ** argv )
```

[Dynamic loading config]

[Dynamic loading config]

[Dynamic loading]

[Dynamic loading]

[Setup and configure]

[Setup and configure]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Starting a web-server]

[Starting a web-server]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 178 of file sound4cl-pipe.cpp.

#### 11.13.1.3 ProcessLoop()

```
void ProcessLoop (
            sound4::CInstance & instance,
            sound4::CProcessor & processor )
```

[Processing Loop]

[Processing Loop]

Definition at line 69 of file sound4cl-pipe.cpp.

## 11.14 sound4cl-pipe.cpp

```
00001
00006 #include <unistd.h>
00007 #include <iostream>
00008 #include <strings.h>
00009 #include <sys/types.h>
00010 #include <filesystem>
00011 #include <fstream>
00012 #include <sstream>
00013 #if defined(_WIN32) || defined(WIN32)
00014     #include <io.h>
00015     #include <fcntl.h>
00016 #endif
00017
00018 #include "sound4cl.hpp"
00019
00020 // ------------------------------------------------------------
00060 std::string GetEnv(const std::string& name)
00061 {
00062     char *data=getenv(name.c_str());
00063     if (!data) return {};
00064     return std::string(data);
00065 }
00066
00067
00068 // Reads from stdin, write to stdout
00069 void ProcessLoop(sound4::CInstance &instance, sound4::CProcessor &processor)
```

```
{
bool stopthread=false;

    #if defined(_WIN32) || defined(WIN32)
        // Windows: The stdin, stdout, and stderr streams always open in text mode by default
        _setmode(_fileno(stdin), _O_BINARY);
        _setmode(_fileno(stdout), _O_BINARY);
    #endif

    size_t input_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
    size_t output_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
    while (!stopthread)
    {
        // Read a chunk from stdin
        if (std::fread(instance.GetBufferIn(), sizeof(float), input_sample_count, stdin) !=
 input_sample_count) {
            std::cerr « "ProcessLoop: input stopped"« std::endl;
            stopthread=true;
            break;
        }
        if (stopthread) break;
        // Process it
        instance.ProcessAudio();
        // Write it to stdout
        if (fwrite(instance.GetBufferOut(), sizeof(float), output_sample_count, stdout) !=
 output_sample_count) {
            std::cerr « "Failed to write audio output" « std::endl;
            stopthread=true;
        }
    }
}

class MyPresetManager : public sound4::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;

    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
 std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(entry.path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
```

```
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
            if ( f.fail() ) {
                return false;
            }
            return true;
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};

int main(int argc, const char **argv)
{
    int listen_port = 8080;
    auto store_path_str=GetEnv("S4STORE_PATH");
    std::string libname =  "libsound4.impact.cl.so";


    // Get credentials from environment
    auto LoginKey=GetEnv("S4LOGINKEY");
    if (LoginKey.empty()) {
        LoginKey=GetEnv("S4USERID");
    }
    auto RadioName=GetEnv("RADIO_NAME");
    auto KeyId=GetEnv("ACCESS_KEY_ID");
    auto AccessKey=GetEnv("ACCESS_KEY_SECRET");

    const char *progname=argv[0];
    for (int n=1;n<argc;n++) {
        std::string arg = argv[n];
        if (arg=="-p" || arg=="--port") {
            if (argc<n+1) {
                std::cerr « "--port needs a number"« std::endl;
                return 1;
            }
            n++;
            listen_port=atoi(argv[n]);
        } else if (arg=="-l" || arg=="--lib") {
            if (argc<n+1) {
                std::cerr « "--port needs a number"« std::endl;
                return 1;
            }
            n++;
            libname=argv[n];
        } else if (arg=="-h" || arg=="--help") {
            std::cerr « "Usage: " « progname « " [-p port] [-l libname.so]"«std::endl;;
            std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
            std::cerr « "   -l, --lib   : library to load ["«libname«"]" « std::endl;
            std::cerr « "   -p, --port  : port for web server ["«listen_port«"]" « std::endl;
            std::cerr « std::endl;
            std::cerr « "Environment:"« std::endl;;
            std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"« std::endl;;
            std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
            std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
            std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
            std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;
            std::cerr « std::endl;
            std::cerr « "Example:"« std::endl;
            std::cerr « "  ffmpeg -re -i file.wav -ar 48000 -ac 2 -f f32le - | "«progname«" | ffplay -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
            #if defined(_WIN32) || defined(WIN32)
            std::cerr « "  ffmpeg -f dshow -i audio=\"Microphone\" -ar 48000 -ac 2 -f f32le - | "«progname«" | ffplay -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
            #else
            std::cerr « "  ffmpeg -f pulse -i default -ar 48000 -ac 2 -f f32le - | "«progname«" | ffplay -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
            #endif
            return 0;
        } else {
            std::cerr « "unknown argument '"«arg«"'"« std::endl;
            return 1;
        }
    }


    sound4::CProcessor processor;
    if (!processor.Load(libname)) {
```

```
        std::cerr « "Failed to load processing library" « std::endl;
        return 1;
    }

    if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
        std::cerr « "Missing credentials in environment" « std::endl;
        return 1;
    }
    std::filesystem::path store_path;
    if (!store_path_str.empty()) {
        store_path = std::filesystem::path(store_path_str);
        if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
            std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
std::endl;
            return 1;
        }
    }

    processor.SetLogSeverity(sound4::verbose5);
    processor.SetLoggerCallback([](sound4::LogSeverity severity,const std::string& msg) {
            std::cerr « msg « std::endl;
        });

    sound4::CInstance instance(processor);
    instance.SetParam("ADMIN_USER","admin");
    instance.SetParam("ADMIN_SECRET","admin");

    MyPresetManager preset_manager;
    if (!store_path.empty()) {
        if (!preset_manager.SetupPath(store_path)) {
            std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
std::endl;
            return 1;
        }
        instance.SetPresetManager(&preset_manager);
    }

    std::cerr « "Creating" « std::endl;
    if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
        std::cerr « "Failed to create the processing instance" « std::endl;
        return 1;
    }

    std::cerr « "Starting web server" « std::endl;
    if (!instance.StartWebServer(listen_port,0)) {
        std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
        return 1;
    }

    {
        auto client = instance.NewClient();
        auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
        std::cerr « "Request for processor name returned "«answer«std::endl;
    }

    ProcessLoop(instance, processor);

    std::cerr « "Stopping now" « std::endl;
    instance.Stop();

    std::cerr « "End" « std::endl;
    return 0;
}
```

## 11.15 sound4cl.hpp File Reference

C++ wrapper for dynamic loading library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
```

```
#include <filesystem>
#include <string.h>
#include <libloaderapi.h>
#include "sound4cl_cdef.h"
```

## Classes

- struct sound4::helper::SampleFormat< T >

  *Helper for Sample format types.*
- struct sound4::helper::SampleFormat< int16_t >
- struct sound4::helper::SampleFormat< int32_t >
- struct sound4::helper::SampleFormat< float >
- class sound4::helper::DynFuncHolder< T >

  *Helper to wrap a function pointer.*
- class sound4::helper::CDynLoader

  *Helper to load dynamic library.*
- class sound4::CProcessor

  *Dynamic library interface.*
- class sound4::CBus

  *Shared Bus.*
- class sound4::CPresetLoader

  *Custom preset handler helper.*
- class sound4::CInstance

  *Instance handling class.*
- class sound4::CInstance::CClient

## Namespaces

- sound4
- sound4::helper

## Typedefs

- using sound4::helper::DynLib_t = HMODULE
- using sound4::helper::DynFunc_t = FARPROC
- using sound4::LogSeverity = sound4cl_LogSeverity
- using sound4::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::PresetChange_Kind = sound4cl_PresetChange_Kind

## Variables

- const LogSeverity sound4::none = (LogSeverity) sound4cl_none
- const LogSeverity sound4::fatal = (LogSeverity) sound4cl_fatal
- const LogSeverity sound4::error = (LogSeverity) sound4cl_error
- const LogSeverity sound4::warning = (LogSeverity) sound4cl_warning
- const LogSeverity sound4::info = (LogSeverity) sound4cl_info
- const LogSeverity sound4::verbose = (LogSeverity) sound4cl_verbose
- const LogSeverity sound4::verbose2 = (LogSeverity) sound4cl_verbose2
- const LogSeverity sound4::verbose3 = (LogSeverity) sound4cl_verbose3
- const LogSeverity sound4::verbose4 = (LogSeverity) sound4cl_verbose4
- const LogSeverity sound4::verbose5 = (LogSeverity) sound4cl_verbose5

### 11.15.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4cl.hpp.

## 11.16 sound4cl.hpp

```
00001
00008 #pragma once
00009
00010 #include <string>
00011 #include <exception>
00012 #include <thread>
00013 #include <functional>
00014 #include <memory>
00015 #include <array>
00016 #include <assert.h>
00017 #include <stdexcept>
00018 #include <filesystem>
00019 #include <string.h>
00020
00021 #ifdef __unix__
00022     #include <dlfcn.h>
00023 #elif defined(_WIN32)
00024     #include <libloaderapi.h>
00025 #else
00026     #error "Unsupported OS"
00027 #endif
00028
00029 #include "sound4cl_cdef.h"
00030
00031     // bridge C callbacks to CPresetLoader
00032     extern "C" {
00033         static char *sound4cl_custom_reader(const fs_char *filename, void* userdata);
00034         static void sound4cl_custom_reader_free(char *content, void* userdata);
00035         static int sound4cl_custom_writer(const fs_char *filename, const char *content, void*
      userdata);
00036         static int sound4cl_custom_exists(const fs_char *filename, void* userdata);
00037         static fs_char** sound4cl_custom_getall(void* userdata);
00038         static void sound4cl_custom_getall_free(fs_char** all, void* userdata);
00039         static int sound4cl_custom_remove(const fs_char *filename, void* userdata);
00040         static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041     };
00042
00046 namespace sound4 {
00047
00051     namespace helper {
00052         #ifdef _WIN32
00058         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00059             if (wstr.empty()) return {};
00060             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
      NULL);
00061             std::string str(size, 0);
00062             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00063             return str;
00064         }
00070         static inline std::wstring UTF8ToWString(const std::string& str) {
00071             if (str.empty()) return std::wstring();
00072             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00073             std::wstring wstr(size, 0);
00074             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00075             return wstr;
00076         }
00077         #endif // _WIN32
00078
00084             template<typename T>
00085         struct SampleFormat {
00086         };
00087             template<>
00088         struct SampleFormat<int16_t> {
00089             const sound4cl_SampleFormat format = S16_NATIVE;
00090         };
00091             template<>
00092         struct SampleFormat<int32_t> {
00093             const sound4cl_SampleFormat format = S32_NATIVE;
00094         };
00095             template<>
00096         struct SampleFormat<float> {
00097             const sound4cl_SampleFormat format = F32_NATIVE;
```

```
        };

        // Storage definition, depending on OS
        #ifdef __unix__
            using DynLib_t = void *;
            using DynFunc_t = void *;
        #else
            using DynLib_t = HMODULE;
            using DynFunc_t = FARPROC;
        #endif

            template <typename T>
        class DynFuncHolder {
        private:
            DynFunc_t m_ptr = NULL;
        public:
            DynFuncHolder() = default;
            DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
            operator bool() const { return (m_ptr!=NULL); }
            bool IsOk() const { return m_ptr!=NULL; }
            operator T* () const { return reinterpret_cast<T*>(m_ptr); }
        };

        class CDynLoader {
            private:
                DynLib_t m_lib {};
            public:
                CDynLoader() = default;
                CDynLoader(CDynLoader&&) = default;
                // No copy allowed
                CDynLoader(const CDynLoader&) = delete;
                CDynLoader& operator=(CDynLoader const&) = delete;
                operator bool() const {
                    return (m_lib!=NULL);
                }
                bool IsOk() const {
                    return (m_lib!=NULL);
                }
                #ifdef __unix__
                    void Close() {
                        if (IsOk()) {
                            #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
                            dlclose(m_lib);
                            #endif // __SANITIZE_ADDRESS__
                            m_lib=NULL;
                        }
                    }
                    bool Load(const std::filesystem::path& dynlib)
                    {
                        Close();
                        m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
                        return IsOk();
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
                        }
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static Dl_info info;
                        if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
                        auto rp=realpath(info.dli_fname, NULL);
                        std::filesystem::path p(rp);
                        free(rp);
                        return p;
                    }
                #else
                    bool Load(const std::filesystem::path& dynlib, uint32_t loadflags =
LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_
                    {
                        Close();
                        m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
                        return IsOk();
                    }
                    void Close() {
                        if (IsOk()) {
                            // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
FreeLibraryAndExitThread but...
                            // So define SOUND4_CALL_FREELIBRARY before including this file if you
```

```
really want to unload the library dynamically
                            #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
                                FreeLibraryAndExitThread(m_lib, 0);
                            #elif defined(SOUND4_CALL_FREELIBRARY)
                                FreeLibrary(m_lib);
                            #endif // SOUND4_CALL_FREELIBRARY
                            m_lib=NULL;
                        }
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
                        }
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static wchar_t path[MAX_PATH]={};
                        HMODULE hm = NULL;
                        if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
                                GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
                                (LPCWSTR) &GetThisLibraryPath, &hm))
                        {
                            if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
                                return std::filesystem::path(path);
                            }
                        }
                        return {};
                    }
                #endif
                virtual ~CDynLoader() {
                    Close();
                }
        }; // class CDynLoader
    }; // namespace helper


    using LogSeverity = sound4cl_LogSeverity;
    const LogSeverity none     = (LogSeverity) sound4cl_none;
    const LogSeverity fatal    = (LogSeverity) sound4cl_fatal;
    const LogSeverity error    = (LogSeverity) sound4cl_error;
    const LogSeverity warning  = (LogSeverity) sound4cl_warning;
    const LogSeverity info     = (LogSeverity) sound4cl_info;
    const LogSeverity verbose  = (LogSeverity) sound4cl_verbose;
    const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2;
    const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3;
    const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4;
    const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5;

    using log_cb_t = std::function<void(LogSeverity,const std::string&)>;

    static log_cb_t _log_cb;
    extern "C" {
        static inline void _log_cb_c(LogSeverity severity, const char *c_msg) {
            _log_cb(severity, std::string(c_msg));
        }
    }

    class CProcessor {
    private:
        helper::CDynLoader m_lib;
        bool m_bOK = false;
        std::string m_prefix; // The prefix of functions, depends on the library (cloud_, impact_...)
    protected:
        friend class CBus;
        friend class CInstance;

        // All those wrapped functions have the same signature as there C sound4cl_XXX
        helper::DynFuncHolder< decltype(sound4cl_GetVersion             ) > fnGetVersion             ;
        helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames   ) > fnGetChunkSizeInFrames   ;
        helper::DynFuncHolder< decltype(sound4cl_GetChannelCount        ) > fnGetChannelCount        ;
        helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount     ) > fnGetAudioInputCount     ;
        helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount    ) > fnGetAudioOutputCount    ;
        helper::DynFuncHolder< decltype(sound4cl_GetSampleRate          ) > fnGetSampleRate          ;
        helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback      ) > fnSetLoggerCallback      ;
        helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity         ) > fnSetLogSeverity         ;
        helper::DynFuncHolder< decltype(sound4cl_NewParameters          ) > fnNewParameters          ;
        helper::DynFuncHolder< decltype(sound4cl_FreeParameters         ) > fnFreeParameters         ;
        helper::DynFuncHolder< decltype(sound4cl_SetParameter           ) > fnSetParameter           ;
        helper::DynFuncHolder< decltype(sound4cl_GetParameter           ) > fnGetParameter           ;
        helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue     ) > fnFreeParameterValue     ;
```

```
        helper::DynFuncHolder< decltype(sound4cl_InitProcess            ) > fnInitProcess            ;
        helper::DynFuncHolder< decltype(sound4cl_InitProcess2           ) > fnInitProcess2           ;
        helper::DynFuncHolder< decltype(sound4cl_TerminateProcess       ) > fnTerminateProcess       ;
        helper::DynFuncHolder< decltype(sound4cl_ExitProcess            ) > fnExitProcess            ;
        helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread      ) > fnStartUpdateThread      ;
        helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread       ) > fnStopUpdateThread       ;
        helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady  ) > fnWaitUpdateThreadReady  ;
        helper::DynFuncHolder< decltype(sound4cl_ProcessAudio           ) > fnProcessAudio           ;
        helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar    ) > fnProcessAudio_Planar    ;
        helper::DynFuncHolder< decltype(sound4cl_GetBufferIn            ) > fnGetBufferIn            ;
        helper::DynFuncHolder< decltype(sound4cl_GetBufferOut           ) > fnGetBufferOut           ;
        helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay      ) > fnGetEstimatedDelay      ;
        helper::DynFuncHolder< decltype(sound4cl_GetFormatName          ) > fnGetFormatName          ;
        helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName      ) > fnGetFormatFromName      ;
        helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat     ) > fnGetBytesFromFormat     ;
        helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame      ) > fnGetMaxPacketFrame      ;
        helper::DynFuncHolder< decltype(sound4cl_AddAudio               ) > fnAddAudio               ;
        helper::DynFuncHolder< decltype(sound4cl_AddPadAudio            ) > fnAddPadAudio            ;
        helper::DynFuncHolder< decltype(sound4cl_GetOutputCount         ) > fnGetOutputCount         ;
        helper::DynFuncHolder< decltype(sound4cl_GetAudio               ) > fnGetAudio               ;
        helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom       ) > fnAudioConvertFrom       ;
        helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo         ) > fnAudioConvertTo         ;
        helper::DynFuncHolder< decltype(sound4cl_StereoToMono           ) > fnStereoToMono           ;
        helper::DynFuncHolder< decltype(sound4cl_MonoToStereo           ) > fnMonoToStereo           ;
        helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > fnAudioMonoFromLiveStereo;
        helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo  ) > fnAudioMonoToLiveStereo  ;
        helper::DynFuncHolder< decltype(sound4cl_NewClient              ) > fnNewClient              ;
        helper::DynFuncHolder< decltype(sound4cl_DeleteClient           ) > fnDeleteClient           ;
        helper::DynFuncHolder< decltype(sound4cl_ProcessJson            ) > fnProcessJson            ;
        helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer         ) > fnFreeJsonAnswer         ;
        helper::DynFuncHolder< decltype(sound4cl_SaveState              ) > fnSaveState              ;

        // only for processes supporting webserver
        helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp          ) > fnWebserver_tcp          ;
        helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2         ) > fnWebserver_tcp2         ;
        helper::DynFuncHolder< decltype(sound4cl_Webserver              ) > fnWebserver              ;
        helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop         ) > fnWebserver_Stop         ;
        helper::DynFuncHolder< decltype(sound4cl_Webserver_Status       ) > fnWebserver_Status       ;

        // added 2023-03-22
        helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar    ) > fnStereoToMono_Planar    ;
        helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar    ) > fnMonoToStereo_Planar    ;

        // added 2023-05-22
        helper::DynFuncHolder< decltype(sound4cl_SetMetadata            ) > fnSetMetadata            ;
        helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos       ) > fnGetMetadataInfos       ;
        helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos      ) > fnFreeMetadataInfos      ;

        // added 2023-06-20
        helper::DynFuncHolder< decltype(sound4cl_SetPresetManager                ) > fnSetPresetManager      ;
        helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange      ) > fnPresetManager_InformChange      ;

        // added 2023-09-21
        helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames    ) > fnGetPossibleChunkSizeInFrames   ;
        helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames           ) > fnGetProcessChunkFrames   ;
        helper::DynFuncHolder< decltype(sound4cl_InitProcess3                    ) > fnInitProcess3
        ;

        // added 2023-06-26, only for processes supporting bus
        helper::DynFuncHolder< decltype(sound4cl_NewBus                 ) > fnNewBus      ;
        helper::DynFuncHolder< decltype(sound4cl_FreeBus                ) > fnFreeBus      ;
        helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus         ) > fnSetInstanceBus      ;

        // added 2024-02-19
        helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti       ) > fnSetMetadataMulti
    ;

            template <typename T>
        helper::DynFuncHolder<T> GetPrefixSymbol(const std::string& name) {
            std::string prefix_name = m_prefix+name;
            return m_lib.GetSymbol<T>(prefix_name);
        }

    public:
        CProcessor() = default;

        bool Load(const std::filesystem::path& filepath) {
            m_lib.Close();
            // If a path is given, use it directly and do not try other path
            if (filepath.has_parent_path() && !m_lib.Load(filepath)) {
                return false;
            } else if (!m_lib.IsOk()) {
                auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
```

```
                auto filename = filepath.filename();
                #ifndef _WIN32
                // Linux: if in a bin directory, try ../lib first
                if (!m_lib.IsOk() && thisdir.filename()=="bin") {
                    auto libdir = thisdir.parent_path() / "lib";
                    m_lib.Load(libdir / filename);
                }
                #endif // !_WIN32
                if (!m_lib.IsOk()) {
                    // Search in the same directory this code is in
                    m_lib.Load(thisdir / filename);
                }
                if (!m_lib.IsOk()) {
                    // Try current path
                    m_lib.Load(std::filesystem::current_path() / filename);
                }
                if (!m_lib.IsOk()) {
                    return false;
                }
            }
            // Get information from the library itself
            try {
                auto GetInfo = m_lib.GetSymbol< decltype(sound4cl_SOUND4_GetProcessInfo              )
    >("SOUND4_GetProcessInfo"   );
                auto info = GetInfo();
                m_prefix = info->prefix;
            } catch (std::runtime_error& ) {
                // Compatibility: search in known libraries without this function
                #ifdef _WIN32
                    std::string filename = helper::WStringToUTF8(filepath.stem());
                #else // !_WIN32
                    std::string filename = filepath.stem();
                    // remove lib prefix
                    filename=filename.substr(3);
                #endif // ! _WIN32
                if (filename=="sound4.x1.cloud") {
                    m_prefix="cloudx1_";
                } else if (filename=="sound4.impact.cl") {
                    m_prefix="impact_";
                } else if (filename=="sound4.bigvoice.cl") {
                    m_prefix="bigvoice_";
                } else {
                    return false;
                }
            }

            // Load all C functions
            try {
                fnGetVersion              = GetPrefixSymbol< decltype(sound4cl_GetVersion
    ) >("GetVersion"             );
                fnGetChunkSizeInFrames    = GetPrefixSymbol< decltype(sound4cl_GetChunkSizeInFrames
    ) >("GetChunkSizeInFrames"   );
                fnGetChannelCount         = GetPrefixSymbol< decltype(sound4cl_GetChannelCount
    ) >("GetChannelCount"        );
                fnGetAudioInputCount      = GetPrefixSymbol< decltype(sound4cl_GetAudioInputCount
    ) >("GetAudioInputCount"     );
                fnGetAudioOutputCount     = GetPrefixSymbol< decltype(sound4cl_GetAudioOutputCount
    ) >("GetAudioOutputCount"    );
                fnGetSampleRate           = GetPrefixSymbol< decltype(sound4cl_GetSampleRate
    ) >("GetSampleRate"          );
                fnSetLoggerCallback       = GetPrefixSymbol< decltype(sound4cl_SetLoggerCallback
    ) >("SetLoggerCallback"      );
                fnSetLogSeverity          = GetPrefixSymbol< decltype(sound4cl_SetLogSeverity
    ) >("SetLogSeverity"         );
                fnNewParameters           = GetPrefixSymbol< decltype(sound4cl_NewParameters
    ) >("NewParameters"          );
                fnFreeParameters          = GetPrefixSymbol< decltype(sound4cl_FreeParameters
    ) >("FreeParameters"         );
                fnSetParameter            = GetPrefixSymbol< decltype(sound4cl_SetParameter
    ) >("SetParameter"           );
                fnGetParameter            = GetPrefixSymbol< decltype(sound4cl_GetParameter
    ) >("GetParameter"           );
                fnFreeParameterValue      = GetPrefixSymbol< decltype(sound4cl_FreeParameterValue
    ) >("FreeParameterValue"     );
                fnInitProcess             = GetPrefixSymbol< decltype(sound4cl_InitProcess
    ) >("InitProcess"            );
                fnInitProcess2            = GetPrefixSymbol< decltype(sound4cl_InitProcess2
    ) >("InitProcess2"           );
                fnTerminateProcess        = GetPrefixSymbol< decltype(sound4cl_TerminateProcess
    ) >("TerminateProcess"       );
                fnExitProcess             = GetPrefixSymbol< decltype(sound4cl_ExitProcess
    ) >("ExitProcess"            );
                fnStartUpdateThread       = GetPrefixSymbol< decltype(sound4cl_StartUpdateThread
    ) >("StartUpdateThread"      );
                fnStopUpdateThread        = GetPrefixSymbol< decltype(sound4cl_StopUpdateThread
    ) >("StopUpdateThread"       );
                fnWaitUpdateThreadReady   = GetPrefixSymbol< decltype(sound4cl_WaitUpdateThreadReady
```

```
    ) >("WaitUpdateThreadReady"  );
00448                fnProcessAudio            = GetPrefixSymbol< decltype(sound4cl_ProcessAudio
    ) >("ProcessAudio"           );
00449                fnProcessAudio_Planar     = GetPrefixSymbol< decltype(sound4cl_ProcessAudio_Planar
    ) >("ProcessAudio_Planar"    );
00450                fnGetBufferIn             = GetPrefixSymbol< decltype(sound4cl_GetBufferIn
    ) >("GetBufferIn"            );
00451                fnGetBufferOut            = GetPrefixSymbol< decltype(sound4cl_GetBufferOut
    ) >("GetBufferOut"           );
00452                fnGetEstimatedDelay       = GetPrefixSymbol< decltype(sound4cl_GetEstimatedDelay
    ) >("GetEstimatedDelay"      );
00453                fnGetFormatName           = GetPrefixSymbol< decltype(sound4cl_GetFormatName
    ) >("GetFormatName"          );
00454                fnGetFormatFromName       = GetPrefixSymbol< decltype(sound4cl_GetFormatFromName
    ) >("GetFormatFromName"      );
00455                fnGetBytesFromFormat      = GetPrefixSymbol< decltype(sound4cl_GetBytesFromFormat
    ) >("GetBytesFromFormat"     );
00456                fnGetMaxPacketFrame       = GetPrefixSymbol< decltype(sound4cl_GetMaxPacketFrame
    ) >("GetMaxPacketFrame"      );
00457                fnAddAudio                = GetPrefixSymbol< decltype(sound4cl_AddAudio
    ) >("AddAudio"               );
00458                fnAddPadAudio             = GetPrefixSymbol< decltype(sound4cl_AddPadAudio
    ) >("AddPadAudio"            );
00459                fnGetOutputCount          = GetPrefixSymbol< decltype(sound4cl_GetOutputCount
    ) >("GetOutputCount"         );
00460                fnGetAudio                = GetPrefixSymbol< decltype(sound4cl_GetAudio
    ) >("GetAudio"               );
00461                fnAudioConvertFrom        = GetPrefixSymbol< decltype(sound4cl_AudioConvertFrom
    ) >("AudioConvertFrom"       );
00462                fnAudioConvertTo          = GetPrefixSymbol< decltype(sound4cl_AudioConvertTo
    ) >("AudioConvertTo"         );
00463                fnStereoToMono            = GetPrefixSymbol< decltype(sound4cl_StereoToMono
    ) >("StereoToMono"           );
00464                fnMonoToStereo            = GetPrefixSymbol< decltype(sound4cl_MonoToStereo
    ) >("MonoToStereo"           );
00465                fnAudioMonoFromLiveStereo = GetPrefixSymbol<
    decltype(sound4cl_AudioMonoFromLiveStereo) >("AudioMonoFromLiveStereo");
00466                fnAudioMonoToLiveStereo   = GetPrefixSymbol< decltype(sound4cl_AudioMonoToLiveStereo
    ) >("AudioMonoToLiveStereo"  );
00467                fnNewClient               = GetPrefixSymbol< decltype(sound4cl_NewClient
    ) >("NewClient"              );
00468                fnDeleteClient            = GetPrefixSymbol< decltype(sound4cl_DeleteClient
    ) >("DeleteClient"           );
00469                fnProcessJson             = GetPrefixSymbol< decltype(sound4cl_ProcessJson
    ) >("ProcessJson"            );
00470                fnFreeJsonAnswer          = GetPrefixSymbol< decltype(sound4cl_FreeJsonAnswer
    ) >("FreeJsonAnswer"         );
00471                fnSaveState               = GetPrefixSymbol< decltype(sound4cl_SaveState
    ) >("SaveState"              );
00472            } catch (std::runtime_error& ) {
00473                return false;
00474            }
00475            // Webserver : not for all projects
00476            try {
00477                fnWebserver_tcp           = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp
    ) >("Webserver_tcp"          );
00478                fnWebserver_tcp2          = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp2
    ) >("Webserver_tcp2"         );
00479                fnWebserver               = GetPrefixSymbol< decltype(sound4cl_Webserver
    ) >("Webserver"              );
00480                fnWebserver_Stop          = GetPrefixSymbol< decltype(sound4cl_Webserver_Stop
    ) >("Webserver_Stop"         );
00481                fnWebserver_Status        = GetPrefixSymbol< decltype(sound4cl_Webserver_Status
    ) >("Webserver_Status"       );
00482            } catch (std::runtime_error& ) {
00483                // Ignored, handler will take care
00484            }
00485            // Stereo/Mono conversion : must be present
00486            try {
00487                // added 2023-03-22
00488                fnStereoToMono_Planar     = GetPrefixSymbol< decltype(sound4cl_StereoToMono_Planar
    ) >("StereoToMono_Planar"    );
00489                fnMonoToStereo_Planar     = GetPrefixSymbol< decltype(sound4cl_MonoToStereo_Planar
    ) >("MonoToStereo_Planar"    );
00490
00491            } catch (std::runtime_error& ) {
00492                return false;
00493            }
00494            // C functions allowed to be missing
00495            try {
00496                // added 2023-05-22
00497                fnSetMetadata             = GetPrefixSymbol< decltype(sound4cl_SetMetadata
    ) >("SetMetadata"            );
00498                fnGetMetadataInfos        = GetPrefixSymbol< decltype(sound4cl_GetMetadataInfos
    ) >("GetMetadataInfos"       );
00499                fnFreeMetadataInfos       = GetPrefixSymbol< decltype(sound4cl_FreeMetadataInfos
    ) >("FreeMetadataInfos"      );
```

```
                // added 2023-06-20
                fnSetPresetManager            = GetPrefixSymbol< decltype(sound4cl_SetPresetManager
            ) >("SetPresetManager"              );
                fnPresetManager_InformChange  = GetPrefixSymbol<
decltype(sound4cl_PresetManager_InformChange     ) >("PresetManager_InformChange"    );

                // added 2023-09-21
                fnGetPossibleChunkSizeInFrames = GetPrefixSymbol<
decltype(sound4cl_GetPossibleChunkSizeInFrames  ) >("GetPossibleChunkSizeInFrames"  );
                fnGetProcessChunkFrames        = GetPrefixSymbol<
decltype(sound4cl_GetProcessChunkFrames         ) >("GetProcessChunkFrames"         );
                fnInitProcess3                 = GetPrefixSymbol< decltype(sound4cl_InitProcess3
            ) >("InitProcess3"                  );
            }  catch (std::runtime_error& ) {
                // Ignored, handler will take care
            }

            // bus: not for all projects
            try {
                // added 2023-06-26
                fnNewBus                   = GetPrefixSymbol< decltype(sound4cl_NewBus
) >("NewBus"                 );
                fnFreeBus                  = GetPrefixSymbol< decltype(sound4cl_FreeBus
) >("FreeBus"                );
                fnSetInstanceBus           = GetPrefixSymbol< decltype(sound4cl_SetInstanceBus
) >("SetInstanceBus"         );
            }  catch (std::runtime_error& ) {
                // Ignored, handler will take care
            }

            // Other optional
            try {
                // added 2024-02-19
                fnSetMetadataMulti         = GetPrefixSymbol< decltype(sound4cl_SetMetadataMulti
) >("SetMetadataMulti"       );

            }  catch (std::runtime_error& ) {
                // Ignored, handler will take care
            }

            m_bOK=true;
            return true;
        }
        bool IsOk() const {
            return m_bOK;
        }

        void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
sound4cl_SampleFormat fmt)
            { fnAudioConvertFrom(payload, output, nSpl, fmt); }
        void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat
fmt)
            { fnAudioConvertTo(input, payload, nSpl, fmt); }
        void StereoToMono(const float *input, float *output, size_t nSpl)
            { fnStereoToMono(input, output, nSpl); }
        void MonoToStereo(const float *input, float *output, size_t nSpl)
            { fnMonoToStereo(input, output, nSpl); }
        void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl)
            { fnStereoToMono_Planar(inputL, inputR, output, nSpl); }
        void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl)
            { fnMonoToStereo_Planar(input, outputL, outputR, nSpl); }
        void AudioMonoFromLiveStereo(const uint8_t *payload, float *output)
            { fnAudioMonoFromLiveStereo(payload, output); }
        void AudioMonoToLiveStereo(const float *input, uint8_t *payload)
            { fnAudioMonoToLiveStereo(input, payload); }

        std::string GetVersion()
            { return std::string(fnGetVersion()); }

        unsigned int GetChunkSizeInFrames()
            { return fnGetChunkSizeInFrames(); }

        std::vector<unsigned int> GetPossibleChunkSizeInFrames()
            {
                std::vector<unsigned int> list;
                if (fnGetPossibleChunkSizeInFrames) {
                    for (unsigned int* src=fnGetPossibleChunkSizeInFrames();*src;src++) {
                        list.push_back(*src);
                    }
                } else {
                    list.push_back(fnGetChunkSizeInFrames());
                }
                return list;
            }

        unsigned int GetChannelCount()
            { return fnGetChannelCount(); }
```

```
        unsigned int GetAudioInputCount()
            { return fnGetAudioInputCount(); }

        unsigned int GetAudioOutputCount()
            { return fnGetAudioOutputCount(); }

        unsigned int GetSampleRate()
            { return fnGetSampleRate(); }

        std::string GetFormatName( const sound4cl_SampleFormat fmt)
            { return std::string(fnGetFormatName(fmt)); }

        sound4cl_SampleFormat GetFormatFromName( const std::string& name)
            { return fnGetFormatFromName(name.c_str()); }

        unsigned int GetBytesFromFormat( const sound4cl_SampleFormat fmt)
            { return fnGetBytesFromFormat(fmt); }

        void SetLogSeverity(LogSeverity severity)
            { fnSetLogSeverity(severity); }

        void SetLoggerCallback(log_cb_t cb)
            { _log_cb=cb; fnSetLoggerCallback(&_log_cb_c); }
    };

    class CBus {
        public:
            CBus(CProcessor& dynlib)
                : m_dynlib(dynlib)
                {
                    if (m_dynlib.fnNewBus) {
                        m_bus=m_dynlib.fnNewBus();
                    }
                }
            ~CBus()
                {
                    if (m_bus && m_dynlib.fnFreeBus) {
                        m_dynlib.fnFreeBus(m_bus);
                    }
                    m_bus = nullptr;
                }
            sound4cl_CBus *Get() const { return m_bus; }
        private:
            CProcessor& m_dynlib;
            sound4cl_CBus *m_bus;
    };

    class CPresetLoader {
    public:
        CPresetLoader() = default;
        virtual ~CPresetLoader() = default;

        virtual bool IsReadOnly() = 0;
        virtual bool Exists(const std::filesystem::path &name) = 0;
        virtual bool Remove(const std::filesystem::path &name) = 0;
        virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
        virtual std::vector<std::filesystem::path> GetAll() = 0;
        virtual std::string Read(const std::filesystem::path &filename) = 0;
        virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
    };
    using PresetChange_Kind = sound4cl_PresetChange_Kind;

    class CInstance {
    public:
        CInstance(CProcessor& dynlib)
        : m_dynlib(dynlib)
            {
                params = m_dynlib.fnNewParameters();
                if (!params) throw std::bad_alloc();
            }
        // non-copyable
        CInstance( const CInstance& ) = delete;
        CInstance& operator=( const CInstance& ) = delete;
        ~CInstance()
            {
                Stop();
                if (params) m_dynlib.fnFreeParameters(params);
            }
       bool IsOk() const { return instance!=nullptr; }
        void SetParam(const std::string& name, const std::string& value)
            {
                assert(params);
                m_dynlib.fnSetParameter(params, name.c_str(), value.c_str());
            }
        std::string GetParam(const std::string& name)
            {
```

```
                assert(params);
                auto c_value=m_dynlib.fnGetParameter(params, name.c_str());
                std::string ret(c_value);
                m_dynlib.fnFreeParameterValue(c_value);
                return ret;
            }

        void SetBus(const CBus& bus) {
            assert(params);
            if (m_dynlib.fnSetInstanceBus) {
                m_dynlib.fnSetInstanceBus(params,bus.Get());
            }
        }

        void SetPresetManager(CPresetLoader *preset_manager)
            {
                if (!preset_manager) return;
                if (!m_dynlib.fnSetPresetManager) return;
                assert(params);
                m_dynlib.fnSetPresetManager(params,
                    sound4cl_custom_reader,
                    sound4cl_custom_reader_free,
                    sound4cl_custom_writer,
                    sound4cl_custom_exists,
                    sound4cl_custom_getall,
                    sound4cl_custom_getall_free,
                    sound4cl_custom_remove,
                    sound4cl_custom_rename,
                    preset_manager->IsReadOnly(),
                    preset_manager
                );
            }

        bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string& Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int json_port = 0, unsigned int frames_per_chunk=0)
            {
                assert(!instance);
                std::string l_save_path;
                if (!save_path.empty()) {
                    l_save_path = save_path.u8string();
                }
                if (frames_per_chunk==0) {
                    frames_per_chunk=m_dynlib.GetChunkSizeInFrames();
                }
                if (m_dynlib.fnInitProcess3) {
                    instance = m_dynlib.fnInitProcess3(LoginKey.c_str(), RadioName.c_str(), Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
                } else if (frames_per_chunk!=m_dynlib.GetChunkSizeInFrames()) {
                    return false;
                } else {
                    instance = m_dynlib.fnInitProcess2(LoginKey.c_str(), RadioName.c_str(), Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
                }
                if (!instance) return false;
                if (!init_metadata.empty()) {
                    for (auto&& [key,value]: init_metadata) {
                        m_dynlib.fnSetMetadata(instance, key.c_str(), value.c_str());
                    }
                    init_metadata.clear();
                }
                update_thread = std::thread([this, json_port](){
                    OnUpdateThreadStart();
                    m_dynlib.fnStartUpdateThread(instance, json_port);
                    OnUpdateThreadStop();
                });
                int timeout=5000;
                #ifdef DEBUG
                    timeout=60*1000; // For debugger pauses
                #endif
                if (m_dynlib.fnWaitUpdateThreadReady(instance, timeout)<0) {
                    return false;
                }
                return true;
            }

        unsigned int GetChunkFrames() {
            if (instance && m_dynlib.fnGetProcessChunkFrames) {
                return m_dynlib.fnGetProcessChunkFrames(instance);
            }
            return m_dynlib.fnGetChunkSizeInFrames();
        }

        void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind change_kind)
            {
                if (m_dynlib.fnPresetManager_InformChange) {
```

```
                    m_dynlib.fnPresetManager_InformChange(instance, relative_path.c_str(), (sound4cl_PresetChange_Kind) change_kind);
                }
            }

        void SetMetadata(const std::string& key, const char* value)
            {
                if (m_dynlib.fnSetMetadata) {
                    if (instance) {
                        m_dynlib.fnSetMetadata(instance, key.c_str(), value);
                    } else {
                        init_metadata.push_back( {key, value} );
                    }
                }
            }


        void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
            {
                if (instance) {
                    if (m_dynlib.fnSetMetadataMulti) {
                        typedef const char* pchar;
                        pchar* keyvalue=new pchar[2*list.size()+1];
                        size_t n=0;
                        for (auto&& [key,value]: list) {
                            keyvalue[2*n+0]=key.c_str();
                            keyvalue[2*n+1]=value;
                            n++;
                        }
                        keyvalue[2*n+0]=nullptr;

                        sound4cl_SetMetadataMulti(instance, keyvalue);

                        delete[] keyvalue;
                    } else {
                        for (auto&& [key,value]: list) {
                            SetMetadata(key, value);
                        }
                    }
                } else {
                    for (auto&& [key,value]: list) {
                        if (value) {
                            init_metadata.push_back( {key, value} );
                        }
                    }
                }
            }

        std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
            {
                std::vector< std::tuple<std::string,std::string> > values;
                if (m_dynlib.fnGetMetadataInfos) {
                    const char** c_values = m_dynlib.fnGetMetadataInfos(instance);
                    if (c_values) {
                        for (const char** c_value=c_values; *c_value; ) {
                            std::string key(*c_value);
                            c_value++;
                            if (*c_value) {
                                std::string descr(*c_value);
                                c_value++;
                                values.push_back( {key,descr} );
                            }
                        }
                        m_dynlib.fnFreeMetadataInfos(instance, c_values);
                    }
                }
                return values;
            }

        // Can be overloaded, called at start of UpdateThread in the thread
        virtual void OnUpdateThreadStart() {}
        // Can be overloaded to call at end of UpdateThread in the thread
        virtual void OnUpdateThreadStop() {}

        bool StartWebServer(int http_port, int https_port=0)
            {
                assert(instance);
                assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
                if (!m_dynlib.fnWebserver) {
                    return false;
                }
                webserver = m_dynlib.fnWebserver(http_port,https_port,instance);
                if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
                    return false;
                }
                int web_status=m_dynlib.fnWebserver_Status(webserver);
                if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
```

```
                    return false;
                }
                if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
                    return false;
                }
                return true;
            }
        void StopWebServer()
            {
                if (!m_dynlib.fnWebserver_Stop) return;
                if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
                    m_dynlib.fnWebserver_Stop(webserver, 1000);
                    webserver = SOUND4_INVALID_WEBSERVER_ID;
                }
            }

        int Terminate()
            {
                if (!instance) return 0;
                return m_dynlib.fnTerminateProcess(instance);
            }

        void Stop()
            {
                if (!instance) return;
                StopWebServer();
                if (update_thread.joinable()) {
                    m_dynlib.fnStopUpdateThread(instance); // this will exit the update thread
                    update_thread.join();
                }
                m_dynlib.fnExitProcess(instance);
                instance=nullptr;
            }

        unsigned int GetEstimatedDelay()
            {
                assert(instance);
                return m_dynlib.fnGetEstimatedDelay(instance);
            }

        float* GetBufferIn()
            {
                assert(instance);
                return m_dynlib.fnGetBufferIn(instance);

            }
        float* GetBufferOut()
            {
                assert(instance);
                return m_dynlib.fnGetBufferOut(instance);
            }
        void ProcessAudio()
            {
                assert(instance);
                m_dynlib.fnProcessAudio(instance, m_dynlib.fnGetBufferIn(instance),
m_dynlib.fnGetBufferOut(instance));
            }
        // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
        void ProcessAudio_Planar(float const * const *input, float * const *output)
            {
                assert(instance);
                m_dynlib.fnProcessAudio_Planar(instance, input, output);
            }

            template<typename T>
        std::vector<T> ProcessAnyAudio(const std::vector<T> input)
            {
                assert(instance);
                std::vector<T> output;
                unsigned int out_offset=0;
                unsigned int in_offset=0;
                unsigned int todo = input.size();
                while (todo>0) {
                    unsigned int left = AddAudio(&input[in_offset], todo);
                    unsigned int out_avail = m_dynlib.fnGetOutputCount(instance);
                    output.resize(out_offset + out_avail);
                    GetAudio(&output[out_offset], out_avail);
                    out_offset+=out_avail;
                    in_offset += todo-left;
                    todo=left;
                }
                return output;
            }

        class CClient {
            public:
```

```
                CClient(sound4cl_CInstance* instance, CProcessor& dynlib)
                : m_dynlib(dynlib)
                    {
                        assert(instance);
                        client=m_dynlib.fnNewClient(instance);
                        if (!client) throw std::bad_alloc();
                    }
                // non-copyable
                CClient( const CInstance& ) = delete;
                CClient& operator=( const CInstance& ) = delete;
                ~CClient()
                    {
                        if (client) {
                            m_dynlib.fnDeleteClient(client);
                            client=nullptr;
                        }
                    }
                std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
                    {
                        assert(client);
                        int need_save=0;
                        const char *canswer = m_dynlib.fnProcessJson(client, request.c_str(), &need_save);
                        if (!canswer) return {};
                        std::string answer(canswer);
                        m_dynlib.fnFreeJsonAnswer (canswer);
                        if (NeedSave) {
                            *NeedSave=(need_save!=0);
                        }
                        return answer;
                    }
            private:
                CProcessor& m_dynlib;
                sound4cl_CClientInstance *client = nullptr;
        };
        std::shared_ptr<CClient> NewClient()
            {
                assert(instance);
                return std::make_shared<CClient>(instance, m_dynlib);
            }

    protected:
            template<typename T>
        unsigned int AddAudio(const T* payload, unsigned int nFrame)
            {
                assert(instance);
                return m_dynlib.fnAddAudio(instance, reinterpret_cast<const T*>(payload), nFrame, helper::SampleFormat<T>::format);
            }
            template<typename T>
        unsigned int GetAudio(T* payload, unsigned int max_nFrame)
            {
                assert(instance);
                return m_dynlib.fnGetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame, helper::SampleFormat<T>::format);
            }

    private:
        CProcessor& m_dynlib;
        sound4cl_CParameters* params = nullptr;
        sound4cl_CInstance* instance = nullptr;
        std::thread update_thread;
        std::vector< std::pair<std::string,std::string> > init_metadata;
        uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
    };

}; // namespace sound4


// bridge C callbacks to CPresetLoader
extern "C" {
    static char *sound4cl_custom_reader(const fs_char *filename, void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
        std::string content=preset_loader->Read(filename);
        return strdup(content.c_str());
    }
    static void sound4cl_custom_reader_free(char *content, void* userdata) {
        if (!content) return;
        free(content);
    }
    static int sound4cl_custom_writer(const fs_char *filename, const char *content, void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
        auto res=preset_loader->Write(filename,content);
        return res?0:-1;
    }
    static int sound4cl_custom_exists(const fs_char *filename, void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
```

```
        auto res=preset_loader->Exists(filename);
        return res?0:-1;
    }
    static fs_char** sound4cl_custom_getall(void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
        auto res=preset_loader->GetAll();
        fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
        for (size_t n=0;n<res.size();n++) {
            all[n]=fs_strdup(res[n].c_str());
        }
        all[res.size()]=nullptr;
        return all;
    }
    static void sound4cl_custom_getall_free(fs_char** all, void* userdata) {
        if (!all) return;
        for (fs_char** one=all;*one;one++) {
            free(*one);
        }
        free(all);
    }
    static int sound4cl_custom_remove(const fs_char *filename, void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
        auto res=preset_loader->Remove(filename);
        return res?0:-1;
    }
    static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
        sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
        auto res=preset_loader->Rename(from,to);
        return res?0:-1;
    }

};
```

## 11.17 sound4cl_cdef.h File Reference

C function signature definition for all sound4.cl libraries.

```
#include <stdint.h>
#include <stringapiset.h>
#include <wchar.h>
```

### Classes

- struct sound4cl_SOUND4_ProcessInfo

### Macros

- #define fs_strdup wcsdup

  *helper for OS dependant file name copy*
- #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
- #define SOUND4_WEBSERVER_HTTP_OK (1<<0)
- #define SOUND4_WEBSERVER_HTTPS_OK (1<<1)

### Typedefs

- typedef wchar_t fs_char
- typedef void(∗ sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char ∗)
- typedef char ∗(∗ sound4cl_storage_reader) (const fs_char ∗filename, void ∗userdata)
- typedef void(∗ sound4cl_storage_reader_free) (char ∗content, void ∗userdata)
- typedef int(∗ sound4cl_storage_writer) (const fs_char ∗filename, const char ∗content, void ∗userdata)
- typedef int(∗ sound4cl_storage_exists) (const fs_char ∗filename, void ∗userdata)
- typedef fs_char ∗∗(∗ sound4cl_storage_getall) (void ∗userdata)
- typedef void(∗ sound4cl_storage_getall_free) (fs_char ∗∗all, void ∗userdata)
- typedef int(∗ sound4cl_storage_remove) (const fs_char ∗filename, void ∗userdata)
- typedef int(∗ sound4cl_storage_rename) (const fs_char ∗from, const fs_char ∗to, void ∗userdata)

## Enumerations

- enum sound4cl_LogSeverity {
  sound4cl_none = 0 , sound4cl_fatal = 1 , sound4cl_error = 2 , sound4cl_warning = 3 ,
  sound4cl_info = 4 , sound4cl_verbose = 5 , sound4cl_verbose2 = 6 , sound4cl_verbose3 = 7 ,
  sound4cl_verbose4 = 8 , sound4cl_verbose5 = 9 }
- enum sound4cl_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }
- enum sound4cl_SampleFormat {
  INVALID_FORMAT =0 , S16_LE , S16_BE , S24_LE ,
  S24_BE , S32_LE , S32_BE , F32_LE ,
  F32_BE , S16_NATIVE =S16_LE , S24_NATIVE =S24_LE , S32_NATIVE =S32_LE ,
  F32_NATIVE =F32_LE }

## Functions

- const struct sound4cl_SOUND4_ProcessInfo ∗ sound4cl_SOUND4_GetProcessInfo ()
- const char ∗ sound4cl_GetVersion ()
- unsigned int sound4cl_GetChunkSizeInFrames ()
- unsigned int ∗ sound4cl_GetPossibleChunkSizeInFrames ()
- unsigned int sound4cl_GetChannelCount ()
- unsigned int sound4cl_GetAudioInputCount ()
- unsigned int sound4cl_GetAudioOutputCount ()
- unsigned int sound4cl_GetSampleRate ()
- void sound4cl_SetLoggerCallback (sound4cl_loggerfn logger)
- void sound4cl_SetLogSeverity (enum sound4cl_LogSeverity severity)
- struct sound4cl_CParameters ∗ sound4cl_NewParameters ()
- void sound4cl_FreeParameters (struct sound4cl_CParameters ∗params)
- void sound4cl_SetParameter (struct sound4cl_CParameters ∗params, const char ∗name, const char ∗value)
- const char ∗ sound4cl_GetParameter (struct sound4cl_CParameters ∗params, const char ∗name)
- void sound4cl_FreeParameterValue (const char ∗value)
- struct sound4cl_CInstance ∗ sound4cl_InitProcess (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path)
- struct sound4cl_CInstance ∗ sound4cl_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct sound4cl_↩CParameters ∗parameters)
- struct sound4cl_CInstance ∗ sound4cl_InitProcess3 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct sound4cl_↩CParameters ∗parameters, unsigned int frames_per_chunk)
- unsigned int sound4cl_GetProcessChunkFrames (struct sound4cl_CInstance ∗instance)
- int sound4cl_TerminateProcess (struct sound4cl_CInstance ∗instance)
- void sound4cl_ExitProcess (struct sound4cl_CInstance ∗instance)
- struct sound4cl_CBus ∗ sound4cl_NewBus ()
- void sound4cl_FreeBus (struct sound4cl_CBus ∗bus)
- void sound4cl_SetInstanceBus (struct sound4cl_CParameters ∗params, struct sound4cl_CBus ∗bus)
- void sound4cl_SetPresetManager (struct sound4cl_CParameters ∗params, sound4cl_storage_reader reader, sound4cl_storage_reader_free, sound4cl_storage_writer writer, sound4cl_storage_exists exists, sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free, sound4cl_storage_remove remove, sound4cl_storage_rename rename, int IsReadOnly, void ∗userdata)
- void sound4cl_PresetManager_InformChange (struct sound4cl_CInstance ∗instance, const fs_char ∗relative_path, enum sound4cl_PresetChange_Kind change_kind)
- void sound4cl_SetMetadata (struct sound4cl_CInstance ∗instance, const char ∗key, const char ∗value)
- void sound4cl_SetMetadataMulti (struct sound4cl_CInstance ∗instance, const char ∗∗keyvalue)
- const char ∗∗ sound4cl_GetMetadataInfos (struct sound4cl_CInstance ∗instance)
- void sound4cl_FreeMetadataInfos (struct sound4cl_CInstance ∗instance, const char ∗∗infos)

- void sound4cl_StartUpdateThread (struct sound4cl_CInstance ∗instance, unsigned int port)
- void sound4cl_StopUpdateThread (struct sound4cl_CInstance ∗instance)
- int sound4cl_WaitUpdateThreadReady (struct sound4cl_CInstance ∗instance, int milliseconds)
- void sound4cl_ProcessAudio (struct sound4cl_CInstance ∗instance, const float ∗input, float ∗output)
- void sound4cl_ProcessAudio_Planar (struct sound4cl_CInstance ∗instance, float const ∗const ∗input, float ∗const ∗output)
- float ∗ sound4cl_GetBufferIn (struct sound4cl_CInstance ∗instance)
- float ∗ sound4cl_GetBufferOut (struct sound4cl_CInstance ∗instance)
- unsigned int sound4cl_GetEstimatedDelay (struct sound4cl_CInstance ∗instance)
- const char ∗ sound4cl_GetFormatName (const enum sound4cl_SampleFormat fmt)
- enum sound4cl_SampleFormat sound4cl_GetFormatFromName (const char ∗name)
- unsigned int sound4cl_GetBytesFromFormat (const enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_GetMaxPacketFrame (struct sound4cl_CInstance ∗instance)
- unsigned int sound4cl_AddAudio (struct sound4cl_CInstance ∗instance, const uint8_t ∗payload, unsigned int nFrame, enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_AddPadAudio (struct sound4cl_CInstance ∗instance)
- unsigned int sound4cl_GetOutputCount (struct sound4cl_CInstance ∗instance)
- unsigned int sound4cl_GetAudio (struct sound4cl_CInstance ∗instance, uint8_t ∗payload, unsigned int max↩_nFrame, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertFrom (const uint8_t ∗payload, float ∗output, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertTo (const float ∗input, uint8_t ∗payload, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_StereoToMono (const float ∗input, float ∗output, size_t nFrame)
- void sound4cl_MonoToStereo (const float ∗input, float ∗output, size_t nFrame)
- void sound4cl_StereoToMono_Planar (const float ∗inputL, const float ∗inputR, float ∗output, size_t nFrame)
- void sound4cl_MonoToStereo_Planar (const float ∗input, float ∗outputL, float ∗outputR, size_t nFrame)
- void sound4cl_AudioMonoFromLiveStereo (const uint8_t ∗payload, float ∗output)
- void sound4cl_AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)
- struct sound4cl_CClientInstance ∗ sound4cl_NewClient (struct sound4cl_CInstance ∗instance)
- void sound4cl_DeleteClient (struct sound4cl_CClientInstance ∗client)
- const char ∗ sound4cl_ProcessJson (struct sound4cl_CClientInstance ∗client, const char ∗json_str, int ∗need_save)
- void sound4cl_FreeJsonAnswer (const char ∗json_str)
- int sound4cl_SaveState (struct sound4cl_CInstance ∗instance)
- uint64_t sound4cl_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char ∗socket_ip, unsigned int socket_port)
- uint64_t sound4cl_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char ∗socket_ip, unsigned int socket_port, const struct sound4cl_CParameters ∗parameters)
- uint64_t sound4cl_Webserver (unsigned int listenport, unsigned int listenport_secure, struct sound4cl_↩CInstance ∗instance)
- int sound4cl_Webserver_Stop (uint64_t id, int timeout_ms)
- int sound4cl_Webserver_Status (uint64_t id)

## 11.17.1 Macro Definition Documentation

### 11.17.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 17 of file sound4cl_cdef.h.

#### 11.17.1.2 SOUND4_INVALID_WEBSERVER_ID

```
#define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
```

Definition at line 62 of file sound4cl_cdef.h.

#### 11.17.1.3 SOUND4_WEBSERVER_HTTP_OK

```
#define SOUND4_WEBSERVER_HTTP_OK (1<<0)
```

Definition at line 63 of file sound4cl_cdef.h.

#### 11.17.1.4 SOUND4_WEBSERVER_HTTPS_OK

```
#define SOUND4_WEBSERVER_HTTPS_OK (1<<1)
```

Definition at line 64 of file sound4cl_cdef.h.

### 11.17.2 Typedef Documentation

#### 11.17.2.1 fs_char

```
typedef wchar_t fs_char
```

Definition at line 18 of file sound4cl_cdef.h.

#### 11.17.2.2 sound4cl_loggerfn

```
typedef void(* sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *)
```

Definition at line 102 of file sound4cl_cdef.h.

#### 11.17.2.3 sound4cl_storage_exists

```
typedef int(* sound4cl_storage_exists) (const fs_char *filename, void *userdata)
```

Definition at line 143 of file sound4cl_cdef.h.

#### 11.17.2.4 sound4cl_storage_getall

```
typedef fs_char**(* sound4cl_storage_getall) (void *userdata)
```

Definition at line 144 of file sound4cl_cdef.h.

#### 11.17.2.5 sound4cl_storage_getall_free

```
typedef void(* sound4cl_storage_getall_free) (fs_char **all, void *userdata)
```

Definition at line 145 of file sound4cl_cdef.h.

#### 11.17.2.6 sound4cl_storage_reader

```
typedef char*(* sound4cl_storage_reader) (const fs_char *filename, void *userdata)
```

Definition at line 140 of file sound4cl_cdef.h.

#### 11.17.2.7 sound4cl_storage_reader_free

```
typedef void(* sound4cl_storage_reader_free) (char *content, void *userdata)
```

Definition at line 141 of file sound4cl_cdef.h.

#### 11.17.2.8 sound4cl_storage_remove

```
typedef int(* sound4cl_storage_remove) (const fs_char *filename, void *userdata)
```

Definition at line 146 of file sound4cl_cdef.h.

#### 11.17.2.9 sound4cl_storage_rename

```
typedef int(* sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)
```

Definition at line 147 of file sound4cl_cdef.h.

#### 11.17.2.10 sound4cl_storage_writer

```
typedef int(* sound4cl_storage_writer) (const fs_char *filename, const char *content, void
*userdata)
```

Definition at line 142 of file sound4cl_cdef.h.

### 11.17.3 Enumeration Type Documentation

#### 11.17.3.1 sound4cl_LogSeverity

```
enum sound4cl_LogSeverity
```

**Enumerator**

| | |
|---|---|
| sound4cl_none | |
| sound4cl_fatal | |
| sound4cl_error | |
| sound4cl_warning | |
| sound4cl_info | |
| sound4cl_verbose | |
| sound4cl_verbose2 | |
| sound4cl_verbose3 | |
| sound4cl_verbose4 | |
| sound4cl_verbose5 | |

Definition at line 41 of file sound4cl_cdef.h.

#### 11.17.3.2 sound4cl_PresetChange_Kind

`enum sound4cl_PresetChange_Kind`

**Enumerator**

| | |
|---|---|
| change_kind_created | Path has been created. |
| change_kind_modified | Path has been modified. |
| change_kind_deleted | Path has been deleted. |

Definition at line 57 of file sound4cl_cdef.h.

#### 11.17.3.3 sound4cl_SampleFormat

`enum sound4cl_SampleFormat`

Supported formats for audio I/O

**Enumerator**

| | |
|---|---|
| INVALID_FORMAT | Invalid. |
| S16_LE | 16-bit signed integer, little-endian |
| S16_BE | 16-bit signed integer, big-endian |
| S24_LE | 24-bit signed integer, little-endian |
| S24_BE | 24-bit signed integer, big-endian |
| S32_LE | 32-bit signed integer, little-endian |
| S32_BE | 32-bit signed integer, big-endian |
| F32_LE | 32-bit floating-point, little-endian |
| F32_BE | 32-bit floating-point, big-endian |

**Enumerator**

| | |
|---|---|
| S16_NATIVE | 16-bit signed integer, native |
| S24_NATIVE | 24-bit signed integer, native |
| S32_NATIVE | 32-bit signed integer, native |
| F32_NATIVE | 32-bit floating-point, native |

Definition at line 69 of file sound4cl_cdef.h.

## 11.17.4 Function Documentation

### 11.17.4.1 sound4cl_AddAudio()

```
unsigned int sound4cl_AddAudio (
            struct sound4cl_CInstance * instance,
            const uint8_t * payload,
            unsigned int nFrame,
            enum sound4cl_SampleFormat fmt )
```

### 11.17.4.2 sound4cl_AddPadAudio()

```
unsigned int sound4cl_AddPadAudio (
            struct sound4cl_CInstance * instance )
```

### 11.17.4.3 sound4cl_AudioConvertFrom()

```
void sound4cl_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

### 11.17.4.4 sound4cl_AudioConvertTo()

```
void sound4cl_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

#### 11.17.4.5 sound4cl_AudioMonoFromLiveStereo()

```
void sound4cl_AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )
```

#### 11.17.4.6 sound4cl_AudioMonoToLiveStereo()

```
void sound4cl_AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )
```

#### 11.17.4.7 sound4cl_DeleteClient()

```
void sound4cl_DeleteClient (
            struct sound4cl_CClientInstance * client )
```

#### 11.17.4.8 sound4cl_ExitProcess()

```
void sound4cl_ExitProcess (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.9 sound4cl_FreeBus()

```
void sound4cl_FreeBus (
            struct sound4cl_CBus * bus )
```

#### 11.17.4.10 sound4cl_FreeJsonAnswer()

```
void sound4cl_FreeJsonAnswer (
            const char * json_str )
```

#### 11.17.4.11 sound4cl_FreeMetadataInfos()

```
void sound4cl_FreeMetadataInfos (
            struct sound4cl_CInstance * instance,
            const char ** infos )
```

#### 11.17.4.12 sound4cl_FreeParameters()

```
void sound4cl_FreeParameters (
            struct sound4cl_CParameters * params )
```

#### 11.17.4.13 sound4cl_FreeParameterValue()

```
void sound4cl_FreeParameterValue (
            const char * value )
```

#### 11.17.4.14 sound4cl_GetAudio()

```
unsigned int sound4cl_GetAudio (
            struct sound4cl_CInstance * instance,
            uint8_t * payload,
            unsigned int max_nFrame,
            enum sound4cl_SampleFormat fmt )
```

#### 11.17.4.15 sound4cl_GetAudioInputCount()

```
unsigned int sound4cl_GetAudioInputCount ( )
```

#### 11.17.4.16 sound4cl_GetAudioOutputCount()

```
unsigned int sound4cl_GetAudioOutputCount ( )
```

#### 11.17.4.17 sound4cl_GetBufferIn()

```
float* sound4cl_GetBufferIn (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.18 sound4cl_GetBufferOut()

```
float* sound4cl_GetBufferOut (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.19 sound4cl_GetBytesFromFormat()

```
unsigned int sound4cl_GetBytesFromFormat (
            const enum sound4cl_SampleFormat fmt )
```

#### 11.17.4.20 sound4cl_GetChannelCount()

```
unsigned int sound4cl_GetChannelCount ( )
```

#### 11.17.4.21 sound4cl_GetChunkSizeInFrames()

```
unsigned int sound4cl_GetChunkSizeInFrames ( )
```

#### 11.17.4.22 sound4cl_GetEstimatedDelay()

```
unsigned int sound4cl_GetEstimatedDelay (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.23 sound4cl_GetFormatFromName()

```
enum sound4cl_SampleFormat sound4cl_GetFormatFromName (
            const char * name )
```

#### 11.17.4.24 sound4cl_GetFormatName()

```
const char* sound4cl_GetFormatName (
            const enum sound4cl_SampleFormat fmt )
```

#### 11.17.4.25 sound4cl_GetMaxPacketFrame()

```
unsigned int sound4cl_GetMaxPacketFrame (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.26 sound4cl_GetMetadataInfos()

```
const char** sound4cl_GetMetadataInfos (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.27 sound4cl_GetOutputCount()

```
unsigned int sound4cl_GetOutputCount (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.28 sound4cl_GetParameter()

```
const char* sound4cl_GetParameter (
            struct sound4cl_CParameters * params,
            const char * name )
```

#### 11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames()

```
unsigned int* sound4cl_GetPossibleChunkSizeInFrames ( )
```

#### 11.17.4.30 sound4cl_GetProcessChunkFrames()

```
unsigned int sound4cl_GetProcessChunkFrames (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.31 sound4cl_GetSampleRate()

```
unsigned int sound4cl_GetSampleRate ( )
```

#### 11.17.4.32 sound4cl_GetVersion()

```
const char* sound4cl_GetVersion ( )
```

#### 11.17.4.33 sound4cl_InitProcess()

```
struct sound4cl_CInstance* sound4cl_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path )
```

#### 11.17.4.34 sound4cl_InitProcess2()

```
struct sound4cl_CInstance* sound4cl_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters )
```

#### 11.17.4.35 sound4cl_InitProcess3()

```
struct sound4cl_CInstance* sound4cl_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters,
            unsigned int frames_per_chunk )
```

#### 11.17.4.36 sound4cl_MonoToStereo()

```
void sound4cl_MonoToStereo (
            const float * input,
            float * output,
            size_t nFrame )
```

#### 11.17.4.37 sound4cl_MonoToStereo_Planar()

```
void sound4cl_MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nFrame )
```

#### 11.17.4.38 sound4cl_NewBus()

```
struct sound4cl_CBus* sound4cl_NewBus ( )
```

#### 11.17.4.39 sound4cl_NewClient()

```
struct sound4cl_CClientInstance* sound4cl_NewClient (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.40 sound4cl_NewParameters()

```
struct sound4cl_CParameters* sound4cl_NewParameters ( )
```

#### 11.17.4.41 sound4cl_PresetManager_InformChange()

```
void sound4cl_PresetManager_InformChange (
            struct sound4cl_CInstance * instance,
            const fs_char * relative_path,
            enum sound4cl_PresetChange_Kind change_kind )
```

#### 11.17.4.42 sound4cl_ProcessAudio()

```
void sound4cl_ProcessAudio (
            struct sound4cl_CInstance * instance,
            const float * input,
            float * output )
```

#### 11.17.4.43 sound4cl_ProcessAudio_Planar()

```
void sound4cl_ProcessAudio_Planar (
            struct sound4cl_CInstance * instance,
            float const *const * input,
            float *const * output )
```

#### 11.17.4.44 sound4cl_ProcessJson()

```
const char* sound4cl_ProcessJson (
            struct sound4cl_CClientInstance * client,
            const char * json_str,
            int * need_save )
```

#### 11.17.4.45 sound4cl_SaveState()

```
int sound4cl_SaveState (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.46 sound4cl_SetInstanceBus()

```
void sound4cl_SetInstanceBus (
            struct sound4cl_CParameters * params,
            struct sound4cl_CBus * bus )
```

#### 11.17.4.47 sound4cl_SetLoggerCallback()

```
void sound4cl_SetLoggerCallback (
            sound4cl_loggerfn logger )
```

#### 11.17.4.48 sound4cl_SetLogSeverity()

```
void sound4cl_SetLogSeverity (
            enum sound4cl_LogSeverity severity )
```

#### 11.17.4.49 sound4cl_SetMetadata()

```
void sound4cl_SetMetadata (
            struct sound4cl_CInstance * instance,
            const char * key,
            const char * value )
```

#### 11.17.4.50 sound4cl_SetMetadataMulti()

```
void sound4cl_SetMetadataMulti (
            struct sound4cl_CInstance * instance,
            const char ** keyvalue )
```

#### 11.17.4.51 sound4cl_SetParameter()

```
void sound4cl_SetParameter (
            struct sound4cl_CParameters * params,
            const char * name,
            const char * value )
```

#### 11.17.4.52 sound4cl_SetPresetManager()

```
void sound4cl_SetPresetManager (
            struct sound4cl_CParameters * params,
            sound4cl_storage_reader reader,
            sound4cl_storage_reader_free ,
            sound4cl_storage_writer writer,
            sound4cl_storage_exists exists,
            sound4cl_storage_getall getall,
            sound4cl_storage_getall_free getall_free,
            sound4cl_storage_remove remove,
            sound4cl_storage_rename rename,
            int IsReadOnly,
            void * userdata )
```

#### 11.17.4.53 sound4cl_SOUND4_GetProcessInfo()

```
const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo ( )
```

#### 11.17.4.54 sound4cl_StartUpdateThread()

```
void sound4cl_StartUpdateThread (
            struct sound4cl_CInstance * instance,
            unsigned int port )
```

#### 11.17.4.55 sound4cl_StereoToMono()

```
void sound4cl_StereoToMono (
            const float * input,
            float * output,
            size_t nFrame )
```

#### 11.17.4.56 sound4cl_StereoToMono_Planar()

```
void sound4cl_StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nFrame )
```

#### 11.17.4.57 sound4cl_StopUpdateThread()

```
void sound4cl_StopUpdateThread (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.58 sound4cl_TerminateProcess()

```
int sound4cl_TerminateProcess (
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.59 sound4cl_WaitUpdateThreadReady()

```
int sound4cl_WaitUpdateThreadReady (
            struct sound4cl_CInstance * instance,
            int milliseconds )
```

#### 11.17.4.60 sound4cl_Webserver()

```
uint64_t sound4cl_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct sound4cl_CInstance * instance )
```

#### 11.17.4.61 sound4cl_Webserver_Status()

```
int sound4cl_Webserver_Status (
            uint64_t id )
```

#### 11.17.4.62 sound4cl_Webserver_Stop()

```
int sound4cl_Webserver_Stop (
            uint64_t id,
            int timeout_ms )
```

#### 11.17.4.63 sound4cl_Webserver_tcp()

```
uint64_t sound4cl_Webserver_tcp (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port )
```

#### 11.17.4.64 sound4cl_Webserver_tcp2()

```
uint64_t sound4cl_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct sound4cl_CParameters * parameters )
```

## 11.18 sound4cl_cdef.h

```
#pragma once

#include <stdint.h>
#ifdef __unix__
    #include <sys/types.h>
#endif

#ifdef _WIN32
    #include <stringapiset.h>
    #include <wchar.h>
    #define fs_strdup wcsdup
    typedef wchar_t fs_char;
    #ifdef _MSC_VER
        #define __BYTE_ORDER__  REG_DWORD
        #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
        #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
    #endif
#else // !_WIN32
    #define fs_strdup strdup
    typedef char fs_char;
#endif // !_WIN32

#ifdef __cplusplus
extern "C" {
#endif // __cplusplus

struct sound4cl_SOUND4_ProcessInfo {
    size_t struct_size;
    const char *name;
    const char *prefix;
    const char *shortname;
    const char *version;
};
enum sound4cl_LogSeverity {
    sound4cl_none = 0,
    sound4cl_fatal = 1,
    sound4cl_error = 2,
    sound4cl_warning = 3,
    sound4cl_info = 4,
    sound4cl_verbose = 5,
    sound4cl_verbose2 = 6,
    sound4cl_verbose3 = 7,
    sound4cl_verbose4 = 8,
    sound4cl_verbose5 = 9,
};
struct sound4cl_CInstance;
struct sound4cl_CParameters;
struct sound4cl_CBus;
struct sound4cl_CClientInstance;
enum sound4cl_PresetChange_Kind {
    change_kind_created = 1,
    change_kind_modified   ,
    change_kind_deleted    ,
};
#define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)
#define SOUND4_WEBSERVER_HTTP_OK               (1«0)
#define SOUND4_WEBSERVER_HTTPS_OK              (1«1)

enum sound4cl_SampleFormat {
    INVALID_FORMAT=0,
    S16_LE,
    S16_BE,
    S24_LE,
    S24_BE,
    S32_LE,
    S32_BE,
    F32_LE,
    F32_BE,

    // Native definitions, only links to the physical ones
    #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
        S16_NATIVE=S16_LE,
        S24_NATIVE=S24_LE,
        S32_NATIVE=S32_LE,
        F32_NATIVE=F32_LE,
    #else
        S16_NATIVE=S16_BE,
        S24_NATIVE=S24_BE,
        S32_NATIVE=S32_BE,
        F32_NATIVE=F32_BE,
    #endif
};

const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo();
```

```
const char *sound4cl_GetVersion();
unsigned int sound4cl_GetChunkSizeInFrames();
unsigned int* sound4cl_GetPossibleChunkSizeInFrames();
unsigned int sound4cl_GetChannelCount();
unsigned int sound4cl_GetAudioInputCount();
unsigned int sound4cl_GetAudioOutputCount();
unsigned int sound4cl_GetSampleRate();
typedef void (*sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *);
void sound4cl_SetLoggerCallback( sound4cl_loggerfn logger );
void sound4cl_SetLogSeverity( enum sound4cl_LogSeverity severity );
struct sound4cl_CParameters *sound4cl_NewParameters();
void sound4cl_FreeParameters(struct sound4cl_CParameters *params);
void sound4cl_SetParameter(struct sound4cl_CParameters *params, const char *name, const char *value);
const char *sound4cl_GetParameter(struct sound4cl_CParameters *params, const char *name);
void sound4cl_FreeParameterValue(const char *value);
struct sound4cl_CInstance *sound4cl_InitProcess(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path
    );
struct sound4cl_CInstance *sound4cl_InitProcess2(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path,
        const struct sound4cl_CParameters *parameters
    );
struct sound4cl_CInstance *sound4cl_InitProcess3(
        const char *LoginKey,
        const char *RadioName,
        const char *Access_Key_ID,
        const char *Access_Key_Secret,
        const char *save_path,
        const struct sound4cl_CParameters *parameters,
        unsigned int frames_per_chunk
    );
unsigned int sound4cl_GetProcessChunkFrames(struct sound4cl_CInstance *instance);
int sound4cl_TerminateProcess(struct sound4cl_CInstance *instance);
void sound4cl_ExitProcess(struct sound4cl_CInstance *instance);
struct sound4cl_CBus* sound4cl_NewBus();
void sound4cl_FreeBus(struct sound4cl_CBus *bus);
void sound4cl_SetInstanceBus(struct sound4cl_CParameters *params, struct sound4cl_CBus *bus);
typedef char * (*sound4cl_storage_reader) (const fs_char *filename, void* userdata);
typedef void (*sound4cl_storage_reader_free) (char *content, void* userdata);
typedef int (*sound4cl_storage_writer) (const fs_char *filename, const char *content, void* userdata);
typedef int (*sound4cl_storage_exists) (const fs_char *filename, void* userdata);
typedef fs_char** (*sound4cl_storage_getall) (void* userdata);
typedef void (*sound4cl_storage_getall_free) (fs_char**all, void* userdata);
typedef int (*sound4cl_storage_remove) (const fs_char *filename, void* userdata);
typedef int (*sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
void sound4cl_SetPresetManager(struct sound4cl_CParameters *params
    , sound4cl_storage_reader reader, sound4cl_storage_reader_free
    , sound4cl_storage_writer writer
    , sound4cl_storage_exists exists
    , sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free
    , sound4cl_storage_remove remove
    , sound4cl_storage_rename rename
    , int IsReadOnly
    , void* userdata
    );
void sound4cl_PresetManager_InformChange(struct sound4cl_CInstance *instance, const fs_char
    *relative_path, enum sound4cl_PresetChange_Kind change_kind);
void sound4cl_SetMetadata(struct sound4cl_CInstance *instance, const char* key, const char* value);
void sound4cl_SetMetadataMulti(struct sound4cl_CInstance *instance, const char** keyvalue);
const char** sound4cl_GetMetadataInfos(struct sound4cl_CInstance *instance);
void sound4cl_FreeMetadataInfos(struct sound4cl_CInstance *instance, const char** infos);
void sound4cl_StartUpdateThread(struct sound4cl_CInstance *instance, unsigned int port);
void sound4cl_StopUpdateThread(struct sound4cl_CInstance *instance);
int sound4cl_WaitUpdateThreadReady(struct sound4cl_CInstance *instance, int milliseconds);
void sound4cl_ProcessAudio(struct sound4cl_CInstance *instance, const float *input, float *output );
void sound4cl_ProcessAudio_Planar(struct sound4cl_CInstance *instance
    , float const * const *input
    , float * const *output);
float *sound4cl_GetBufferIn(struct sound4cl_CInstance *instance);
float *sound4cl_GetBufferOut(struct sound4cl_CInstance *instance);
unsigned int sound4cl_GetEstimatedDelay(struct sound4cl_CInstance *instance);
const char *sound4cl_GetFormatName(const enum sound4cl_SampleFormat fmt);
enum sound4cl_SampleFormat sound4cl_GetFormatFromName(const char *name);
unsigned int sound4cl_GetBytesFromFormat(const enum sound4cl_SampleFormat fmt);
unsigned int sound4cl_GetMaxPacketFrame(struct sound4cl_CInstance *instance);
unsigned int sound4cl_AddAudio(struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned
    int nFrame, enum sound4cl_SampleFormat fmt );
unsigned int sound4cl_AddPadAudio(struct sound4cl_CInstance *instance);
unsigned int sound4cl_GetOutputCount(struct sound4cl_CInstance *instance);
```

```
unsigned int sound4cl_GetAudio(struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum sound4cl_SampleFormat fmt );
void sound4cl_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum sound4cl_SampleFormat fmt);
void sound4cl_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum sound4cl_SampleFormat fmt);
void sound4cl_StereoToMono(const float *input, float *output, size_t nFrame);
void sound4cl_MonoToStereo(const float *input, float *output, size_t nFrame);
void sound4cl_StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nFrame);
void sound4cl_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nFrame);
void sound4cl_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
void sound4cl_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
struct sound4cl_CClientInstance *sound4cl_NewClient(struct sound4cl_CInstance *instance);
void sound4cl_DeleteClient(struct sound4cl_CClientInstance *client);
const char *sound4cl_ProcessJson(struct sound4cl_CClientInstance *client, const char *json_str, int *need_save);
void sound4cl_FreeJsonAnswer(const char *json_str);
int sound4cl_SaveState(struct sound4cl_CInstance *instance);
uint64_t sound4cl_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port);
uint64_t sound4cl_Webserver_tcp2(unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct sound4cl_CParameters *parameters);
uint64_t sound4cl_Webserver(unsigned int listenport, unsigned int listenport_secure, struct sound4cl_CInstance *instance);
int sound4cl_Webserver_Stop(uint64_t id, int timeout_ms);
int sound4cl_Webserver_Status(uint64_t id);

#ifdef __cplusplus
} // extern "C"
#endif // __cplusplus
```

# Index


~CBus
sound4::CBus, 83
~CClient
sound4::CInstance::CClient, 85
sound4::impact::CInstance::CClient, 87
sound4::impact::dyn::CInstance::CClient, 89
~CDynLoader
sound4::helper::CDynLoader, 104
sound4::impact::dyn::helper::CDynLoader, 107
~CInstance
sound4::CInstance, 110
sound4::impact::CInstance, 121
sound4::impact::dyn::CInstance, 131
~CPresetLoader
sound4::CPresetLoader, 140
sound4::impact::CPresetLoader, 142
sound4::impact::dyn::CPresetLoader, 145
~MyPresetManager
MyPresetManager, 171

AddAudio
sound4::CInstance, 110
sound4::impact::CInstance, 121
sound4::impact::dyn::CDynLib, 92
sound4::impact::dyn::CInstance, 131
AddPadAudio
sound4::impact::dyn::CDynLib, 93
Audio convert helpers, 48
impact_AudioConvertFrom, 48
impact_AudioConvertTo, 49
impact_AudioMonoFromLiveStereo, 49
impact_AudioMonoToLiveStereo, 49
impact_MonoToStereo, 50
impact_MonoToStereo_Planar, 50
impact_StereoToMono, 51
impact_StereoToMono_Planar, 51
Audio managing (any audio), 43
F32_BE, 44
F32_LE, 44
F32_NATIVE, 44
impact_AddAudio, 45
impact_AddPadAudio, 45
impact_GetAudio, 45
impact_GetBytesFromFormat, 46
impact_GetFormatFromName, 46
impact_GetFormatName, 47
impact_GetMaxPacketFrame, 47
impact_GetOutputCount, 47
impact_SampleFormat, 44
INVALID_FORMAT, 44
S16_BE, 44
S16_LE, 44
S16_NATIVE, 44
S24_BE, 44
S24_LE, 44
S24_NATIVE, 44
S32_BE, 44
S32_LE, 44
S32_NATIVE, 44
AudioConvertFrom
sound4::CProcessor, 149
sound4::impact::dyn::CDynLib, 93
AudioConvertTo
sound4::CProcessor, 149
sound4::impact::dyn::CDynLib, 93
AudioMonoFromLiveStereo
sound4::CProcessor, 149
sound4::impact::dyn::CDynLib, 93
AudioMonoToLiveStereo
sound4::CProcessor, 150
sound4::impact::dyn::CDynLib, 93

CBus
sound4::CBus, 83
sound4::CProcessor, 155
CClient
sound4::CInstance::CClient, 85
sound4::impact::CInstance::CClient, 87
sound4::impact::dyn::CInstance::CClient, 88, 89
CDynLib
sound4::impact::dyn::CDynLib, 92
CDynLoader
sound4::helper::CDynLoader, 103, 104
sound4::impact::dyn::helper::CDynLoader, 106, 107
change_kind_created
Custom preset managagement, 36
sound4cl_cdef.h, 255
change_kind_deleted
Custom preset managagement, 36
sound4cl_cdef.h, 255
change_kind_modified
Custom preset managagement, 36
sound4cl_cdef.h, 255
CHANNEL_COUNT
sound4.impact.cl-pipe.cpp, 196
CHUNK_SIZE
sound4.impact.cl-pipe.cpp, 197
ChunkFrames
sound4::impact, 77

sound4::impact::dyn, 79
CInstance
sound4::CInstance, 110
sound4::CProcessor, 155
sound4::impact::CInstance, 120, 121
sound4::impact::dyn::CInstance, 131
Client's JSon config interfacing, 52
impact_DeleteClient, 52
impact_FreeJsonAnswer, 52
impact_NewClient, 53
impact_ProcessJson, 53
impact_SaveState, 54
Close
sound4::helper::CDynLoader, 104
sound4::impact::dyn::helper::CDynLoader, 107
CPresetLoader
sound4::CPresetLoader, 140
sound4::impact::CPresetLoader, 142
sound4::impact::dyn::CPresetLoader, 144
CProcessor
sound4::CProcessor, 149
Create
sound4::CInstance, 111
sound4::impact::CInstance, 121
sound4::impact::dyn::CInstance, 131
Custom preset managagement, 31
change_kind_created, 36
change_kind_deleted, 36
change_kind_modified, 36
fs_char, 32
impact_PresetChange_Kind, 35
impact_PresetManager_InformChange, 36
impact_SetPresetManager, 36
impact_storage_exists, 32
impact_storage_getall, 33
impact_storage_getall_free, 33
impact_storage_reader, 33
impact_storage_reader_free, 34
impact_storage_remove, 34
impact_storage_rename, 34
impact_storage_writer, 35

DeleteClient
sound4::impact::dyn::CDynLib, 93
Direct processing, 40
impact_GetBufferIn, 41
impact_GetBufferOut, 41
impact_GetEstimatedDelay, 42
impact_ProcessAudio, 42
impact_ProcessAudio_Planar, 43
dyn, 71
DynFunc_t
sound4::helper, 75
sound4::impact::dyn::helper, 81
DynFuncHolder
sound4::helper::DynFuncHolder< T >, 167
sound4::impact::dyn::helper::DynFuncHolder< T >, 168
DynLib_t
sound4::helper, 75
sound4::impact::dyn::helper, 81

Environment variables and parameters, 19
error
Global setup, 23
sound4, 73
Exists
MyPresetManager, 172
sound4::CPresetLoader, 140
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 145
ExitProcess
sound4::impact::dyn::CDynLib, 94

F32_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
F32_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
F32_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 256
fatal
Global setup, 23
sound4, 73
fnAddAudio
sound4::CProcessor, 156
fnAddPadAudio
sound4::CProcessor, 156
fnAudioConvertFrom
sound4::CProcessor, 156
fnAudioConvertTo
sound4::CProcessor, 156
fnAudioMonoFromLiveStereo
sound4::CProcessor, 156
fnAudioMonoToLiveStereo
sound4::CProcessor, 156
fnDeleteClient
sound4::CProcessor, 157
fnExitProcess
sound4::CProcessor, 157
fnFreeBus
sound4::CProcessor, 157
fnFreeJsonAnswer
sound4::CProcessor, 157
fnFreeMetadataInfos
sound4::CProcessor, 157
fnFreeParameters
sound4::CProcessor, 157
fnFreeParameterValue
sound4::CProcessor, 158
fnGetAudio
sound4::CProcessor, 158
fnGetAudioInputCount
sound4::CProcessor, 158
fnGetAudioOutputCount
sound4::CProcessor, 158

fnGetBufferIn
sound4::CProcessor, 158
fnGetBufferOut
sound4::CProcessor, 158
fnGetBytesFromFormat
sound4::CProcessor, 159
fnGetChannelCount
sound4::CProcessor, 159
fnGetChunkSizeInFrames
sound4::CProcessor, 159
fnGetEstimatedDelay
sound4::CProcessor, 159
fnGetFormatFromName
sound4::CProcessor, 159
fnGetFormatName
sound4::CProcessor, 159
fnGetMaxPacketFrame
sound4::CProcessor, 160
fnGetMetadataInfos
sound4::CProcessor, 160
fnGetOutputCount
sound4::CProcessor, 160
fnGetParameter
sound4::CProcessor, 160
fnGetPossibleChunkSizeInFrames
sound4::CProcessor, 160
fnGetProcessChunkFrames
sound4::CProcessor, 160
fnGetSampleRate
sound4::CProcessor, 161
fnGetVersion
sound4::CProcessor, 161
fnInitProcess
sound4::CProcessor, 161
fnInitProcess2
sound4::CProcessor, 161
fnInitProcess3
sound4::CProcessor, 161
fnMonoToStereo
sound4::CProcessor, 161
fnMonoToStereo_Planar
sound4::CProcessor, 162
fnNewBus
sound4::CProcessor, 162
fnNewClient
sound4::CProcessor, 162
fnNewParameters
sound4::CProcessor, 162
fnPresetManager_InformChange
sound4::CProcessor, 162
fnProcessAudio
sound4::CProcessor, 162
fnProcessAudio_Planar
sound4::CProcessor, 163
fnProcessJson
sound4::CProcessor, 163
fnSaveState
sound4::CProcessor, 163
fnSetInstanceBus
sound4::CProcessor, 163
fnSetLoggerCallback
sound4::CProcessor, 163
fnSetLogSeverity
sound4::CProcessor, 163
fnSetMetadata
sound4::CProcessor, 164
fnSetMetadataMulti
sound4::CProcessor, 164
fnSetParameter
sound4::CProcessor, 164
fnSetPresetManager
sound4::CProcessor, 164
fnStartUpdateThread
sound4::CProcessor, 164
fnStereoToMono
sound4::CProcessor, 164
fnStereoToMono_Planar
sound4::CProcessor, 165
fnStopUpdateThread
sound4::CProcessor, 165
fnTerminateProcess
sound4::CProcessor, 165
fnWaitUpdateThreadReady
sound4::CProcessor, 165
fnWebserver
sound4::CProcessor, 165
fnWebserver_Status
sound4::CProcessor, 165
fnWebserver_Stop
sound4::CProcessor, 166
fnWebserver_tcp
sound4::CProcessor, 166
fnWebserver_tcp2
sound4::CProcessor, 166
format
sound4::helper::SampleFormat< float >, 177
sound4::helper::SampleFormat< int16_t >, 179
sound4::helper::SampleFormat< int32_t >, 180
sound4::impact::dyn::helper::SampleFormat< float >, 177
sound4::impact::dyn::helper::SampleFormat< int16_t >, 179
sound4::impact::dyn::helper::SampleFormat< int32_t >, 181
sound4::impact::helper::SampleFormat< float >, 178
sound4::impact::helper::SampleFormat< int16_t >, 180
sound4::impact::helper::SampleFormat< int32_t >, 182
FreeJsonAnswer
sound4::impact::dyn::CDynLib, 94
FreeMetadataInfos
sound4::impact::dyn::CDynLib, 94
FreeParameters
sound4::impact::dyn::CDynLib, 94

FreeParameterValue
sound4::impact::dyn::CDynLib, 94
fs_char
Custom preset managagement, 32
sound4cl_cdef.h, 253
fs_strdup
sound4.impact.cl.hpp, 210
sound4.impact.cl_dyn.hpp, 219
sound4cl_cdef.h, 252

Get
sound4::CBus, 84
GetAll
MyPresetManager, 172
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 145
GetAudio
sound4::CInstance, 111
sound4::impact::CInstance, 122
sound4::impact::dyn::CDynLib, 94
sound4::impact::dyn::CInstance, 132
GetAudioInputCount
sound4::CProcessor, 150
sound4::impact::dyn::CDynLib, 95
GetAudioOutputCount
sound4::CProcessor, 150
sound4::impact::dyn::CDynLib, 95
GetBufferIn
sound4::CInstance, 112
sound4::impact::CInstance, 122
sound4::impact::dyn::CDynLib, 95
sound4::impact::dyn::CInstance, 132
GetBufferOut
sound4::CInstance, 112
sound4::impact::CInstance, 122
sound4::impact::dyn::CDynLib, 95
sound4::impact::dyn::CInstance, 132
GetBytesFromFormat
sound4::CProcessor, 150
sound4::impact::dyn::CDynLib, 95
GetChannelCount
sound4::CProcessor, 151
sound4::impact::dyn::CDynLib, 95
GetChunkFrames
sound4::CInstance, 112
sound4::impact::CInstance, 123
sound4::impact::dyn::CInstance, 133
GetChunkSizeInFrames
sound4::CProcessor, 151
sound4::impact::dyn::CDynLib, 96
GetEnv
sound4.impact.cl-pipe-cxx.cpp, 187
sound4.impact.cl-pipe-dyn.cpp, 191
sound4cl-pipe.cpp, 232
GetEstimatedDelay
sound4::CInstance, 112
sound4::impact::CInstance, 123
sound4::impact::dyn::CDynLib, 96
sound4::impact::dyn::CInstance, 133
GetFormatFromName
sound4::CProcessor, 151
sound4::impact::dyn::CDynLib, 96
GetFormatName
sound4::CProcessor, 152
sound4::impact::dyn::CDynLib, 96
GetMaxPacketFrame
sound4::impact::dyn::CDynLib, 96
GetMetadataInfos
sound4::CInstance, 113
sound4::impact::CInstance, 123
sound4::impact::dyn::CDynLib, 96
sound4::impact::dyn::CInstance, 133
GetOutputCount
sound4::impact::dyn::CDynLib, 97
GetParam
sound4::CInstance, 113
sound4::impact::CInstance, 124
sound4::impact::dyn::CInstance, 134
GetParameter
sound4::impact::dyn::CDynLib, 97
GetPossibleChunkSizeInFrames
sound4::CProcessor, 152
sound4::impact::dyn::CDynLib, 97
GetPrefixSymbol
sound4::CProcessor, 152
GetProcessChunkFrames
sound4::impact::dyn::CDynLib, 97
GetSampleRate
sound4::CProcessor, 153
sound4::impact::dyn::CDynLib, 97
GetSymbol
sound4::helper::CDynLoader, 104
sound4::impact::dyn::helper::CDynLoader, 107
GetSymbol_safe
sound4::helper::CDynLoader, 104
sound4::impact::dyn::helper::CDynLoader, 107
GetThisLibraryPath
sound4::helper::CDynLoader, 105
sound4::impact::dyn::helper::CDynLoader, 108
GetVersion
sound4::CProcessor, 153
sound4::impact::dyn::CDynLib, 97
Global setup, 20
error, 23
fatal, 23
IMPACT_AUDIO_INPUT_COUNT, 21
IMPACT_AUDIO_OUTPUT_COUNT, 21
IMPACT_AUDIOFRAME_COUNT, 21
IMPACT_CHANNEL_COUNT, 21
impact_GetAudioInputCount, 23
impact_GetAudioOutputCount, 23
impact_GetChannelCount, 23
impact_GetChunkSizeInFrames, 24
impact_GetPossibleChunkSizeInFrames, 24
impact_GetSampleRate, 24
impact_GetVersion, 25

impact_loggerfn, 22
impact_LogSeverity, 22
IMPACT_SAMPLE_RATE, 22
impact_SetLoggerCallback, 25
impact_SetLogSeverity, 25
info, 23
none, 23
SOUND4_GetProcessInfo, 26
verbose, 23
verbose2, 23
verbose3, 23
verbose4, 23
verbose5, 23
warning, 23

impact_AddAudio
Audio managing (any audio), 45
impact_AddPadAudio
Audio managing (any audio), 45
IMPACT_AUDIO_INPUT_COUNT
Global setup, 21
IMPACT_AUDIO_OUTPUT_COUNT
Global setup, 21
impact_AudioConvertFrom
Audio convert helpers, 48
impact_AudioConvertTo
Audio convert helpers, 49
IMPACT_AUDIOFRAME_COUNT
Global setup, 21
impact_AudioMonoFromLiveStereo
Audio convert helpers, 49
impact_AudioMonoToLiveStereo
Audio convert helpers, 49
impact_CClientInstance, 169
IMPACT_CHANNEL_COUNT
Global setup, 21
impact_CInstance, 170
impact_CParameters, 170
IMPACT_DECL
sound4.impact.cl.h, 204
impact_DeleteClient
Client's JSon config interfacing, 52
impact_ExitProcess
Process creation/destruction, 27
impact_FreeJsonAnswer
Client's JSon config interfacing, 52
impact_FreeMetadataInfos
Metadata Management, 37
impact_FreeParameters
Process creation/destruction, 27
impact_FreeParameterValue
Process creation/destruction, 27
impact_GetAudio
Audio managing (any audio), 45
impact_GetAudioInputCount
Global setup, 23
impact_GetAudioOutputCount
Global setup, 23
impact_GetBufferIn
Direct processing, 41
impact_GetBufferOut
Direct processing, 41
impact_GetBytesFromFormat
Audio managing (any audio), 46
impact_GetChannelCount
Global setup, 23
impact_GetChunkSizeInFrames
Global setup, 24
impact_GetEstimatedDelay
Direct processing, 42
impact_GetFormatFromName
Audio managing (any audio), 46
impact_GetFormatName
Audio managing (any audio), 47
impact_GetMaxPacketFrame
Audio managing (any audio), 47
impact_GetMetadataInfos
Metadata Management, 38
impact_GetOutputCount
Audio managing (any audio), 47
impact_GetParameter
Process creation/destruction, 27
impact_GetPossibleChunkSizeInFrames
Global setup, 24
impact_GetProcessChunkFrames
Process creation/destruction, 28
impact_GetSampleRate
Global setup, 24
impact_GetVersion
Global setup, 25
IMPACT_HAS_CLOUDBUS
sound4.impact.cl.h, 204
IMPACT_HAS_WEBSERVER
sound4.impact.cl.h, 204
impact_InitProcess
Process creation/destruction, 28
impact_InitProcess2
Process creation/destruction, 28
impact_InitProcess3
Process creation/destruction, 29
impact_loggerfn
Global setup, 22
impact_LogSeverity
Global setup, 22
impact_MonoToStereo
Audio convert helpers, 50
impact_MonoToStereo_Planar
Audio convert helpers, 50
impact_NewClient
Client's JSon config interfacing, 53
impact_NewParameters
Process creation/destruction, 30
impact_PresetChange_Kind
Custom preset managagement, 35
impact_PresetManager_InformChange
Custom preset managagement, 36
impact_ProcessAudio

Direct processing, 42
impact_ProcessAudio_Planar
Direct processing, 43
impact_ProcessJson
Client's JSon config interfacing, 53
IMPACT_SAMPLE_RATE
Global setup, 22
impact_SampleFormat
Audio managing (any audio), 44
impact_SaveState
Client's JSon config interfacing, 54
impact_SetLoggerCallback
Global setup, 25
impact_SetLogSeverity
Global setup, 25
impact_SetMetadata
Metadata Management, 38
impact_SetMetadataMulti
Metadata Management, 38
impact_SetParameter
Process creation/destruction, 30
impact_SetPresetManager
Custom preset managagement, 36
impact_StartUpdateThread
Update Management, 39
impact_StereoToMono
Audio convert helpers, 51
impact_StereoToMono_Planar
Audio convert helpers, 51
impact_StopUpdateThread
Update Management, 40
impact_storage_exists
Custom preset managagement, 32
impact_storage_getall
Custom preset managagement, 33
impact_storage_getall_free
Custom preset managagement, 33
impact_storage_reader
Custom preset managagement, 33
impact_storage_reader_free
Custom preset managagement, 34
impact_storage_remove
Custom preset managagement, 34
impact_storage_rename
Custom preset managagement, 34
impact_storage_writer
Custom preset managagement, 35
impact_TerminateProcess
Process creation/destruction, 31
impact_WaitUpdateThreadReady
Update Management, 40
impact_Webserver
Web server, 62
impact_Webserver_Status
Web server, 62
impact_Webserver_Stop
Web server, 63
impact_Webserver_tcp
Web server, 63
impact_Webserver_tcp2
Web server, 64
info
Global setup, 23
sound4, 73
InitProcess
sound4::impact::dyn::CDynLib, 98
InitProcess2
sound4::impact::dyn::CDynLib, 98
InitProcess3
sound4::impact::dyn::CDynLib, 98
InputChannels
sound4::impact, 77
sound4::impact::dyn, 79
InputSampleSize
sound4::impact, 77
sound4::impact::dyn, 80
INVALID_FORMAT
Audio managing (any audio), 44
sound4cl_cdef.h, 255
IsOk
sound4::CInstance, 113
sound4::CProcessor, 153
sound4::helper::CDynLoader, 105
sound4::helper::DynFuncHolder< T >, 167
sound4::impact::CInstance, 124
sound4::impact::dyn::CDynLib, 92
sound4::impact::dyn::CInstance, 134
sound4::impact::dyn::helper::CDynLoader, 108
sound4::impact::dyn::helper::DynFuncHolder< T >, 169
IsReadOnly
MyPresetManager, 172, 173
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 145

JSON syntax, 54
JSON syntax for internal setup, 57

Load
sound4::CProcessor, 153
sound4::helper::CDynLoader, 105
sound4::impact::dyn::CDynLib, 92
sound4::impact::dyn::helper::CDynLoader, 108
log_cb_t
sound4, 72
sound4::impact, 76
sound4::impact::dyn, 79
LogSeverity
sound4, 72
sound4::impact, 76
sound4::impact::dyn, 79

main
sound4.impact.cl-pipe-cxx.cpp, 187
sound4.impact.cl-pipe-dyn.cpp, 191
sound4.impact.cl-pipe.cpp, 197

sound4cl-pipe.cpp, 232
Metadata Management, 37
impact_FreeMetadataInfos, 37
impact_GetMetadataInfos, 38
impact_SetMetadata, 38
impact_SetMetadataMulti, 38
MonoToStereo
sound4::CProcessor, 154
sound4::impact::dyn::CDynLib, 98
MonoToStereo_Planar
sound4::CProcessor, 154
sound4::impact::dyn::CDynLib, 98
MyLogger
sound4.impact.cl-pipe-cxx.cpp, 188
sound4.impact.cl-pipe-dyn.cpp, 192
sound4.impact.cl-pipe.cpp, 197
MyPresetManager, 170
~MyPresetManager, 171
Exists, 172
GetAll, 172
IsReadOnly, 172, 173
MyPresetManager, 171
Read, 173
Remove, 173
Rename, 174
SetupPath, 174
Write, 174, 175

name
SOUND4_ProcessInfo, 183
sound4cl_SOUND4_ProcessInfo, 184
NewClient
sound4::CInstance, 113
sound4::impact::CInstance, 124
sound4::impact::dyn::CDynLib, 98
sound4::impact::dyn::CInstance, 134
NewParameters
sound4::impact::dyn::CDynLib, 99
none
Global setup, 23
sound4, 73

OnUpdateThreadStart
sound4::CInstance, 114
sound4::impact::CInstance, 124
sound4::impact::dyn::CInstance, 134
OnUpdateThreadStop
sound4::CInstance, 114
sound4::impact::CInstance, 125
sound4::impact::dyn::CInstance, 135
operator bool
sound4::helper::CDynLoader, 105
sound4::helper::DynFuncHolder< T >, 167
sound4::impact::dyn::helper::CDynLoader, 108
sound4::impact::dyn::helper::DynFuncHolder< T >, 169
operator T*
sound4::helper::DynFuncHolder< T >, 167
sound4::impact::dyn::helper::DynFuncHolder< T >, 169
operator=
sound4::CInstance, 114
sound4::CInstance::CClient, 85
sound4::helper::CDynLoader, 105
sound4::impact::CInstance, 125
sound4::impact::CInstance::CClient, 87
sound4::impact::dyn::CInstance, 135
sound4::impact::dyn::CInstance::CClient, 89
sound4::impact::dyn::helper::CDynLoader, 108
OutputChannels
sound4::impact, 77
sound4::impact::dyn, 80
OutputSampleSize
sound4::impact, 77
sound4::impact::dyn, 80

prefix
SOUND4_ProcessInfo, 183
sound4cl_SOUND4_ProcessInfo, 184
PresetChange_Kind
sound4, 72
sound4::impact, 76
sound4::impact::dyn, 79
PresetManager_InformChange
sound4::CInstance, 114
sound4::impact::CInstance, 125
sound4::impact::dyn::CDynLib, 99
sound4::impact::dyn::CInstance, 135
Process creation/destruction, 26
impact_ExitProcess, 27
impact_FreeParameters, 27
impact_FreeParameterValue, 27
impact_GetParameter, 27
impact_GetProcessChunkFrames, 28
impact_InitProcess, 28
impact_InitProcess2, 28
impact_InitProcess3, 29
impact_NewParameters, 30
impact_SetParameter, 30
impact_TerminateProcess, 31
ProcessAnyAudio
sound4::CInstance, 115
sound4::impact::CInstance, 125
sound4::impact::dyn::CInstance, 135
ProcessAudio
sound4::CInstance, 115
sound4::impact::CInstance, 126
sound4::impact::dyn::CDynLib, 99
sound4::impact::dyn::CInstance, 136
ProcessAudio_Planar
sound4::CInstance, 115
sound4::impact::CInstance, 126
sound4::impact::dyn::CDynLib, 99
sound4::impact::dyn::CInstance, 136
ProcessJson
sound4::CInstance::CClient, 85
sound4::impact::CInstance::CClient, 87

sound4::impact::dyn::CDynLib, 99
sound4::impact::dyn::CInstance::CClient, 89
ProcessLoop
sound4.impact.cl-pipe-cxx.cpp, 188
sound4.impact.cl-pipe-dyn.cpp, 192
sound4.impact.cl-pipe.cpp, 198
sound4cl-pipe.cpp, 233

Read
MyPresetManager, 173
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 145
Remove
MyPresetManager, 173
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 145
Rename
MyPresetManager, 174
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 143
sound4::impact::dyn::CPresetLoader, 146

S16_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S16_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S16_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 256
S24_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S24_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S24_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 256
S32_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S32_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 255
S32_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 256
Sample using C interface, 17
Sample using C++ interface wrapper, 65
Sample using C++ interface wrapper with dynamic library loading, 65
Sample using Generic C++ interface (multi-process compatible), 68
SAMPLE_RATE
sound4.impact.cl-pipe.cpp, 197
SampleFormat
sound4::impact, 76
sound4::impact::dyn, 79
SampleRate
sound4::impact, 78
sound4::impact::dyn, 80
SaveState
sound4::impact::dyn::CDynLib, 99
SetBus
sound4::CInstance, 116
SetLoggerCallback
sound4::CProcessor, 154
sound4::impact::dyn::CDynLib, 100
SetLogSeverity
sound4::CProcessor, 154
sound4::impact::dyn::CDynLib, 100
SetMetadata
sound4::CInstance, 116
sound4::impact::CInstance, 127
sound4::impact::dyn::CDynLib, 100
sound4::impact::dyn::CInstance, 137
SetMetadataMulti
sound4::CInstance, 117
sound4::impact::CInstance, 127
sound4::impact::dyn::CDynLib, 100
sound4::impact::dyn::CInstance, 137
SetParam
sound4::CInstance, 117
sound4::impact::CInstance, 127
sound4::impact::dyn::CInstance, 137
SetParameter
sound4::impact::dyn::CDynLib, 100
SetPresetManager
sound4::CInstance, 117
sound4::impact::CInstance, 128
sound4::impact::dyn::CDynLib, 100
sound4::impact::dyn::CInstance, 138
SetupPath
MyPresetManager, 174
shortname
SOUND4_ProcessInfo, 183
sound4cl_SOUND4_ProcessInfo, 184
sound4, 71
error, 73
fatal, 73
info, 73
log_cb_t, 72
LogSeverity, 72
none, 73
PresetChange_Kind, 72
verbose, 73
verbose2, 73
verbose3, 74
verbose4, 74
verbose5, 74
warning, 74
sound4.impact.cl-pipe-cxx.cpp, 187, 189
GetEnv, 187

main, 187
MyLogger, 188
ProcessLoop, 188
Usage, 188
sound4.impact.cl-pipe-dyn.cpp, 190, 193
GetEnv, 191
main, 191
MyLogger, 192
ProcessLoop, 192
Usage, 192
sound4.impact.cl-pipe.cpp, 196, 198
CHANNEL_COUNT, 196
CHUNK_SIZE, 197
main, 197
MyLogger, 197
ProcessLoop, 198
SAMPLE_RATE, 197
Usage, 198
sound4.impact.cl.h, 201, 205
IMPACT_DECL, 204
IMPACT_HAS_CLOUDBUS, 204
IMPACT_HAS_WEBSERVER, 204
sound4.impact.cl.hpp, 208, 210
fs_strdup, 210
sound4.impact.cl_dyn.hpp, 217, 219
fs_strdup, 219
sound4::CBus, 83
~CBus, 83
CBus, 83
Get, 84
sound4::CInstance, 109
~CInstance, 110
AddAudio, 110
CInstance, 110
Create, 111
GetAudio, 111
GetBufferIn, 112
GetBufferOut, 112
GetChunkFrames, 112
GetEstimatedDelay, 112
GetMetadataInfos, 113
GetParam, 113
IsOk, 113
NewClient, 113
OnUpdateThreadStart, 114
OnUpdateThreadStop, 114
operator=, 114
PresetManager_InformChange, 114
ProcessAnyAudio, 115
ProcessAudio, 115
ProcessAudio_Planar, 115
SetBus, 116
SetMetadata, 116
SetMetadataMulti, 117
SetParam, 117
SetPresetManager, 117
StartWebServer, 118
Stop, 118
StopWebServer, 118
Terminate, 119
sound4::CInstance::CClient, 84
~CClient, 85
CClient, 85
operator=, 85
ProcessJson, 85
sound4::CPresetLoader, 140
~CPresetLoader, 140
CPresetLoader, 140
Exists, 140
GetAll, 141
IsReadOnly, 141
Read, 141
Remove, 141
Rename, 141
Write, 141
sound4::CProcessor, 146
AudioConvertFrom, 149
AudioConvertTo, 149
AudioMonoFromLiveStereo, 149
AudioMonoToLiveStereo, 150
CBus, 155
CInstance, 155
CProcessor, 149
fnAddAudio, 156
fnAddPadAudio, 156
fnAudioConvertFrom, 156
fnAudioConvertTo, 156
fnAudioMonoFromLiveStereo, 156
fnAudioMonoToLiveStereo, 156
fnDeleteClient, 157
fnExitProcess, 157
fnFreeBus, 157
fnFreeJsonAnswer, 157
fnFreeMetadataInfos, 157
fnFreeParameters, 157
fnFreeParameterValue, 158
fnGetAudio, 158
fnGetAudioInputCount, 158
fnGetAudioOutputCount, 158
fnGetBufferIn, 158
fnGetBufferOut, 158
fnGetBytesFromFormat, 159
fnGetChannelCount, 159
fnGetChunkSizeInFrames, 159
fnGetEstimatedDelay, 159
fnGetFormatFromName, 159
fnGetFormatName, 159
fnGetMaxPacketFrame, 160
fnGetMetadataInfos, 160
fnGetOutputCount, 160
fnGetParameter, 160
fnGetPossibleChunkSizeInFrames, 160
fnGetProcessChunkFrames, 160
fnGetSampleRate, 161
fnGetVersion, 161
fnInitProcess, 161

fnInitProcess2, 161
fnInitProcess3, 161
fnMonoToStereo, 161
fnMonoToStereo_Planar, 162
fnNewBus, 162
fnNewClient, 162
fnNewParameters, 162
fnPresetManager_InformChange, 162
fnProcessAudio, 162
fnProcessAudio_Planar, 163
fnProcessJson, 163
fnSaveState, 163
fnSetInstanceBus, 163
fnSetLoggerCallback, 163
fnSetLogSeverity, 163
fnSetMetadata, 164
fnSetMetadataMulti, 164
fnSetParameter, 164
fnSetPresetManager, 164
fnStartUpdateThread, 164
fnStereoToMono, 164
fnStereoToMono_Planar, 165
fnStopUpdateThread, 165
fnTerminateProcess, 165
fnWaitUpdateThreadReady, 165
fnWebserver, 165
fnWebserver_Status, 165
fnWebserver_Stop, 166
fnWebserver_tcp, 166
fnWebserver_tcp2, 166
GetAudioInputCount, 150
GetAudioOutputCount, 150
GetBytesFromFormat, 150
GetChannelCount, 151
GetChunkSizeInFrames, 151
GetFormatFromName, 151
GetFormatName, 152
GetPossibleChunkSizeInFrames, 152
GetPrefixSymbol, 152
GetSampleRate, 153
GetVersion, 153
IsOk, 153
Load, 153
MonoToStereo, 154
MonoToStereo_Planar, 154
SetLoggerCallback, 154
SetLogSeverity, 154
StereoToMono, 155
StereoToMono_Planar, 155
sound4::helper, 74
DynFunc_t, 75
DynLib_t, 75
sound4::helper::CDynLoader, 103
~CDynLoader, 104
CDynLoader, 103, 104
Close, 104
GetSymbol, 104
GetSymbol_safe, 104
GetThisLibraryPath, 105
IsOk, 105
Load, 105
operator bool, 105
operator=, 105
sound4::helper::DynFuncHolder< T >, 166
DynFuncHolder, 167
IsOk, 167
operator bool, 167
operator T*, 167
sound4::helper::SampleFormat< float >, 177
format, 177
sound4::helper::SampleFormat< int16_t >, 178
format, 179
sound4::helper::SampleFormat< int32_t >, 180
format, 180
sound4::helper::SampleFormat< T >, 175
sound4::impact, 75
ChunkFrames, 77
InputChannels, 77
InputSampleSize, 77
log_cb_t, 76
LogSeverity, 76
OutputChannels, 77
OutputSampleSize, 77
PresetChange_Kind, 76
SampleFormat, 76
SampleRate, 78
sound4::impact::CInstance, 119
~CInstance, 121
AddAudio, 121
CInstance, 120, 121
Create, 121
GetAudio, 122
GetBufferIn, 122
GetBufferOut, 122
GetChunkFrames, 123
GetEstimatedDelay, 123
GetMetadataInfos, 123
GetParam, 124
IsOk, 124
NewClient, 124
OnUpdateThreadStart, 124
OnUpdateThreadStop, 125
operator=, 125
PresetManager_InformChange, 125
ProcessAnyAudio, 125
ProcessAudio, 126
ProcessAudio_Planar, 126
SetMetadata, 127
SetMetadataMulti, 127
SetParam, 127
SetPresetManager, 128
StartWebServer, 128
Stop, 128
StopWebServer, 129
Terminate, 129
sound4::impact::CInstance::CClient, 86

~CClient, 87
CClient, 87
operator=, 87
ProcessJson, 87
sound4::impact::CPresetLoader, 142
~CPresetLoader, 142
CPresetLoader, 142
Exists, 143
GetAll, 143
IsReadOnly, 143
Read, 143
Remove, 143
Rename, 143
Write, 144
sound4::impact::dyn, 78
ChunkFrames, 79
InputChannels, 79
InputSampleSize, 80
log_cb_t, 79
LogSeverity, 79
OutputChannels, 80
OutputSampleSize, 80
PresetChange_Kind, 79
SampleFormat, 79
SampleRate, 80
sound4::impact::dyn::CDynLib, 90
AddAudio, 92
AddPadAudio, 93
AudioConvertFrom, 93
AudioConvertTo, 93
AudioMonoFromLiveStereo, 93
AudioMonoToLiveStereo, 93
CDynLib, 92
DeleteClient, 93
ExitProcess, 94
FreeJsonAnswer, 94
FreeMetadataInfos, 94
FreeParameters, 94
FreeParameterValue, 94
GetAudio, 94
GetAudioInputCount, 95
GetAudioOutputCount, 95
GetBufferIn, 95
GetBufferOut, 95
GetBytesFromFormat, 95
GetChannelCount, 95
GetChunkSizeInFrames, 96
GetEstimatedDelay, 96
GetFormatFromName, 96
GetFormatName, 96
GetMaxPacketFrame, 96
GetMetadataInfos, 96
GetOutputCount, 97
GetParameter, 97
GetPossibleChunkSizeInFrames, 97
GetProcessChunkFrames, 97
GetSampleRate, 97
GetVersion, 97
InitProcess, 98
InitProcess2, 98
InitProcess3, 98
IsOk, 92
Load, 92
MonoToStereo, 98
MonoToStereo_Planar, 98
NewClient, 98
NewParameters, 99
PresetManager_InformChange, 99
ProcessAudio, 99
ProcessAudio_Planar, 99
ProcessJson, 99
SaveState, 99
SetLoggerCallback, 100
SetLogSeverity, 100
SetMetadata, 100
SetMetadataMulti, 100
SetParameter, 100
SetPresetManager, 100
StartUpdateThread, 101
StereoToMono, 101
StereoToMono_Planar, 101
StopUpdateThread, 101
TerminateProcess, 101
WaitUpdateThreadReady, 101
Webserver, 102
Webserver_Status, 102
Webserver_Stop, 102
Webserver_tcp, 102
Webserver_tcp2, 102
sound4::impact::dyn::CInstance, 129
~CInstance, 131
AddAudio, 131
CInstance, 131
Create, 131
GetAudio, 132
GetBufferIn, 132
GetBufferOut, 132
GetChunkFrames, 133
GetEstimatedDelay, 133
GetMetadataInfos, 133
GetParam, 134
IsOk, 134
NewClient, 134
OnUpdateThreadStart, 134
OnUpdateThreadStop, 135
operator=, 135
PresetManager_InformChange, 135
ProcessAnyAudio, 135
ProcessAudio, 136
ProcessAudio_Planar, 136
SetMetadata, 137
SetMetadataMulti, 137
SetParam, 137
SetPresetManager, 138
StartWebServer, 138
Stop, 138

StopWebServer, 139
Terminate, 139
sound4::impact::dyn::CInstance::CClient, 88
~CClient, 89
CClient, 88, 89
operator=, 89
ProcessJson, 89
sound4::impact::dyn::CPresetLoader, 144
~CPresetLoader, 145
CPresetLoader, 144
Exists, 145
GetAll, 145
IsReadOnly, 145
Read, 145
Remove, 145
Rename, 146
Write, 146
sound4::impact::dyn::helper, 81
DynFunc_t, 81
DynLib_t, 81
sound4::impact::dyn::helper::CDynLoader, 106
~CDynLoader, 107
CDynLoader, 106, 107
Close, 107
GetSymbol, 107
GetSymbol_safe, 107
GetThisLibraryPath, 108
IsOk, 108
Load, 108
operator bool, 108
operator=, 108
sound4::impact::dyn::helper::DynFuncHolder< T >, 168
DynFuncHolder, 168
IsOk, 169
operator bool, 169
operator T*, 169
sound4::impact::dyn::helper::SampleFormat< float >, 177
format, 177
sound4::impact::dyn::helper::SampleFormat< int16_t >, 179
format, 179
sound4::impact::dyn::helper::SampleFormat< int32_t >, 181
format, 181
sound4::impact::dyn::helper::SampleFormat< T >, 176
sound4::impact::helper, 82
sound4::impact::helper::SampleFormat< float >, 178
format, 178
sound4::impact::helper::SampleFormat< int16_t >, 180
format, 180
sound4::impact::helper::SampleFormat< int32_t >, 181
format, 182
sound4::impact::helper::SampleFormat< T >, 176
SOUND4_GetProcessInfo
Global setup, 26
SOUND4_INVALID_WEBSERVER_ID
sound4cl_cdef.h, 252
Web server, 61
SOUND4_ProcessInfo, 182
name, 183
prefix, 183
shortname, 183
struct_size, 183
version, 183
SOUND4_WEBSERVER_HTTP_OK
sound4cl_cdef.h, 253
Web server, 61
SOUND4_WEBSERVER_HTTPS_OK
sound4cl_cdef.h, 253
Web server, 61
sound4cl-pipe.cpp, 231, 233
GetEnv, 232
main, 232
ProcessLoop, 233
sound4cl.hpp, 236, 238
sound4cl_AddAudio
sound4cl_cdef.h, 256
sound4cl_AddPadAudio
sound4cl_cdef.h, 256
sound4cl_AudioConvertFrom
sound4cl_cdef.h, 256
sound4cl_AudioConvertTo
sound4cl_cdef.h, 256
sound4cl_AudioMonoFromLiveStereo
sound4cl_cdef.h, 256
sound4cl_AudioMonoToLiveStereo
sound4cl_cdef.h, 257
sound4cl_cdef.h, 250, 267
change_kind_created, 255
change_kind_deleted, 255
change_kind_modified, 255
F32_BE, 255
F32_LE, 255
F32_NATIVE, 256
fs_char, 253
fs_strdup, 252
INVALID_FORMAT, 255
S16_BE, 255
S16_LE, 255
S16_NATIVE, 256
S24_BE, 255
S24_LE, 255
S24_NATIVE, 256
S32_BE, 255
S32_LE, 255
S32_NATIVE, 256
SOUND4_INVALID_WEBSERVER_ID, 252
SOUND4_WEBSERVER_HTTP_OK, 253
SOUND4_WEBSERVER_HTTPS_OK, 253
sound4cl_AddAudio, 256
sound4cl_AddPadAudio, 256
sound4cl_AudioConvertFrom, 256
sound4cl_AudioConvertTo, 256
sound4cl_AudioMonoFromLiveStereo, 256

sound4cl_AudioMonoToLiveStereo, 257
sound4cl_DeleteClient, 257
sound4cl_error, 255
sound4cl_ExitProcess, 257
sound4cl_fatal, 255
sound4cl_FreeBus, 257
sound4cl_FreeJsonAnswer, 257
sound4cl_FreeMetadataInfos, 257
sound4cl_FreeParameters, 258
sound4cl_FreeParameterValue, 258
sound4cl_GetAudio, 258
sound4cl_GetAudioInputCount, 258
sound4cl_GetAudioOutputCount, 258
sound4cl_GetBufferIn, 258
sound4cl_GetBufferOut, 258
sound4cl_GetBytesFromFormat, 259
sound4cl_GetChannelCount, 259
sound4cl_GetChunkSizeInFrames, 259
sound4cl_GetEstimatedDelay, 259
sound4cl_GetFormatFromName, 259
sound4cl_GetFormatName, 259
sound4cl_GetMaxPacketFrame, 259
sound4cl_GetMetadataInfos, 260
sound4cl_GetOutputCount, 260
sound4cl_GetParameter, 260
sound4cl_GetPossibleChunkSizeInFrames, 260
sound4cl_GetProcessChunkFrames, 260
sound4cl_GetSampleRate, 260
sound4cl_GetVersion, 260
sound4cl_info, 255
sound4cl_InitProcess, 261
sound4cl_InitProcess2, 261
sound4cl_InitProcess3, 261
sound4cl_loggerfn, 253
sound4cl_LogSeverity, 254
sound4cl_MonoToStereo, 261
sound4cl_MonoToStereo_Planar, 261
sound4cl_NewBus, 262
sound4cl_NewClient, 262
sound4cl_NewParameters, 262
sound4cl_none, 255
sound4cl_PresetChange_Kind, 255
sound4cl_PresetManager_InformChange, 262
sound4cl_ProcessAudio, 262
sound4cl_ProcessAudio_Planar, 262
sound4cl_ProcessJson, 263
sound4cl_SampleFormat, 255
sound4cl_SaveState, 263
sound4cl_SetInstanceBus, 263
sound4cl_SetLoggerCallback, 263
sound4cl_SetLogSeverity, 263
sound4cl_SetMetadata, 263
sound4cl_SetMetadataMulti, 264
sound4cl_SetParameter, 264
sound4cl_SetPresetManager, 264
sound4cl_SOUND4_GetProcessInfo, 264
sound4cl_StartUpdateThread, 264
sound4cl_StereoToMono, 265
sound4cl_StereoToMono_Planar, 265
sound4cl_StopUpdateThread, 265
sound4cl_storage_exists, 253
sound4cl_storage_getall, 253
sound4cl_storage_getall_free, 254
sound4cl_storage_reader, 254
sound4cl_storage_reader_free, 254
sound4cl_storage_remove, 254
sound4cl_storage_rename, 254
sound4cl_storage_writer, 254
sound4cl_TerminateProcess, 265
sound4cl_verbose, 255
sound4cl_verbose2, 255
sound4cl_verbose3, 255
sound4cl_verbose4, 255
sound4cl_verbose5, 255
sound4cl_WaitUpdateThreadReady, 265
sound4cl_warning, 255
sound4cl_Webserver, 265
sound4cl_Webserver_Status, 266
sound4cl_Webserver_Stop, 266
sound4cl_Webserver_tcp, 266
sound4cl_Webserver_tcp2, 266
sound4cl_DeleteClient
sound4cl_cdef.h, 257
sound4cl_error
sound4cl_cdef.h, 255
sound4cl_ExitProcess
sound4cl_cdef.h, 257
sound4cl_fatal
sound4cl_cdef.h, 255
sound4cl_FreeBus
sound4cl_cdef.h, 257
sound4cl_FreeJsonAnswer
sound4cl_cdef.h, 257
sound4cl_FreeMetadataInfos
sound4cl_cdef.h, 257
sound4cl_FreeParameters
sound4cl_cdef.h, 258
sound4cl_FreeParameterValue
sound4cl_cdef.h, 258
sound4cl_GetAudio
sound4cl_cdef.h, 258
sound4cl_GetAudioInputCount
sound4cl_cdef.h, 258
sound4cl_GetAudioOutputCount
sound4cl_cdef.h, 258
sound4cl_GetBufferIn
sound4cl_cdef.h, 258
sound4cl_GetBufferOut
sound4cl_cdef.h, 258
sound4cl_GetBytesFromFormat
sound4cl_cdef.h, 259
sound4cl_GetChannelCount
sound4cl_cdef.h, 259
sound4cl_GetChunkSizeInFrames
sound4cl_cdef.h, 259
sound4cl_GetEstimatedDelay

sound4cl_cdef.h, 259
sound4cl_GetFormatFromName
sound4cl_cdef.h, 259
sound4cl_GetFormatName
sound4cl_cdef.h, 259
sound4cl_GetMaxPacketFrame
sound4cl_cdef.h, 259
sound4cl_GetMetadataInfos
sound4cl_cdef.h, 260
sound4cl_GetOutputCount
sound4cl_cdef.h, 260
sound4cl_GetParameter
sound4cl_cdef.h, 260
sound4cl_GetPossibleChunkSizeInFrames
sound4cl_cdef.h, 260
sound4cl_GetProcessChunkFrames
sound4cl_cdef.h, 260
sound4cl_GetSampleRate
sound4cl_cdef.h, 260
sound4cl_GetVersion
sound4cl_cdef.h, 260
sound4cl_info
sound4cl_cdef.h, 255
sound4cl_InitProcess
sound4cl_cdef.h, 261
sound4cl_InitProcess2
sound4cl_cdef.h, 261
sound4cl_InitProcess3
sound4cl_cdef.h, 261
sound4cl_loggerfn
sound4cl_cdef.h, 253
sound4cl_LogSeverity
sound4cl_cdef.h, 254
sound4cl_MonoToStereo
sound4cl_cdef.h, 261
sound4cl_MonoToStereo_Planar
sound4cl_cdef.h, 261
sound4cl_NewBus
sound4cl_cdef.h, 262
sound4cl_NewClient
sound4cl_cdef.h, 262
sound4cl_NewParameters
sound4cl_cdef.h, 262
sound4cl_none
sound4cl_cdef.h, 255
sound4cl_PresetChange_Kind
sound4cl_cdef.h, 255
sound4cl_PresetManager_InformChange
sound4cl_cdef.h, 262
sound4cl_ProcessAudio
sound4cl_cdef.h, 262
sound4cl_ProcessAudio_Planar
sound4cl_cdef.h, 262
sound4cl_ProcessJson
sound4cl_cdef.h, 263
sound4cl_SampleFormat
sound4cl_cdef.h, 255
sound4cl_SaveState
sound4cl_cdef.h, 263
sound4cl_SetInstanceBus
sound4cl_cdef.h, 263
sound4cl_SetLoggerCallback
sound4cl_cdef.h, 263
sound4cl_SetLogSeverity
sound4cl_cdef.h, 263
sound4cl_SetMetadata
sound4cl_cdef.h, 263
sound4cl_SetMetadataMulti
sound4cl_cdef.h, 264
sound4cl_SetParameter
sound4cl_cdef.h, 264
sound4cl_SetPresetManager
sound4cl_cdef.h, 264
sound4cl_SOUND4_GetProcessInfo
sound4cl_cdef.h, 264
sound4cl_SOUND4_ProcessInfo, 184
name, 184
prefix, 184
shortname, 184
struct_size, 184
version, 185
sound4cl_StartUpdateThread
sound4cl_cdef.h, 264
sound4cl_StereoToMono
sound4cl_cdef.h, 265
sound4cl_StereoToMono_Planar
sound4cl_cdef.h, 265
sound4cl_StopUpdateThread
sound4cl_cdef.h, 265
sound4cl_storage_exists
sound4cl_cdef.h, 253
sound4cl_storage_getall
sound4cl_cdef.h, 253
sound4cl_storage_getall_free
sound4cl_cdef.h, 254
sound4cl_storage_reader
sound4cl_cdef.h, 254
sound4cl_storage_reader_free
sound4cl_cdef.h, 254
sound4cl_storage_remove
sound4cl_cdef.h, 254
sound4cl_storage_rename
sound4cl_cdef.h, 254
sound4cl_storage_writer
sound4cl_cdef.h, 254
sound4cl_TerminateProcess
sound4cl_cdef.h, 265
sound4cl_verbose
sound4cl_cdef.h, 255
sound4cl_verbose2
sound4cl_cdef.h, 255
sound4cl_verbose3
sound4cl_cdef.h, 255
sound4cl_verbose4
sound4cl_cdef.h, 255
sound4cl_verbose5

sound4cl_cdef.h, 255
sound4cl_WaitUpdateThreadReady
sound4cl_cdef.h, 265
sound4cl_warning
sound4cl_cdef.h, 255
sound4cl_Webserver
sound4cl_cdef.h, 265
sound4cl_Webserver_Status
sound4cl_cdef.h, 266
sound4cl_Webserver_Stop
sound4cl_cdef.h, 266
sound4cl_Webserver_tcp
sound4cl_cdef.h, 266
sound4cl_Webserver_tcp2
sound4cl_cdef.h, 266
StartUpdateThread
sound4::impact::dyn::CDynLib, 101
StartWebServer
sound4::CInstance, 118
sound4::impact::CInstance, 128
sound4::impact::dyn::CInstance, 138
std, 82
StereoToMono
sound4::CProcessor, 155
sound4::impact::dyn::CDynLib, 101
StereoToMono_Planar
sound4::CProcessor, 155
sound4::impact::dyn::CDynLib, 101
Stop
sound4::CInstance, 118
sound4::impact::CInstance, 128
sound4::impact::dyn::CInstance, 138
StopUpdateThread
sound4::impact::dyn::CDynLib, 101
StopWebServer
sound4::CInstance, 118
sound4::impact::CInstance, 129
sound4::impact::dyn::CInstance, 139
struct_size
SOUND4_ProcessInfo, 183
sound4cl_SOUND4_ProcessInfo, 184

Terminate
sound4::CInstance, 119
sound4::impact::CInstance, 129
sound4::impact::dyn::CInstance, 139
TerminateProcess
sound4::impact::dyn::CDynLib, 101

Update Management, 39
impact_StartUpdateThread, 39
impact_StopUpdateThread, 40
impact_WaitUpdateThreadReady, 40
Usage
sound4.impact.cl-pipe-cxx.cpp, 188
sound4.impact.cl-pipe-dyn.cpp, 192
sound4.impact.cl-pipe.cpp, 198

verbose
Global setup, 23
sound4, 73
verbose2
Global setup, 23
sound4, 73
verbose3
Global setup, 23
sound4, 74
verbose4
Global setup, 23
sound4, 74
verbose5
Global setup, 23
sound4, 74
version
SOUND4_ProcessInfo, 183
sound4cl_SOUND4_ProcessInfo, 185

WaitUpdateThreadReady
sound4::impact::dyn::CDynLib, 101
warning
Global setup, 23
sound4, 74
Web server, 61
impact_Webserver, 62
impact_Webserver_Status, 62
impact_Webserver_Stop, 63
impact_Webserver_tcp, 63
impact_Webserver_tcp2, 64
SOUND4_INVALID_WEBSERVER_ID, 61
SOUND4_WEBSERVER_HTTP_OK, 61
SOUND4_WEBSERVER_HTTPS_OK, 61
Webserver
sound4::impact::dyn::CDynLib, 102
Webserver_Status
sound4::impact::dyn::CDynLib, 102
Webserver_Stop
sound4::impact::dyn::CDynLib, 102
Webserver_tcp
sound4::impact::dyn::CDynLib, 102
Webserver_tcp2
sound4::impact::dyn::CDynLib, 102
Write
MyPresetManager, 174, 175
sound4::CPresetLoader, 141
sound4::impact::CPresetLoader, 144
sound4::impact::dyn::CPresetLoader, 146