

SOUND4 IMPACT.CL Library [1.3.5]

Generated by Doxygen 1.9.8

1 SOUND4 IMPACT.CL Library	1
1.1 General usage	1
1.2 Examples	1
2 Deprecated List	3
3 Topic Index	5
3.1 Topics	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 Topic 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 authentication 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	22
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()	25
8.3.5.10 SOUND4_GetProcessInfo()	27
8.4 Process creation/destruction	27
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()	28
8.4.2.3 impact_FreeParameterValue()	28
8.4.2.4 impact_GetParameter()	28
8.4.2.5 impact_GetProcessChunkFrames()	28
8.4.2.6 impact_InitProcess()	29
8.4.2.7 impact_InitProcess2()	29
8.4.2.8 impact_InitProcess3()	30
8.4.2.9 impact_NewParameters()	31
8.4.2.10 impact_SetParameter()	31
8.4.2.11 impact_TerminateProcess()	31
8.5 Custom preset management	32
8.5.1 Detailed Description	33
8.5.2 Typedef Documentation	33
8.5.2.1 fs_char	33
8.5.2.2 impact_storage_exists	33
8.5.2.3 impact_storage_getall	34
8.5.2.4 impact_storage_getall_free	34
8.5.2.5 impact_storage_reader	34
8.5.2.6 impact_storage_reader_free	35
8.5.2.7 impact_storage_remove	35
8.5.2.8 impact_storage_rename	35
8.5.2.9 impact_storage_writer	36
8.5.3 Enumeration Type Documentation	36
8.5.3.1 impact_PresetChange_Kind	36
8.5.4 Function Documentation	36
8.5.4.1 impact_PresetManager_InformChange()	36
8.5.4.2 impact_SetPresetManager()	37

8.6 Metadata Management	37
8.6.1 Detailed Description	37
8.6.2 Function Documentation	38
8.6.2.1 impact_FreeMetadataInfos()	38
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	44
8.9.3.1 impact_AddAudio()	44
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()	46
8.9.3.7 impact_GetMaxPacketFrame()	46
8.9.3.8 impact_GetOutputCount()	47
8.10 Audio convert helpers	47
8.10.1 Detailed Description	47
8.10.2 Function Documentation	47
8.10.2.1 impact_AudioConvertFrom()	47
8.10.2.2 impact_AudioConvertTo()	48
8.10.2.3 impact_AudioMonoFromLiveStereo()	48
8.10.2.4 impact_AudioMonoToLiveStereo()	49
8.10.2.5 impact_MonoToStereo()	49

8.10.2.6 impact_MonoToStereo_Planar()	49
8.10.2.7 impact_StereoToMono()	50
8.10.2.8 impact_StereoToMono_Planar()	50
8.11 Client's JSon config interfacing	51
8.11.1 Detailed Description	51
8.11.2 Function Documentation	51
8.11.2.1 impact_DeleteClient()	51
8.11.2.2 impact_FreeJsonAnswer()	51
8.11.2.3 impact_NewClient()	51
8.11.2.4 impact_ProcessJson()	52
8.11.2.5 impact_SaveState()	52
8.12 JSON syntax	53
8.12.1 Commands	53
8.12.2 Answers	54
8.12.3 Examples	54
8.12.3.1 Get status	54
8.12.3.2 Get all changes since last request	54
8.12.3.3 Check license	54
8.12.3.4 Set Processor name	55
8.12.3.5 Get presetlist	55
8.12.3.6 Load a preset	55
8.12.3.7 Get information on preset on air	55
8.12.3.8 Set input gain	55
8.12.3.9 Set bypass	55
8.12.3.10 Export a preset	55
8.12.3.11 Import a preset	56
8.13 JSON syntax for internal setup	56
8.13.1 Get process and modules information	56
8.13.2 Authentication management	56
8.13.2.1 Difference between setup and admin account	56
8.13.2.2 Access encoding	57
8.13.2.3 Change setup account	57
8.13.2.4 Get user info	57
8.13.2.5 Add or modify a user	57
8.13.2.6 Removes a user	58
8.13.2.7 List users	58
8.13.2.8 Get/set unlogged access	58
8.13.2.9 Get realm	58
8.13.2.10 Get API version	58
8.13.2.11 Get setup is admin	58
8.13.2.12 Get/set setup account	58
8.13.2.13 Get/set custom storage	58

8.13.3 Get process' interface description	59
8.13.3.1 Get XML decription	59
8.13.3.2 Get Process Image	59
8.13.3.3 Get Process Ico	59
8.14 Web server	59
8.14.1 Detailed Description	60
8.14.2 Macro Definition Documentation	60
8.14.2.1 SOUND4_INVALID_WEBSERVER_ID	60
8.14.2.2 SOUND4_WEBSERVER_HTTP_OK	60
8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK	60
8.14.3 Function Documentation	60
8.14.3.1 impact_Webserver()	60
8.14.3.2 impact_Webserver_FreeString()	61
8.14.3.3 impact_Webserver_GetAppHealth()	61
8.14.3.4 impact_Webserver_SetAppHealth()	61
8.14.3.5 impact_Webserver_Status()	63
8.14.3.6 impact_Webserver_Stop()	63
8.14.3.7 impact_Webserver_tcp()	64
8.14.3.8 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	66
8.16 Sample using C++ interface wrapper with dynamic library loading	66
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	67
8.16.7 Custom preset manager	67
8.16.8 Custom preset manager loading	68
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	69
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	72
9.2.3.1 error	72
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	73
9.2.3.7 verbose3	73
9.2.3.8 verbose4	73
9.2.3.9 verbose5	73
9.2.3.10 warning	74
9.3 sound4::helper Namespace Reference	74
9.3.1 Detailed Description	74
9.3.2 Typedef Documentation	74
9.3.2.1 DynFunc_t	74
9.3.2.2 DynLib_t	74
9.4 sound4::impact Namespace Reference	75
9.4.1 Detailed Description	75
9.4.2 Typedef Documentation	75
9.4.2.1 log_cb_t	75
9.4.2.2 LogSeverity	75
9.4.2.3 PresetChange_Kind	76
9.4.2.4 SampleFormat	76
9.4.3 Variable Documentation	76
9.4.3.1 ChunkFrames	76
9.4.3.2 InputChannels	76
9.4.3.3 InputSampleSize	76
9.4.3.4 OutputChannels	76
9.4.3.5 OutputSampleSize	77
9.4.3.6 SampleRate	77
9.5 sound4::impact::dyn Namespace Reference	77

9.5.1 Typedef Documentation	78
9.5.1.1 log_cb_t	78
9.5.1.2 LogSeverity	78
9.5.1.3 PresetChange_Kind	78
9.5.1.4 SampleFormat	78
9.5.2 Variable Documentation	78
9.5.2.1 ChunkFrames	78
9.5.2.2 InputChannels	78
9.5.2.3 InputSampleSize	79
9.5.2.4 OutputChannels	79
9.5.2.5 OutputSampleSize	79
9.5.2.6 SampleRate	79
9.6 sound4::impact::dyn::helper Namespace Reference	79
9.6.1 Detailed Description	80
9.6.2 Typedef Documentation	80
9.6.2.1 DynFunc_t	80
9.6.2.2 DynLib_t	80
9.7 sound4::impact::helper Namespace Reference	80
9.7.1 Detailed Description	80
9.8 std Namespace Reference	80
10 Class Documentation	81
10.1 sound4::CBus Class Reference	81
10.1.1 Detailed Description	81
10.1.2 Constructor & Destructor Documentation	81
10.1.2.1 CBus()	81
10.1.2.2 ~CBus()	81
10.1.3 Member Function Documentation	82
10.1.3.1 Get()	82
10.2 sound4::CInstance::CClient Class Reference	82
10.2.1 Detailed Description	82
10.2.2 Constructor & Destructor Documentation	82
10.2.2.1 CClient() [1/2]	82
10.2.2.2 CClient() [2/2]	83
10.2.2.3 ~CClient()	83
10.2.3 Member Function Documentation	83
10.2.3.1 operator=()	83
10.2.3.2 ProcessJson()	83
10.3 sound4::impact::CInstance::CClient Class Reference	84
10.3.1 Detailed Description	84
10.3.2 Constructor & Destructor Documentation	84
10.3.2.1 CClient() [1/2]	84

10.3.2.2 CClient() [2/2]	84
10.3.2.3 ~CClient()	84
10.3.3 Member Function Documentation	84
10.3.3.1 operator=()	84
10.3.3.2 ProcessJson()	84
10.4 sound4::impact::dyn::CInstance::CClient Class Reference	85
10.4.1 Detailed Description	85
10.4.2 Constructor & Destructor Documentation	86
10.4.2.1 CClient() [1/2]	86
10.4.2.2 CClient() [2/2]	86
10.4.2.3 ~CClient()	86
10.4.3 Member Function Documentation	86
10.4.3.1 operator=()	86
10.4.3.2 ProcessJson()	86
10.5 sound4::impact::dyn::CDynLib Class Reference	87
10.5.1 Detailed Description	88
10.5.2 Constructor & Destructor Documentation	88
10.5.2.1 CDynLib()	88
10.5.3 Member Function Documentation	89
10.5.3.1 IsOk()	89
10.5.3.2 Load()	89
10.5.4 Member Data Documentation	89
10.5.4.1 AddAudio	89
10.5.4.2 AddPadAudio	89
10.5.4.3 AudioConvertFrom	90
10.5.4.4 AudioConvertTo	90
10.5.4.5 AudioMonoFromLiveStereo	90
10.5.4.6 AudioMonoToLiveStereo	90
10.5.4.7 DeleteClient	90
10.5.4.8 ExitProcess	90
10.5.4.9 FreeJsonAnswer	90
10.5.4.10 FreeMetadataInfos	91
10.5.4.11 FreeParameters	91
10.5.4.12 FreeParameterValue	91
10.5.4.13 GetAudio	91
10.5.4.14 GetAudioInputCount	91
10.5.4.15 GetAudioOutputCount	91
10.5.4.16 GetBufferIn	91
10.5.4.17 GetBufferOut	92
10.5.4.18 GetBytesFromFormat	92
10.5.4.19 GetChannelCount	92
10.5.4.20 GetChunkSizeInFrames	92

10.5.4.21 GetEstimatedDelay	92
10.5.4.22 GetFormatFromName	92
10.5.4.23 GetFormatName	92
10.5.4.24 GetMaxPacketFrame	93
10.5.4.25 GetMetadataInfos	93
10.5.4.26 GetOutputCount	93
10.5.4.27 GetParameter	93
10.5.4.28 GetPossibleChunkSizeInFrames	93
10.5.4.29 GetProcessChunkFrames	93
10.5.4.30 GetSampleRate	93
10.5.4.31 GetVersion	94
10.5.4.32 InitProcess	94
10.5.4.33 InitProcess2	94
10.5.4.34 InitProcess3	94
10.5.4.35 MonoToStereo	94
10.5.4.36 MonoToStereo_Planar	94
10.5.4.37 NewClient	94
10.5.4.38 NewParameters	95
10.5.4.39 PresetManager_InformChange	95
10.5.4.40 ProcessAudio	95
10.5.4.41 ProcessAudio_Planar	95
10.5.4.42 ProcessJson	95
10.5.4.43 SaveState	95
10.5.4.44 SetLoggerCallback	95
10.5.4.45 SetLogSeverity	96
10.5.4.46 SetMetadata	96
10.5.4.47 SetMetadataMulti	96
10.5.4.48 SetParameter	96
10.5.4.49 SetPresetManager	96
10.5.4.50 StartUpdateThread	96
10.5.4.51 StereoToMono	96
10.5.4.52 StereoToMono_Planar	97
10.5.4.53 StopUpdateThread	97
10.5.4.54 TerminateProcess	97
10.5.4.55 WaitUpdateThreadReady	97
10.5.4.56 Webserver	97
10.5.4.57 Webserver_FreeString	97
10.5.4.58 Webserver_GetAppHealth	97
10.5.4.59 Webserver_SetAppHealth	98
10.5.4.60 Webserver_Status	98
10.5.4.61 Webserver_Stop	98
10.5.4.62 Webserver_tcp	98

10.5.4.63 Webserver_tcp2	98
10.6 sound4::helper::CDynLoader Class Reference	98
10.6.1 Detailed Description	99
10.6.2 Constructor & Destructor Documentation	99
10.6.2.1 CDynLoader() [1/3]	99
10.6.2.2 CDynLoader() [2/3]	99
10.6.2.3 CDynLoader() [3/3]	99
10.6.2.4 ~CDynLoader()	99
10.6.3 Member Function Documentation	100
10.6.3.1 Close()	100
10.6.3.2 GetSymbol()	100
10.6.3.3 GetSymbol_safe()	100
10.6.3.4 GetThisLibraryPath()	100
10.6.3.5 IsOk()	100
10.6.3.6 Load()	100
10.6.3.7 operator bool()	101
10.6.3.8 operator=()	101
10.7 sound4::impact::dyn::helper::CDynLoader Class Reference	101
10.7.1 Detailed Description	101
10.7.2 Constructor & Destructor Documentation	102
10.7.2.1 CDynLoader() [1/3]	102
10.7.2.2 CDynLoader() [2/3]	102
10.7.2.3 CDynLoader() [3/3]	102
10.7.2.4 ~CDynLoader()	102
10.7.3 Member Function Documentation	102
10.7.3.1 Close()	102
10.7.3.2 GetSymbol()	102
10.7.3.3 GetSymbol_safe()	102
10.7.3.4 GetThisLibraryPath()	103
10.7.3.5 IsOk()	103
10.7.3.6 Load()	103
10.7.3.7 operator bool()	103
10.7.3.8 operator=()	103
10.8 sound4::CInstance Class Reference	103
10.8.1 Detailed Description	104
10.8.2 Constructor & Destructor Documentation	105
10.8.2.1 CInstance() [1/2]	105
10.8.2.2 CInstance() [2/2]	105
10.8.2.3 ~CInstance()	105
10.8.3 Member Function Documentation	105
10.8.3.1 AddAudio()	105
10.8.3.2 Create()	105

10.8.3.3 GetAudio()	106
10.8.3.4 GetBufferIn()	106
10.8.3.5 GetBufferOut()	106
10.8.3.6 GetChunkFrames()	107
10.8.3.7 GetEstimatedDelay()	107
10.8.3.8 GetMetadataInfos()	107
10.8.3.9 GetParam()	107
10.8.3.10 GetWebServerAppHealth()	108
10.8.3.11 IsOk()	108
10.8.3.12 NewClient()	108
10.8.3.13 OnUpdateThreadStart()	108
10.8.3.14 OnUpdateThreadStop()	108
10.8.3.15 operator=()	108
10.8.3.16 PresetManager_InformChange()	108
10.8.3.17 ProcessAnyAudio()	109
10.8.3.18 ProcessAudio()	109
10.8.3.19 ProcessAudio_Planar()	110
10.8.3.20 SetBus()	110
10.8.3.21 SetMetadata()	110
10.8.3.22 SetMetadataMulti()	111
10.8.3.23 SetParam()	111
10.8.3.24 SetPresetManager()	111
10.8.3.25 SetWebServerAppHealth()	112
10.8.3.26 StartWebServer()	112
10.8.3.27 Stop()	112
10.8.3.28 StopWebServer()	113
10.8.3.29 Terminate()	113
10.9 sound4::impact::CInstance Class Reference	113
10.9.1 Detailed Description	114
10.9.2 Constructor & Destructor Documentation	115
10.9.2.1 CInstance() [1/2]	115
10.9.2.2 CInstance() [2/2]	115
10.9.2.3 ~CInstance()	115
10.9.3 Member Function Documentation	115
10.9.3.1 AddAudio()	115
10.9.3.2 Create()	115
10.9.3.3 GetAudio()	116
10.9.3.4 GetBufferIn()	116
10.9.3.5 GetBufferOut()	116
10.9.3.6 GetChunkFrames()	117
10.9.3.7 GetEstimatedDelay()	117
10.9.3.8 GetMetadataInfos()	117

10.9.3.9	GetParam()	117
10.9.3.10	GetWebServerAppHealth()	118
10.9.3.11	IsOk()	118
10.9.3.12	NewClient()	118
10.9.3.13	OnUpdateThreadStart()	118
10.9.3.14	OnUpdateThreadStop()	118
10.9.3.15	operator=()	118
10.9.3.16	PresetManager_InformChange()	118
10.9.3.17	ProcessAnyAudio()	119
10.9.3.18	ProcessAudio()	119
10.9.3.19	ProcessAudio_Planar()	120
10.9.3.20	SetMetadata()	120
10.9.3.21	SetMetadataMulti()	120
10.9.3.22	SetParam()	121
10.9.3.23	SetPresetManager()	121
10.9.3.24	SetWebServerAppHealth()	121
10.9.3.25	StartWebServer()	122
10.9.3.26	Stop()	122
10.9.3.27	StopWebServer()	122
10.9.3.28	Terminate()	123
10.10	sound4::impact::dyn::CInstance Class Reference	123
10.10.1	Detailed Description	124
10.10.2	Constructor & Destructor Documentation	124
10.10.2.1	CInstance() [1/2]	124
10.10.2.2	CInstance() [2/2]	124
10.10.2.3	~CInstance()	124
10.10.3	Member Function Documentation	124
10.10.3.1	AddAudio()	124
10.10.3.2	Create()	125
10.10.3.3	GetAudio()	125
10.10.3.4	GetBufferIn()	125
10.10.3.5	GetBufferOut()	126
10.10.3.6	GetChunkFrames()	126
10.10.3.7	GetEstimatedDelay()	126
10.10.3.8	GetMetadataInfos()	126
10.10.3.9	GetParam()	127
10.10.3.10	GetWebServerAppHealth()	127
10.10.3.11	IsOk()	127
10.10.3.12	NewClient()	127
10.10.3.13	OnUpdateThreadStart()	128
10.10.3.14	OnUpdateThreadStop()	128
10.10.3.15	operator=()	128

10.10.3.16 PresetManager_InformChange()	128
10.10.3.17 ProcessAnyAudio()	128
10.10.3.18 ProcessAudio()	129
10.10.3.19 ProcessAudio_Planar()	129
10.10.3.20 SetMetadata()	130
10.10.3.21 SetMetadataMulti()	130
10.10.3.22 SetParam()	130
10.10.3.23 SetPresetManager()	131
10.10.3.24 SetWebServerAppHealth()	131
10.10.3.25 StartWebServer()	131
10.10.3.26 Stop()	132
10.10.3.27 StopWebServer()	132
10.10.3.28 Terminate()	132
10.11 sound4::CPresetLoader Class Reference	132
10.11.1 Detailed Description	133
10.11.2 Constructor & Destructor Documentation	133
10.11.2.1 CPresetLoader()	133
10.11.2.2 ~CPresetLoader()	133
10.11.3 Member Function Documentation	133
10.11.3.1 Exists()	133
10.11.3.2 GetAll()	133
10.11.3.3 IsReadOnly()	133
10.11.3.4 Read()	134
10.11.3.5 Remove()	134
10.11.3.6 Rename()	134
10.11.3.7 Write()	134
10.12 sound4::impact::CPresetLoader Class Reference	134
10.12.1 Detailed Description	135
10.12.2 Constructor & Destructor Documentation	135
10.12.2.1 CPresetLoader()	135
10.12.2.2 ~CPresetLoader()	135
10.12.3 Member Function Documentation	135
10.12.3.1 Exists()	135
10.12.3.2 GetAll()	135
10.12.3.3 IsReadOnly()	135
10.12.3.4 Read()	135
10.12.3.5 Remove()	135
10.12.3.6 Rename()	136
10.12.3.7 Write()	136
10.13 sound4::impact::dyn::CPresetLoader Class Reference	136
10.13.1 Detailed Description	136
10.13.2 Constructor & Destructor Documentation	137

10.13.2.1 CPresetLoader()	137
10.13.2.2 ~CpresetLoader()	137
10.13.3 Member Function Documentation	137
10.13.3.1 Exists()	137
10.13.3.2 GetAll()	137
10.13.3.3 IsReadOnly()	137
10.13.3.4 Read()	137
10.13.3.5 Remove()	137
10.13.3.6 Rename()	138
10.13.3.7 Write()	138
10.14 sound4::CProcessor Class Reference	138
10.14.1 Detailed Description	140
10.14.2 Constructor & Destructor Documentation	140
10.14.2.1 CProcessor()	140
10.14.3 Member Function Documentation	140
10.14.3.1 AudioConvertFrom()	140
10.14.3.2 AudioConvertTo()	141
10.14.3.3 AudioMonoFromLiveStereo()	141
10.14.3.4 AudioMonoToLiveStereo()	141
10.14.3.5 GetAudioInputCount()	141
10.14.3.6 GetAudioOutputCount()	141
10.14.3.7 GetBytesFromFormat()	142
10.14.3.8 GetChannelCount()	142
10.14.3.9 GetChunkSizeInFrames()	142
10.14.3.10 GetFormatFromName()	142
10.14.3.11 GetFormatName()	143
10.14.3.12 GetPossibleChunkSizeInFrames()	143
10.14.3.13 GetPrefixSymbol()	144
10.14.3.14 GetSampleRate()	144
10.14.3.15 GetVersion()	144
10.14.3.16 IsOk()	144
10.14.3.17 Load()	144
10.14.3.18 MonoToStereo()	145
10.14.3.19 MonoToStereo_Planar()	145
10.14.3.20 SetLoggerCallback()	145
10.14.3.21 SetLogSeverity()	145
10.14.3.22 StereoToMono()	146
10.14.3.23 StereoToMono_Planar()	146
10.14.4 Friends And Related Symbol Documentation	146
10.14.4.1 CBus	146
10.14.4.2 CInstance	146
10.14.5 Member Data Documentation	146

10.14.5.1 fnAddAudio	146
10.14.5.2 fnAddPadAudio	146
10.14.5.3 fnAudioConvertFrom	147
10.14.5.4 fnAudioConvertTo	147
10.14.5.5 fnAudioMonoFromLiveStereo	147
10.14.5.6 fnAudioMonoToLiveStereo	147
10.14.5.7 fnDeleteClient	147
10.14.5.8 fnExitProcess	147
10.14.5.9 fnFreeBus	147
10.14.5.10 fnFreeJsonAnswer	148
10.14.5.11 fnFreeMetadataInfos	148
10.14.5.12 fnFreeParameters	148
10.14.5.13 fnFreeParameterValue	148
10.14.5.14 fnGetAudio	148
10.14.5.15 fnGetAudioInputCount	148
10.14.5.16 fnGetAudioOutputCount	148
10.14.5.17 fnGetBufferIn	149
10.14.5.18 fnGetBufferOut	149
10.14.5.19 fnGetBytesFromFormat	149
10.14.5.20 fnGetChannelCount	149
10.14.5.21 fnGetChunkSizeInFrames	149
10.14.5.22 fnGetEstimatedDelay	149
10.14.5.23 fnGetFormatFromName	149
10.14.5.24 fnGetFormatName	150
10.14.5.25 fnGetMaxPacketFrame	150
10.14.5.26 fnGetMetadataInfos	150
10.14.5.27 fnGetOutputCount	150
10.14.5.28 fnGetParameter	150
10.14.5.29 fnGetPossibleChunkSizeInFrames	150
10.14.5.30 fnGetProcessChunkFrames	150
10.14.5.31 fnGetSampleRate	151
10.14.5.32 fnGetVersion	151
10.14.5.33 fnInitProcess	151
10.14.5.34 fnInitProcess2	151
10.14.5.35 fnInitProcess3	151
10.14.5.36 fnMonoToStereo	151
10.14.5.37 fnMonoToStereo_Planar	151
10.14.5.38 fnNewBus	152
10.14.5.39 fnNewClient	152
10.14.5.40 fnNewParameters	152
10.14.5.41 fnPresetManager_InformChange	152
10.14.5.42 fnProcessAudio	152

10.14.5.43 fnProcessAudio_Planar	152
10.14.5.44 fnProcessJson	152
10.14.5.45 fnSaveState	153
10.14.5.46 fnSetInstanceBus	153
10.14.5.47 fnSetLoggerCallback	153
10.14.5.48 fnSetLogSeverity	153
10.14.5.49 fnSetMetadata	153
10.14.5.50 fnSetMetadataMulti	153
10.14.5.51 fnSetParameter	153
10.14.5.52 fnSetPresetManager	154
10.14.5.53 fnStartUpdateThread	154
10.14.5.54 fnStereoToMono	154
10.14.5.55 fnStereoToMono_Planar	154
10.14.5.56 fnStopUpdateThread	154
10.14.5.57 fnTerminateProcess	154
10.14.5.58 fnWaitUpdateThreadReady	154
10.14.5.59 fnWebserver	155
10.14.5.60 fnWebserver_FreeString	155
10.14.5.61 fnWebserver_GetAppHealth	155
10.14.5.62 fnWebserver_SetAppHealth	155
10.14.5.63 fnWebserver_Status	155
10.14.5.64 fnWebserver_Stop	155
10.14.5.65 fnWebserver_tcp	155
10.14.5.66 fnWebserver_tcp2	156
10.15 sound4::helper::DynFuncHolder< T > Class Template Reference	156
10.15.1 Detailed Description	156
10.15.2 Constructor & Destructor Documentation	156
10.15.2.1 DynFuncHolder() [1/2]	156
10.15.2.2 DynFuncHolder() [2/2]	157
10.15.3 Member Function Documentation	157
10.15.3.1 IsOk()	157
10.15.3.2 operator bool()	157
10.15.3.3 operator T*()	157
10.16 sound4::impact::dyn::helper::DynFuncHolder< T > Class Template Reference	157
10.16.1 Detailed Description	157
10.16.2 Constructor & Destructor Documentation	158
10.16.2.1 DynFuncHolder() [1/2]	158
10.16.2.2 DynFuncHolder() [2/2]	158
10.16.3 Member Function Documentation	158
10.16.3.1 IsOk()	158
10.16.3.2 operator bool()	158
10.16.3.3 operator T*()	158

10.17 impact_CClientInstance Struct Reference	159
10.17.1 Detailed Description	159
10.18 impact_CInstance Struct Reference	159
10.18.1 Detailed Description	159
10.19 impact_CParameters Struct Reference	159
10.19.1 Detailed Description	159
10.20 MyPresetManager Class Reference	159
10.20.1 Detailed Description	160
10.20.2 Constructor & Destructor Documentation	160
10.20.2.1 MyPresetManager() [1/2]	160
10.20.2.2 ~MyPresetManager() [1/2]	160
10.20.2.3 MyPresetManager() [2/2]	161
10.20.2.4 ~MyPresetManager() [2/2]	161
10.20.3 Member Function Documentation	161
10.20.3.1 Exists() [1/2]	161
10.20.3.2 Exists() [2/2]	161
10.20.3.3 GetAll() [1/2]	161
10.20.3.4 GetAll() [2/2]	161
10.20.3.5 IsReadOnly() [1/2]	162
10.20.3.6 IsReadOnly() [2/2]	162
10.20.3.7 Read() [1/2]	162
10.20.3.8 Read() [2/2]	162
10.20.3.9 Remove() [1/2]	162
10.20.3.10 Remove() [2/2]	162
10.20.3.11 Rename() [1/2]	163
10.20.3.12 Rename() [2/2]	163
10.20.3.13 SetupPath() [1/2]	163
10.20.3.14 SetupPath() [2/2]	163
10.20.3.15 Write() [1/2]	163
10.20.3.16 Write() [2/2]	164
10.21 sound4::helper::SampleFormat< T > Struct Template Reference	164
10.21.1 Detailed Description	164
10.22 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference	164
10.22.1 Detailed Description	164
10.23 sound4::impact::helper::SampleFormat< T > Struct Template Reference	165
10.23.1 Detailed Description	165
10.24 sound4::helper::SampleFormat< float > Struct Reference	165
10.24.1 Detailed Description	165
10.24.2 Member Data Documentation	166
10.24.2.1 format	166
10.25 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference	166
10.25.1 Detailed Description	166

10.25.2 Member Data Documentation	166
10.25.2.1 format	166
10.26 sound4::impact::helper::SampleFormat< float > Struct Reference	166
10.26.1 Detailed Description	167
10.26.2 Member Data Documentation	167
10.26.2.1 format	167
10.27 sound4::helper::SampleFormat< int16_t > Struct Reference	167
10.27.1 Detailed Description	167
10.27.2 Member Data Documentation	167
10.27.2.1 format	167
10.28 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference	167
10.28.1 Detailed Description	168
10.28.2 Member Data Documentation	168
10.28.2.1 format	168
10.29 sound4::impact::helper::SampleFormat< int16_t > Struct Reference	168
10.29.1 Detailed Description	168
10.29.2 Member Data Documentation	168
10.29.2.1 format	168
10.30 sound4::helper::SampleFormat< int32_t > Struct Reference	169
10.30.1 Detailed Description	169
10.30.2 Member Data Documentation	169
10.30.2.1 format	169
10.31 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference	169
10.31.1 Detailed Description	169
10.31.2 Member Data Documentation	170
10.31.2.1 format	170
10.32 sound4::impact::helper::SampleFormat< int32_t > Struct Reference	170
10.32.1 Detailed Description	170
10.32.2 Member Data Documentation	170
10.32.2.1 format	170
10.33 SOUND4_ProcessInfo Struct Reference	170
10.33.1 Detailed Description	171
10.33.2 Member Data Documentation	171
10.33.2.1 name	171
10.33.2.2 prefix	171
10.33.2.3 shortname	171
10.33.2.4 struct_size	171
10.33.2.5 version	172
10.34 sound4cl_SOUND4_ProcessInfo Struct Reference	172
10.34.1 Detailed Description	172
10.34.2 Member Data Documentation	172
10.34.2.1 name	172

10.34.2.2 prefix	172
10.34.2.3 shortname	173
10.34.2.4 struct_size	173
10.34.2.5 version	173
11 File Documentation	175
11.1 sound4.impact.cl-pipe-cxx.cpp File Reference	175
11.1.1 Function Documentation	175
11.1.1.1 GetEnv()	175
11.1.1.2 main()	176
11.1.1.3 MyLogger()	176
11.1.1.4 ProcessLoop()	176
11.1.1.5 Usage()	176
11.2 sound4.impact.cl-pipe-cxx.cpp	177
11.3 sound4.impact.cl-pipe-dyn.cpp File Reference	179
11.3.1 Function Documentation	179
11.3.1.1 GetEnv()	179
11.3.1.2 main()	180
11.3.1.3 MyLogger()	180
11.3.1.4 ProcessLoop()	180
11.3.1.5 Usage()	180
11.4 sound4.impact.cl-pipe-dyn.cpp	181
11.5 sound4.impact.cl-pipe.cpp File Reference	184
11.5.1 Macro Definition Documentation	184
11.5.1.1 CHANNEL_COUNT	184
11.5.1.2 CHUNK_SIZE	185
11.5.1.3 SAMPLE_RATE	185
11.5.2 Function Documentation	185
11.5.2.1 main()	185
11.5.2.2 MyLogger()	185
11.5.2.3 ProcessLoop()	185
11.5.2.4 Usage()	186
11.6 sound4.impact.cl-pipe.cpp	186
11.7 sound4cl-pipe.cpp File Reference	189
11.7.1 Function Documentation	189
11.7.1.1 GetEnv()	189
11.7.1.2 main()	190
11.7.1.3 ProcessLoop()	190
11.8 sound4cl-pipe.cpp	191
11.9 sound4.impact.cl.h File Reference	194
11.9.1 Macro Definition Documentation	196
11.9.1.1 IMPACT_DECL	196

11.9.1.2 IMPACT_HAS_CLOUDBUS	197
11.9.1.3 IMPACT_HAS_WEBSERVER	197
11.10 sound4.impact.cl.h	197
11.11 sound4.impact.cl.hpp File Reference	201
11.11.1 Macro Definition Documentation	202
11.11.1.1 fs_strdup	202
11.12 sound4.impact.cl.hpp	202
11.13 sound4.impact.cl_dyn.hpp File Reference	210
11.13.1 Detailed Description	211
11.13.2 Macro Definition Documentation	211
11.13.2.1 fs_strdup	211
11.13.2.2 UNIXLIKE	211
11.14 sound4.impact.cl_dyn.hpp	212
11.15 sound4cl.hpp File Reference	225
11.15.1 Detailed Description	226
11.15.2 Macro Definition Documentation	226
11.15.2.1 UNIXLIKE	226
11.16 sound4cl.hpp	227
11.17 sound4cl_cdef.h File Reference	239
11.17.1 Macro Definition Documentation	242
11.17.1.1 fs_strdup	242
11.17.1.2 SOUND4_INVALID_WEBSERVER_ID	242
11.17.1.3 SOUND4_WEBSERVER_HTTP_OK	243
11.17.1.4 SOUND4_WEBSERVER_HTTPS_OK	243
11.17.2 Typedef Documentation	243
11.17.2.1 fs_char	243
11.17.2.2 sound4cl_loggerfn	243
11.17.2.3 sound4cl_storage_exists	243
11.17.2.4 sound4cl_storage_getall	243
11.17.2.5 sound4cl_storage_getall_free	243
11.17.2.6 sound4cl_storage_reader	244
11.17.2.7 sound4cl_storage_reader_free	244
11.17.2.8 sound4cl_storage_remove	244
11.17.2.9 sound4cl_storage_rename	244
11.17.2.10 sound4cl_storage_writer	244
11.17.3 Enumeration Type Documentation	244
11.17.3.1 sound4cl_LogSeverity	244
11.17.3.2 sound4cl_PresetChange_Kind	245
11.17.3.3 sound4cl_SampleFormat	245
11.17.4 Function Documentation	245
11.17.4.1 sound4cl_AddAudio()	245
11.17.4.2 sound4cl_AddPadAudio()	246

11.17.4.3 sound4cl_AudioConvertFrom()	246
11.17.4.4 sound4cl_AudioConvertTo()	246
11.17.4.5 sound4cl_AudioMonoFromLiveStereo()	246
11.17.4.6 sound4cl_AudioMonoToLiveStereo()	246
11.17.4.7 sound4cl_DeleteClient()	246
11.17.4.8 sound4cl_ExitProcess()	246
11.17.4.9 sound4cl_FreeBus()	246
11.17.4.10 sound4cl_FreeJsonAnswer()	247
11.17.4.11 sound4cl_FreeMetadataInfos()	247
11.17.4.12 sound4cl_FreeParameters()	247
11.17.4.13 sound4cl_FreeParameterValue()	247
11.17.4.14 sound4cl_GetAudio()	247
11.17.4.15 sound4cl_GetAudioInputCount()	247
11.17.4.16 sound4cl_GetAudioOutputCount()	247
11.17.4.17 sound4cl_GetBufferIn()	247
11.17.4.18 sound4cl_GetBufferOut()	248
11.17.4.19 sound4cl_GetBytesFromFormat()	248
11.17.4.20 sound4cl_GetChannelCount()	248
11.17.4.21 sound4cl_GetChunkSizeInFrames()	248
11.17.4.22 sound4cl_GetEstimatedDelay()	248
11.17.4.23 sound4cl_GetFormatFromName()	248
11.17.4.24 sound4cl_GetFormatName()	248
11.17.4.25 sound4cl_GetMaxPacketFrame()	248
11.17.4.26 sound4cl_GetMetadataInfos()	248
11.17.4.27 sound4cl_GetOutputCount()	249
11.17.4.28 sound4cl_GetParameter()	249
11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames()	249
11.17.4.30 sound4cl_GetProcessChunkFrames()	249
11.17.4.31 sound4cl_GetSampleRate()	249
11.17.4.32 sound4cl_GetVersion()	249
11.17.4.33 sound4cl_InitProcess()	249
11.17.4.34 sound4cl_InitProcess2()	249
11.17.4.35 sound4cl_InitProcess3()	250
11.17.4.36 sound4cl_MonoToStereo()	250
11.17.4.37 sound4cl_MonoToStereo_Planar()	250
11.17.4.38 sound4cl_NewBus()	250
11.17.4.39 sound4cl_NewClient()	250
11.17.4.40 sound4cl_NewParameters()	250
11.17.4.41 sound4cl_PresetManager_InformChange()	250
11.17.4.42 sound4cl_ProcessAudio()	251
11.17.4.43 sound4cl_ProcessAudio_Planar()	251
11.17.4.44 sound4cl_ProcessJson()	251

11.17.4.45 sound4cl_SaveState()	251
11.17.4.46 sound4cl_SetInstanceBus()	251
11.17.4.47 sound4cl_SetLoggerCallback()	251
11.17.4.48 sound4cl_SetLogSeverity()	251
11.17.4.49 sound4cl_SetMetadata()	251
11.17.4.50 sound4cl_SetMetadataMulti()	252
11.17.4.51 sound4cl_SetParameter()	252
11.17.4.52 sound4cl_SetPresetManager()	252
11.17.4.53 sound4cl_SOUND4_GetProcessInfo()	252
11.17.4.54 sound4cl_StartUpdateThread()	252
11.17.4.55 sound4cl_StereoToMono()	252
11.17.4.56 sound4cl_StereoToMono_Planar()	253
11.17.4.57 sound4cl_StopUpdateThread()	253
11.17.4.58 sound4cl_TerminateProcess()	253
11.17.4.59 sound4cl_WaitUpdateThreadReady()	253
11.17.4.60 sound4cl_Webserver()	253
11.17.4.61 sound4cl_Webserver_FreeString()	253
11.17.4.62 sound4cl_Webserver_GetAppHealth()	253
11.17.4.63 sound4cl_Webserver_SetAppHealth()	254
11.17.4.64 sound4cl_Webserver_Status()	254
11.17.4.65 sound4cl_Webserver_Stop()	254
11.17.4.66 sound4cl_Webserver_tcp()	254
11.17.4.67 sound4cl_Webserver_tcp2()	254
11.18 sound4cl_cdef.h	254

Index

259

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

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Sample using C interface	17
Environment variables and parameters	19
Global setup	20
Process creation/destruction	27
Custom preset managagement	32
Metadata Management	37
Update Management	39
Direct processing	40
Audio managing (any audio)	43
Audio convert helpers	47
Client's JSoN config interfacing	51
JSON syntax	53
JSON syntax for internal setup	56
Web server	59
Sample using C++ interface wrapper	65
Sample using C++ interface wrapper with dynamic library loading	66
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	77
sound4::impact::dyn::helper	79
sound4::impact::helper	80
std	80

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

sound4::CBus	81
sound4::CInstance::CClient	82
sound4::impact::CInstance::CClient	84
sound4::impact::dyn::CInstance::CClient	85
sound4::impact::dyn::CDynLib	87
sound4::helper::CDynLoader	98
sound4::impact::dyn::helper::CDynLoader	101
sound4::CInstance	103
sound4::impact::CInstance	113
sound4::impact::dyn::CInstance	123
sound4::CPresetLoader	132
MyPresetManager	159
sound4::impact::CPresetLoader	134
sound4::impact::dyn::CPresetLoader	136
MyPresetManager	159
sound4::CProcessor	138
sound4::helper::DynFuncHolder< T >	156
sound4::impact::dyn::helper::DynFuncHolder< T >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddAudio) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddPadAudio) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertFrom) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertTo) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_DeleteClient) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ExitProcess) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameters) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameterValue) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudio) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioInputCount) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferIn) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferOut) >	157

sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChannelCount) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatFromName) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatName) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMetadataInfos) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetOutputCount) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetParameter) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetSampleRate) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetVersion) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess2) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess3) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewClient) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewParameters) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_PresetManager_InformChange) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessJson) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SaveState) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLoggerCallback) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLogSeverity) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadata) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadataMulti) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetParameter) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetPresetManager) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StartUpdateThread) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StopUpdateThread) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_TerminateProcess) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_FreeString) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_GetAppHealth) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_SetAppHealth) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Status) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Stop) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp) >	157
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp2) >	157
sound4::helper::DynFuncHolder< decltype(sound4cl_AddAudio) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_DeleteClient) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_ExitProcess) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeBus) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameters) >	156

sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudio) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatName) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetParameter) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_GetVersion) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess2) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess3) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_NewBus) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_NewClient) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_NewParameters) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessJson) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SaveState) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadata) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetParameter) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) >	156
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) >	156
impact_CClientInstance	159
impact_CInstance	159
impact_CParameters	159
sound4::helper::SampleFormat< T >	164

sound4::impact::dyn::helper::SampleFormat< T >	164
sound4::impact::helper::SampleFormat< T >	165
sound4::helper::SampleFormat< float >	165
sound4::impact::dyn::helper::SampleFormat< float >	166
sound4::impact::helper::SampleFormat< float >	166
sound4::helper::SampleFormat< int16_t >	167
sound4::impact::dyn::helper::SampleFormat< int16_t >	167
sound4::impact::helper::SampleFormat< int16_t >	168
sound4::helper::SampleFormat< int32_t >	169
sound4::impact::dyn::helper::SampleFormat< int32_t >	169
sound4::impact::helper::SampleFormat< int32_t >	170
SOUND4_ProcessInfo	170
sound4cl_SOUND4_ProcessInfo	172

Chapter 6

Class Index

6.1 Class List

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

sound4::CBus	
Shared Bus	81
sound4::CInstance::CClient	82
sound4::impact::CInstance::CClient	84
sound4::impact::dyn::CInstance::CClient	85
sound4::impact::dyn::CDynLib	
Dynamic library interface	87
sound4::helper::CDynLoader	
Helper to load dynamic library	98
sound4::impact::dyn::helper::CDynLoader	
Helper to load dynamic library	101
sound4::CInstance	
Instance handling class	103
sound4::impact::CInstance	113
sound4::impact::dyn::CInstance	123
sound4::CPresetLoader	
Custom preset handler helper	132
sound4::impact::CPresetLoader	134
sound4::impact::dyn::CPresetLoader	136
sound4::CProcessor	
Dynamic library interface	138
sound4::helper::DynFuncHolder< T >	
Helper to wrap a function pointer	156
sound4::impact::dyn::helper::DynFuncHolder< T >	
Helper to wrap a function pointer	157
impact_CClientInstance	159
impact_CInstance	159
impact_CParameters	159
MyPresetManager	
[Custom preset manager]	159
sound4::helper::SampleFormat< T >	
Helper for Sample format types	164
sound4::impact::dyn::helper::SampleFormat< T >	
Helper for Sample format types	164
sound4::impact::helper::SampleFormat< T >	
Helper for Sample format types	165

sound4::helper::SampleFormat< float >	165
sound4::impact::dyn::helper::SampleFormat< float >	166
sound4::impact::helper::SampleFormat< float >	166
sound4::helper::SampleFormat< int16_t >	167
sound4::impact::dyn::helper::SampleFormat< int16_t >	167
sound4::impact::helper::SampleFormat< int16_t >	168
sound4::helper::SampleFormat< int32_t >	169
sound4::impact::dyn::helper::SampleFormat< int32_t >	169
sound4::impact::helper::SampleFormat< int32_t >	170
SOUND4_ProcessInfo	170
sound4cl_SOUND4_ProcessInfo	172

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	175
sound4.impact.cl-pipe-dyn.cpp	
C++ sample using C++ interface wrapper with dynamic library loading	179
sound4.impact.cl-pipe.cpp	
C++ sample using only C interface	184
sound4cl-pipe.cpp	
C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided	189
sound4.impact.cl.h	
C interface for library	194
sound4.impact.cl.hpp	
C++ wrapper for library	201
sound4.impact.cl_dyn.hpp	
C++ wrapper for dynamic loading library	210
sound4cl.hpp	
C++ wrapper for dynamic loading library	225
sound4cl_cdef.h	
C function signature definition for all sound4.cl libraries	239

Chapter 8

Topic 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];

#ifdef _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](#)

```

#ifdef (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 authentication 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_BUNDLE` : optional certificates bundle 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=boolvalue` : force the storage in read-only mode
- `STATE_RO=boolvalue` : force the state in read-only mode (but can save presets/auth)
- `WAIT_LICENSE=boolvalue` : force to wait the license load before starting (always in file mode)
- `FAST_START=boolvalue` : do not to try loading the license before starting (never in file mode)
- `LOG_HEADER=string` : use string as header for logs in instance as much as possible

Web server options:

- `COOKIE_DOMAIN=domain` : Force login cookie's domain
- `COOKIE_PATH=path` : Force login cookie's path
- `COOKIE_SECURE=mode` : Force login cookie's secure mode [true|false]
- `COOKIE_SAMESITE=mode` : Force login cookie's secure mode [default|strict|lax|none]

NOTE: boolvalue are (case insensitive):

- 0,false,no,n : disable
- 1,true,yes,y,(empty),(any) : enable

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 instances 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 immediately 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 losing 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 184 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 190 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 172 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 178 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 196 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 311 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 292 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

<i>logger</i>	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

<i>severity</i>	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

<i>instance</i>	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

<i>value</i>	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

<i>instance</i>	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 management](#).

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

<i>LoginKey</i>	SOUND4 Login key (was User ID)
<i>RadioName</i>	SOUND4 RadioName
<i>Access_Key_ID</i>	SOUND4 AWS access ID
<i>Access_Key_Secret</i>	SOUND4 AWS access Secret
<i>save_path</i>	Path where we can load/save the configuration. If NULL, memory storage will be used.
<i>parameters</i>	Parameters
<i>frames_per_chunk</i>	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

<i>instance</i>	the instance to terminate
-----------------	---------------------------

Returns

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

8.5 Custom preset management

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 overridden 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 579 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

<i>filename</i>	the name of the file to test.
<i>userdata</i>	the userdata passed to impact_SetPresetManager

Returns

-1 if not exists, 0 if it exists

Definition at line 618 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

<i>userdata</i>	the userdata passed to <code>impact_SetPresetManager</code>
-----------------	---

Returns

A NULL terminated array of file names

Definition at line 627 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

<i>all</i>	the returned value from <code>impact_storage_getall</code>
<i>userdata</i>	the userdata passed to <code>impact_SetPresetManager</code>

Definition at line 635 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

<i>filename</i>	the name of the file to read.
<i>userdata</i>	the userdata passed to <code>impact_SetPresetManager</code>

Returns

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

Definition at line 591 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

<i>content</i>	the returned value from <code>impact_storage_reader</code>
<i>userdata</i>	the userdata passed to impact_SetPresetManager

Definition at line 599 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

<i>filename</i>	the name of the file to remove.
<i>userdata</i>	the userdata passed to impact_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 644 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

<i>from</i>	The source file name (must exist)
<i>to</i>	The destination file name (must not exist)
<i>userdata</i>	the userdata passed to impact_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 654 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

<i>filename</i>	the name of the file to write.
<i>content</i>	the UTF-8 content to write to the file
<i>userdata</i>	the userdata passed to impact_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 609 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 687 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 conjunction with [impact_SetPresetManager](#)

Parameters

<i>instance</i>	the processing instance
<i>relative_path</i>	The path which changed, relative to the root storage
<i>change_kind</i>	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

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

<i>instance</i>	the instance
<i>infos</i>	the infos

8.6.2.2 `impact_GetMetadataInfos()`

```
const char ** impact_GetMetadataInfos (
    struct impact\_CInstance * instance )
```

Get information on known metadata keys.

Parameters

<i>instance</i>	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

<i>instance</i>	the instance
<i>key</i>	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.
<i>value</i>	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

<i>instance</i>	the instance
<i>keyvalue</i>	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

<i>instance</i>	the instance
<i>port</i>	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

<i>instance</i>	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

<i>instance</i>	the processing instance
<i>milliseconds</i>	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

<i>instance</i>	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

<i>instance</i>	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

Output 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

<i>instance</i>	the processing instance
<i>input</i>	audio input buffer (64 byte aligned) or NULL to use internal buffer impact_GetBufferIn
<i>output</i>	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

Output 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

<i>instance</i>	the processing instance
<i>input</i>	array of audio input buffers for each input channel (no alignment required)
<i>output</i>	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 910 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

<i>instance</i>	the processing instance
<i>payload</i>	the binary raw audio buffer
<i>nFrame</i>	the number of frames in the payload
<i>fmt</i>	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

<i>instance</i>	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

<i>instance</i>	the processing instance
<i>payload</i>	the binary raw audio buffer to store output
<i>max_nFrame</i>	the maximum number of frames that can be stored in the payload
<i>fmt</i>	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

<i>fmt</i>	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

<i>name</i>	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

<i>fmt</i>	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

<i>instance</i>	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

<i>instance</i>	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

<i>payload</i>	the binary raw audio buffer
<i>output</i>	audio output buffer (64 byte aligned)
<i>nSpl</i>	the number of samples (not frames)
<i>fmt</i>	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

<i>input</i>	audio input buffer (64 byte aligned)
<i>payload</i>	the binary raw audio buffer
<i>nSpl</i>	the number of samples (not frames)
<i>fmt</i>	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

<i>payload</i>	Livewire Live Stereo raw buffer
<i>output</i>	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

<i>input</i>	audio input mono buffer (64 byte aligned)
<i>payload</i>	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

<i>input</i>	audio input mono buffer (64 byte aligned)
<i>output</i>	audio output stereo buffer (64 byte aligned)
<i>nFrame</i>	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

<i>input</i>	audio input mono buffer (64 byte aligned)
<i>outputL</i>	audio output left buffer (64 byte aligned)
<i>outputR</i>	audio output right buffer (64 byte aligned)
<i>nFrame</i>	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

<i>input</i>	audio input stereo buffer (64 byte aligned)
<i>output</i>	audio output mono buffer (64 byte aligned)
<i>nFrame</i>	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

<i>inputL</i>	audio input left buffer (64 byte aligned)
<i>inputR</i>	audio input right buffer (64 byte aligned)
<i>output</i>	audio output mono buffer (64 byte aligned)
<i>nFrame</i>	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

<i>client</i>	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

<i>json_str</i>	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

<i>instance</i>	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

<i>client</i>	the client instance
<i>json_str</i>	the JSON request
<i>need_save</i>	on return will be 1 if 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

<i>instance</i>	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 $\leq 1.1.7$, the state resetted also for filtered out objects. So avoid alternating `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,"unlogged_access":"rx"}}
```

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":"Authentication 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)
- void [impact_Webserver_SetAppHealth](#) (struct [impact_CInstance](#) *instance, int httpcode, const char *contenttype, const char *content)
- void [impact_Webserver_GetAppHealth](#) (struct [impact_CInstance](#) *instance, int *httpcode, char **contenttype, char **content)
- void [impact_Webserver_FreeString](#) (char *str)

8.14.1 Detailed Description

Run a web server to configure the processor

The web server also expose `/health` which should always answer 200.

The web server also expose `/licvalid` which should answer 200 when license is valid, 402 when it is not valid, other standard code for other errors.

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 1591 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 1596 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 1600 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

<i>listenport</i>	The port to listen for HTTP connections (0 to disable)
<i>listenport_secure</i>	The port to listen for HTTPS connections (0 to disable)
<i>instance</i>	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_FreeString()

```
void impact_Webserver_FreeString (
    char * str )
```

Free a a string returned by [impact_Webserver_GetAppHealth](#)

Parameters

<i>str</i>	the string to free
------------	--------------------

8.14.3.3 impact_Webserver_GetAppHealth()

```
void impact_Webserver_GetAppHealth (
    struct impact_CInstance * instance,
    int * httpcode,
    char ** contenttype,
    char ** content )
```

Get the current web server /apphealth returned http code,content-type and content

Parameters

<i>instance</i>	The process instance
<i>httpcode</i>	The http code pointer
<i>contenttype</i>	The http Content-Type header pointer (to be freed with impact_Webserver_FreeString)
<i>content</i>	The http body content pointer (to be freed with impact_Webserver_FreeString)

8.14.3.4 impact_Webserver_SetAppHealth()

```
void impact_Webserver_SetAppHealth (
    struct impact_CInstance * instance,
    int httpcode,
    const char * contenttype,
    const char * content )
```

Set the web server /apphealth status info

The hosting application can use this to customize the handling of /apphealth path.

Parameters

<i>instance</i>	The process instance
<i>httpcode</i>	The http code
<i>contenttype</i>	The http Content-Type header
<i>content</i>	The http body content

8.14.3.5 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

<i>id</i>	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

8.14.3.6 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

<i>id</i>	The web server Id, returned by impact_Webserver_tcp or impact_Webserver
<i>timeout_ms</i>	The timeout to wait gracefull shutdown

Returns

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

8.14.3.7 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

<i>listenport</i>	The port to listen for HTTP connections
<i>listenport_secure</i>	The port to listen for HTTPS connections
<i>socket_ip</i>	The ip of the process to connect to
<i>socket_port</i>	The port of the process's JSON server to connect to

Returns

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

8.14.3.8 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

<i>listenport</i>	The port to listen for HTTP connections
<i>listenport_secure</i>	The port to listen for HTTPS connections
<i>socket_ip</i>	The ip of the process to connect to
<i>socket_port</i>	The port of the process's JSON server to connect to
<i>parameters</i>	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;
        std::error_code ec;
        if ( !std::filesystem::exists(m_preset_path,ec) ) {
            return false;
        }
        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;
        std::error_code ec;
        for (auto it = std::filesystem::directory_iterator(m_preset_path.c_str(),
            std::filesystem::directory_options::skip_permission_denied, ec);
            !ec && it != std::filesystem::directory_iterator{};
            it.increment(ec) )
        {
            if (it->is_regular_file(ec)) {
                list.push_back(it->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;
        std::error_code ec;
        if ( !std::filesystem::exists(m_preset_path,ec) ) {
            return false;
        }
        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;
    std::error_code ec;
    for (auto it = std::filesystem::directory_iterator(m_preset_path.c_str(),
std::filesystem::directory_options::skip_permission_denied, ec);
!ec && it != std::filesystem::directory_iterator{};
it.increment(ec) )
    {
        if (it->is_regular_file(ec)) {
            list.push_back(it->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

- namespace [helper](#)
- namespace [impact](#)

Classes

- class [CBus](#)
Shared Bus.
- class [CInstance](#)
Instance handling class.
- class [CPresetLoader](#)
Custom preset handler helper.
- class [CProcessor](#)
Dynamic library interface.

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 [258](#) of file [sound4cl.hpp](#).

9.2.2.2 LogSeverity

```
using sound4::LogSeverity = typedef sound4cl\_LogSeverity
```

Log Severity levels

Definition at line [243](#) of file [sound4cl.hpp](#).

9.2.2.3 PresetChange_Kind

```
using sound4::PresetChange\_Kind = typedef sound4cl\_PresetChange\_Kind
```

Definition at line [753](#) of file [sound4cl.hpp](#).

9.2.3 Variable Documentation

9.2.3.1 error

```
const LogSeverity sound4::error = (LogSeverity) sound4cl\_error
```

Definition at line [246](#) of file [sound4cl.hpp](#).

9.2.3.2 fatal

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

Definition at line 245 of file [sound4cl.hpp](#).

9.2.3.3 info

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

Definition at line 248 of file [sound4cl.hpp](#).

9.2.3.4 none

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

Definition at line 244 of file [sound4cl.hpp](#).

9.2.3.5 verbose

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

Definition at line 249 of file [sound4cl.hpp](#).

9.2.3.6 verbose2

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

Definition at line 250 of file [sound4cl.hpp](#).

9.2.3.7 verbose3

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

Definition at line 251 of file [sound4cl.hpp](#).

9.2.3.8 verbose4

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

Definition at line 252 of file [sound4cl.hpp](#).

9.2.3.9 verbose5

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

Definition at line 253 of file [sound4cl.hpp](#).

9.2.3.10 warning

```
const LogSeverity sound4::warning = (LogSeverity) sound4cl\_warning
```

Definition at line [247](#) of file [sound4cl.hpp](#).

9.3 sound4::helper Namespace Reference

Classes

- class [CDynLoader](#)
Helper to load dynamic library.
- class [DynFuncHolder](#)
Helper to wrap a function pointer.
- struct [SampleFormat](#)
Helper for Sample format types.
- struct [SampleFormat< float >](#)
- struct [SampleFormat< int16_t >](#)
- struct [SampleFormat< int32_t >](#)

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 [112](#) of file [sound4cl.hpp](#).

9.3.2.2 DynLib_t

```
using sound4::helper::DynLib\_t = typedef HMODULE
```

Definition at line [111](#) of file [sound4cl.hpp](#).

9.4 sound4::impact Namespace Reference

Namespaces

- namespace [dyn](#)
- namespace [helper](#)

Classes

- class [CInstance](#)
- class [CPresetLoader](#)

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](#) = 64
- const size_t [InputChannels](#) = 2 * 1
- const size_t [OutputChannels](#) = 2 * 1
- const size_t [InputSampleSize](#) = [ChunkFrames](#)*[InputChannels](#)
- const size_t [OutputSampleSize](#) = [ChunkFrames](#)*[OutputChannels](#)
- const size_t [SampleRate](#) = 48000

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 = 64
```

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 = 2 * 1
```

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 = 2 * 1
```

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 = 48000
```

Sample rate of library

Definition at line 77 of file [sound4.impact.cl.hpp](#).

9.5 sound4::impact::dyn Namespace Reference

Namespaces

- namespace [helper](#)

Classes

- class [CDynLib](#)
Dynamic library interface.
- class [CInstance](#)
- class [CPresetLoader](#)

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](#) = 64
- const size_t [InputChannels](#) = 2 * 1
- const size_t [OutputChannels](#) = 2 * 1
- const size_t [InputSampleSize](#) = [ChunkFrames](#)*[InputChannels](#)
- const size_t [OutputSampleSize](#) = [ChunkFrames](#)*[OutputChannels](#)
- const size_t [SampleRate](#) = 48000

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 743 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 739 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 811 of file [sound4.impact.cl_dyn.hpp](#).

9.5.1.4 SampleFormat

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

Definition at line 472 of file [sound4.impact.cl_dyn.hpp](#).

9.5.2 Variable Documentation

9.5.2.1 ChunkFrames

```
const size_t sound4::impact::dyn::ChunkFrames = 64
```

Number of frames in a chunk

Definition at line 476 of file [sound4.impact.cl_dyn.hpp](#).

9.5.2.2 InputChannels

```
const size_t sound4::impact::dyn::InputChannels = 2 * 1
```

Number of float values in an input frame

Definition at line 480 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 488 of file [sound4.impact.cl_dyn.hpp](#).

9.5.2.4 OutputChannels

```
const size_t sound4::impact::dyn::OutputChannels = 2 * 1
```

Number of float values in an output frame

Definition at line 484 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 492 of file [sound4.impact.cl_dyn.hpp](#).

9.5.2.6 SampleRate

```
const size_t sound4::impact::dyn::SampleRate = 48000
```

Sample rate of library

Definition at line 496 of file [sound4.impact.cl_dyn.hpp](#).

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

Classes

- class [CDynLoader](#)
Helper to load dynamic library.
- class [DynFuncHolder](#)
Helper to wrap a function pointer.
- struct [SampleFormat](#)
Helper for Sample format types.
- struct [SampleFormat< float >](#)
- struct [SampleFormat< int16_t >](#)
- struct [SampleFormat< int32_t >](#)

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 83 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 82 of file [sound4.impact.cl_dyn.hpp](#).

9.7 sound4::impact::helper Namespace Reference

Classes

- struct [SampleFormat](#)
Helper for Sample format types.
- struct [SampleFormat< float >](#)
- struct [SampleFormat< int16_t >](#)
- struct [SampleFormat< int32_t >](#)

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 [713](#) of file [sound4cl.hpp](#).

10.1.2 Constructor & Destructor Documentation

10.1.2.1 CBus()

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

Definition at line [715](#) of file [sound4cl.hpp](#).

10.1.2.2 ~CBus()

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

Definition at line [722](#) 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 729 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 1225 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 1227 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 1237 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

<i>request</i>	the JSON request.
----------------	-------------------

See also

[JSON syntax](#)

Parameters

<i>NeedSave</i>	if non null, returns if the saving is required (maybe delayed)
-----------------	--

Returns

the JSON answer

Definition at line 1251 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 827 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 829 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 838 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

<i>request</i>	the JSON request.
----------------	-------------------

See also

[JSON syntax](#)

Parameters

<i>NeedSave</i>	if non null, returns if the saving is required (maybe delayed)
-----------------	--

Returns

the JSON answer

Definition at line 852 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 1282 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 1284 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 1294 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

<i>request</i>	the JSON request.
----------------	-------------------

See also

[JSON syntax](#)

Parameters

<i>NeedSave</i>	if non null, returns if the saving is required (maybe delayed)
-----------------	--

Returns

the JSON answer

Definition at line 1308 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](#)
- [helper::DynFuncHolder< decltype\(impact_Webserver_SetAppHealth\) > Webserver_SetAppHealth](#)
- [helper::DynFuncHolder< decltype\(impact_Webserver_GetAppHealth\) > Webserver_GetAppHealth](#)
- [helper::DynFuncHolder< decltype\(impact_Webserver_FreeString\) > Webserver_FreeString](#)

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 216 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 462 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

<i>path</i>	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 318 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 253 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 254 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 257 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 258 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 261 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 262 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 264 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 240 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 266 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 283 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 233 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 236 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 256 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 227 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 228 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 246 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 247 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 251 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 226 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 225 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 248 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 250 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 249 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 252 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 282 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.26 GetOutputCount

```
helper::DynFuncHolder< decltype(impact_GetOutputCount) > sound4::impact::dyn::CDynLib::GetOutputCount
```

Definition at line 255 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.27 GetParameter

```
helper::DynFuncHolder< decltype(impact_GetParameter) > sound4::impact::dyn::CDynLib::GetParameter
```

Definition at line 235 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 290 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.29 GetProcessChunkFrames

```
helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) > sound4::impact::dyn::CDynLib::GetProcessChunkFrames
```

Definition at line 291 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.30 GetSampleRate

```
helper::DynFuncHolder< decltype(impact_GetSampleRate) > sound4::impact::dyn::CDynLib::GetSampleRate
```

Definition at line 229 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 224 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 237 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 238 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 292 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 260 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 278 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 263 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.38 NewParameters

```
helper::DynFuncHolder< decltype(impact_NewParameters) > sound4::impact::dyn::CDynLib::NewParameters
```

Definition at line 232 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 287 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.40 ProcessAudio

```
helper::DynFuncHolder< decltype(impact_ProcessAudio) > sound4::impact::dyn::CDynLib::ProcessAudio
```

Definition at line 244 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 245 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.42 ProcessJson

```
helper::DynFuncHolder< decltype(impact_ProcessJson) > sound4::impact::dyn::CDynLib::ProcessJson
```

Definition at line 265 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 267 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 230 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 231 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 281 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 302 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 234 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 286 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 241 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 259 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 277 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 242 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 239 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 243 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 271 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.57 Webserver_FreeString

```
helper::DynFuncHolder< decltype(impact_Webserver_FreeString ) > sound4::impact::dyn::CDynLib↔  
::Webserver_FreeString
```

Definition at line 308 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.58 Webserver_GetAppHealth

```
helper::DynFuncHolder< decltype(impact_Webserver_GetAppHealth ) > sound4::impact::dyn::CDyn↔  
Lib::Webserver_GetAppHealth
```

Definition at line 307 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.59 Webserver_SetAppHealth

```
helper::DynFuncHolder< decltype(impact_Webserver_SetAppHealth ) > sound4::impact::dyn::CDynLib::Webserver_SetAppHealth
```

Definition at line 306 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.60 Webserver_Status

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

Definition at line 273 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.61 Webserver_Stop

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

Definition at line 272 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.62 Webserver_tcp

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

Definition at line 269 of file [sound4.impact.cl_dyn.hpp](#).

10.5.4.63 Webserver_tcp2

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

Definition at line 270 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 \(const std::filesystem::path &dynlib, uint32_t loadflags=LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_SYSTEM32\)](#)
- [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 136 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 233 of file [sound4cl.hpp](#).

10.6.3 Member Function Documentation

10.6.3.1 Close()

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

Definition at line 194 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 207 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 215 of file [sound4cl.hpp](#).

10.6.3.4 GetThisLibraryPath()

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

Definition at line 219 of file [sound4cl.hpp](#).

10.6.3.5 IsOk()

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

Definition at line 148 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_SYSTEM32|LOAD_LIBRARY_SEARCH_PATHS   
 ) [inline]
```

Definition at line 188 of file [sound4cl.hpp](#).

10.6.3.7 operator bool()

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

Definition at line 145 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 \(const std::filesystem::path &dynlib, uint32_t loadflags=LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_DEFAULT_PATH\)](#)
- [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 107 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 204 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 165 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 178 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 186 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 190 of file [sound4.impact.cl_dyn.hpp](#).

10.7.3.5 IsOk()

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

Definition at line 119 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_LIE
) [inline]
```

Definition at line 159 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 116 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
- virtual [~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](#) ()
- void [SetWebServerAppHealth](#) (int httpcode, const std::string &contenttype, const std::string &content)
- void [GetWebServerAppHealth](#) (int &httpcode, std::string &contenttype, std::string &content)
- 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 760 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 765 of file [sound4cl.hpp](#).

10.8.2.2 CInstance() [2/2]

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

10.8.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 777 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 1283 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

<i>LoginKey</i>	SOUND4 Login key
<i>RadioName</i>	SOUND4 RadioName
<i>Access_Key_ID</i>	SOUND4 AWS access ID
<i>Access_Key_Secret</i>	SOUND4 AWS access Secret
<i>save_path</i>	Path where we can load/save the configuration
<i>json_port</i>	Optional json listening port
<i>frames_per_chunk</i>	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 864 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 1289 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 1138 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 1154 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 910 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 1122 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 999 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 803 of file [sound4cl.hpp](#).

10.8.3.10 GetWebServerAppHealth()

```
void sound4::CInstance::GetWebServerAppHealth (
    int & httpcode,
    std::string & contenttype,
    std::string & content ) [inline]
```

Get the web server /apphealth status info

Definition at line 1076 of file [sound4cl.hpp](#).

10.8.3.11 IsOk()

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

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

Definition at line 785 of file [sound4cl.hpp](#).

10.8.3.12 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 1275 of file [sound4cl.hpp](#).

10.8.3.13 OnUpdateThreadStart()

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

Definition at line 1021 of file [sound4cl.hpp](#).

10.8.3.14 OnUpdateThreadStop()

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

Definition at line 1023 of file [sound4cl.hpp](#).

10.8.3.15 operator=()

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

10.8.3.16 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 conjunction with [SetPresetManager](#)

Parameters

<i>relative_path</i>	The path which changed, relative to the root storage
<i>change_kind</i>	What happened to this path

Definition at line 924 of file [sound4cl.hpp](#).

10.8.3.17 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

<i>T</i>	the C type for one audio sample
----------	---------------------------------

Parameters

<i>input</i>	audio to process
--------------	------------------

Returns

processed audio

Definition at line 1201 of file [sound4cl.hpp](#).

10.8.3.18 ProcessAudio()

```
void sound4::CInstance::ProcessAudio ( ) [inline]
```

Process one audio chunk, from [GetBufferIn\(\)](#) to [GetBufferOut\(\)](#)

See also

[sound4cl_ProcessAudio](#)

Definition at line 1164 of file [sound4cl.hpp](#).

10.8.3.19 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

<i>input</i>	has GetAudioInputCount() * GetChannelCount() channels
<i>output</i>	has GetAudioOutputCount() * GetChannelCount() channels

Definition at line 1183 of file [sound4cl.hpp](#).

10.8.3.20 SetBus()

```
void sound4::CInstance::SetBus (
    const CBus & bus ) [inline]
```

Set the instance bus

Definition at line 815 of file [sound4cl.hpp](#).

10.8.3.21 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

<i>key</i>	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.
<i>value</i>	the metadata value or nullptr to erase it.

Definition at line 942 of file [sound4cl.hpp](#).

10.8.3.22 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

<i>list</i>	a map of key/value pair.
-------------	--------------------------

Definition at line 963 of file [sound4cl.hpp](#).

10.8.3.23 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 793 of file [sound4cl.hpp](#).

10.8.3.24 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

<i>preset_manager</i>	The custom preset manager
-----------------------	---------------------------

Definition at line 831 of file [sound4cl.hpp](#).

10.8.3.25 SetWebServerAppHealth()

```
void sound4::CInstance::SetWebServerAppHealth (
    int httpcode,
    const std::string & contenttype,
    const std::string & content ) [inline]
```

Set the web server /apphealth status info

Definition at line 1067 of file [sound4cl.hpp](#).

10.8.3.26 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 1030 of file [sound4cl.hpp](#).

10.8.3.27 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 1105 of file [sound4cl.hpp](#).

10.8.3.28 StopWebServer()

```
void sound4::CInstance::StopWebServer ( ) [inline]
```

Stops the Web server

See also

[sound4cl_Webserver_Stop](#)

Definition at line 1055 of file [sound4cl.hpp](#).

10.8.3.29 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 1094 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
- virtual [~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](#) ()
- void [SetWebServerAppHealth](#) (int httpcode, const std::string &contenttype, const std::string &content)
- void [GetWebServerAppHealth](#) (int &httpcode, std::string &contenttype, std::string &content)
- 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()

```
virtual sound4::impact::CInstance::~~CInstance ( ) [inline], [virtual]
```

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 883 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

<i>LoginKey</i>	SOUND4 Login key
<i>RadioName</i>	SOUND4 RadioName
<i>AWS_Access_ID</i>	SOUND4 AWS access ID
<i>AWS_Access_Secret</i>	SOUND4 AWS access Secret
<i>save_path</i>	Path where we can load/save the configuration
<i>json_port</i>	Optional json listening port
<i>frames_per_chunk</i>	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 889 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 738 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 755 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 722 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 GetWebServerAppHealth()

```
void sound4::impact::CInstance::GetWebServerAppHealth (
    int & httpcode,
    std::string & contenttype,
    std::string & content ) [inline]
```

Get the web server /apphealth status info

Definition at line 673 of file [sound4.impact.cl.hpp](#).

10.9.3.11 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](#).

10.9.3.12 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 875 of file [sound4.impact.cl.hpp](#).

10.9.3.13 OnUpdateThreadStart()

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

Definition at line 622 of file [sound4.impact.cl.hpp](#).

10.9.3.14 OnUpdateThreadStop()

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

Definition at line 624 of file [sound4.impact.cl.hpp](#).

10.9.3.15 operator=()

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

10.9.3.16 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 conjunction with [SetPresetManager](#)

Parameters

<i>relative_path</i>	The path which changed, relative to the root storage
<i>change_kind</i>	What happened to this path

Definition at line 538 of file [sound4.impact.cl.hpp](#).

10.9.3.17 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

<i>T</i>	the C type for one audio sample
----------	---------------------------------

Parameters

<i>input</i>	audio to process
--------------	------------------

Returns

processed audio

Definition at line 803 of file [sound4.impact.cl.hpp](#).

10.9.3.18 ProcessAudio()

```
void sound4::impact::CInstance::ProcessAudio ( ) [inline]
```

Process one audio chunk, from [GetBufferIn\(\)](#) to [GetBufferOut\(\)](#)

See also

[impact_ProcessAudio](#)

Definition at line 766 of file [sound4.impact.cl.hpp](#).

10.9.3.19 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

<i>input</i>	has GetAudioInputCount() * GetChannelCount() channels
<i>output</i>	has GetAudioOutputCount() * GetChannelCount() channels

Definition at line 785 of file [sound4.impact.cl.hpp](#).

10.9.3.20 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

<i>key</i>	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.
<i>value</i>	the metadata value or nullptr to erase it.

Definition at line 554 of file [sound4.impact.cl.hpp](#).

10.9.3.21 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

<i>list</i>	a map of key/value pair.
-------------	--------------------------

Definition at line 572 of file [sound4.impact.cl.hpp](#).

10.9.3.22 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](#).

10.9.3.23 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

<i>preset_manager</i>	The custom preset manager
-----------------------	---------------------------

Definition at line 445 of file [sound4.impact.cl.hpp](#).

10.9.3.24 SetWebServerAppHealth()

```
void sound4::impact::CInstance::SetWebServerAppHealth (
    int httpcode,
```

```
const std::string & contenttype,  
const std::string & content ) [inline]
```

Set the web server /apphealth status info

Definition at line 665 of file [sound4.impact.cl.hpp](#).

10.9.3.25 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](#).

10.9.3.26 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 703 of file [sound4.impact.cl.hpp](#).

10.9.3.27 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](#).

10.9.3.28 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 692 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
- virtual [~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](#) ()
- void [SetWebServerAppHealth](#) (int httpcode, const std::string &contenttype, const std::string &content)
- void [GetWebServerAppHealth](#) (int &httpcode, std::string &contenttype, std::string &content)
- 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 816 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 821 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()

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

Destructor, stopping any running tasks

Definition at line 833 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 1340 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

<i>LoginKey</i>	SOUND4 Login key
<i>RadioName</i>	SOUND4 RadioName
<i>Access_Key_ID</i>	SOUND4 AWS access ID
<i>Access_Key_Secret</i>	SOUND4 AWS access Secret
<i>save_path</i>	Path where we can load/save the configuration
<i>json_port</i>	Optional json listening port
<i>frames_per_chunk</i>	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 921 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 1346 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 `GetAudiInputCount()` * `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 1193 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 1210 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 964 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 1177 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 1053 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 859 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.10 GetWebServerAppHealth()

```
void sound4::impact::dyn::CInstance::GetWebServerAppHealth (
    int & httpcode,
    std::string & contenttype,
    std::string & content ) [inline]
```

Get the web server /apphealth status info

Definition at line 1127 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.11 IsOk()

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

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

Definition at line 841 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.12 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 1332 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.13 OnUpdateThreadStart()

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

Definition at line 1075 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.14 OnUpdateThreadStop()

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

Definition at line 1077 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.15 operator=()

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

10.10.3.16 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 conjunction with [SetPresetManager](#)

Parameters

<i>relative_path</i>	The path which changed, relative to the root storage
<i>change_kind</i>	What happened to this path

Definition at line 978 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.17 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

<i>T</i>	the C type for one audio sample
----------	---------------------------------

Parameters

<i>input</i>	audio to process
--------------	------------------

Returns

processed audio

Definition at line 1258 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.18 ProcessAudio()

```
void sound4::impact::dyn::CInstance::ProcessAudio ( ) [inline]
```

Process one audio chunk, from [GetBufferIn\(\)](#) to [GetBufferOut\(\)](#)

See also

[impact_ProcessAudio](#)

Definition at line 1221 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.19 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

<i>input</i>	has GetAudioInputCount() * GetChannelCount() channels
<i>output</i>	has GetAudioOutputCount() * GetChannelCount() channels

Definition at line 1240 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.20 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

<i>key</i>	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.
<i>value</i>	the metadata value or nullptr to erase it.

Definition at line 996 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.21 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

<i>list</i>	a map of key/value pair.
-------------	--------------------------

Definition at line 1017 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.22 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 849 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.23 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

<i>preset_manager</i>	The custom preset manager
-----------------------	---------------------------

Definition at line 888 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.24 SetWebServerAppHealth()

```
void sound4::impact::dyn::CInstance::SetWebServerAppHealth (
    int httpcode,
    const std::string & contenttype,
    const std::string & content ) [inline]
```

Set the web server /apphealth status info

Definition at line 1118 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.25 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 1085 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.26 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 1158 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.27 StopWebServer()

```
void sound4::impact::dyn::CInstance::StopWebServer ( ) [inline]
```

Stops the Web server

See also

[impact_Webserver_Stop](#)

Definition at line 1107 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.28 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 1147 of file [sound4.impact.cl_dyn.hpp](#).

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

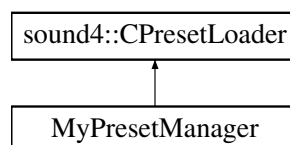
- [sound4.impact.cl_dyn.hpp](#)

10.11 sound4::CPresetLoader Class Reference

Custom preset handler helper.

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

Inheritance diagram for sound4::CPresetLoader:



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 740 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]
```

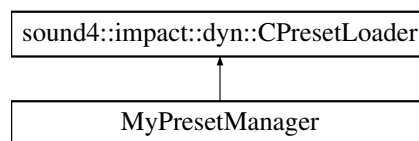
The documentation for this class was generated from the following file:

- [sound4.impact.cl.hpp](#)

10.13 sound4::impact::dyn::CPresetLoader Class Reference

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

Inheritance diagram for sound4::impact::dyn::CPresetLoader:



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 798 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](#)
- [helper::DynFuncHolder](#)< [decltype\(sound4cl_Webserver_SetAppHealth\)](#) > [fnWebserver_SetAppHealth](#)
- [helper::DynFuncHolder](#)< [decltype\(sound4cl_Webserver_GetAppHealth\)](#) > [fnWebserver_GetAppHealth](#)
- [helper::DynFuncHolder](#)< [decltype\(sound4cl_Webserver_FreeString\)](#) > [fnWebserver_FreeString](#)

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 [276](#) 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 [573](#) 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 575 of file [sound4cl.hpp](#).

10.14.3.3 AudioMonoFromLiveStereo()

```
void sound4::CProcessor::AudioMonoFromLiveStereo (
    const uint8_t * payload,
    float * output ) [inline]
```

Definition at line 585 of file [sound4cl.hpp](#).

10.14.3.4 AudioMonoToLiveStereo()

```
void sound4::CProcessor::AudioMonoToLiveStereo (
    const float * input,
    uint8_t * payload ) [inline]
```

Definition at line 587 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 645 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 655 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

<i>fmt</i>	the format
------------	------------

Returns

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

Definition at line 692 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 635 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 606 of file [sound4cl.hpp](#).

10.14.3.10 GetFormatFromName()

```
sound4cl_SampleFormat sound4::CProcessor::GetFormatFromName (
    const std::string & name ) [inline]
```

Convert string to format

Parameters

<i>name</i>	the format string
-------------	-------------------

Returns

the matching format, or INVALID_FORMAT if invalid.

Definition at line 683 of file [sound4cl.hpp](#).

10.14.3.11 GetFormatName()

```
std::string sound4::CProcessor::GetFormatName (
    const sound4cl_SampleFormat fmt ) [inline]
```

Convert format to string

Parameters

<i>fmt</i>	the format
------------	------------

Returns

the string name of the format

Definition at line 674 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 617 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 370 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 665 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 595 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 565 of file [sound4cl.hpp](#).

10.14.3.17 Load()

```
bool sound4::CProcessor::Load (   
    const std::filesystem::path & filepath ) [inline]
```

Parameters

<i>filepath</i>	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 385 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 579 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 583 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 704 of file [sound4cl.hpp](#).

10.14.3.21 SetLogSeverity()

```
void sound4::CProcessor::SetLogSeverity (
    LogSeverity severity ) [inline]
```

Sets the log severity (global)

Definition at line 698 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 577 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 581 of file [sound4cl.hpp](#).

10.14.4 Friends And Related Symbol Documentation

10.14.4.1 CBus

```
friend class CBus [friend]
```

Definition at line 282 of file [sound4cl.hpp](#).

10.14.4.2 CInstance

```
friend class CInstance [friend]
```

Definition at line 283 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 315 of file [sound4cl.hpp](#).

10.14.5.2 fnAddPadAudio

```
helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > sound4::CProcessor::fnAddPadAudio
[protected]
```

Definition at line 316 of file [sound4cl.hpp](#).

10.14.5.3 fnAudioConvertFrom

```
helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > sound4::CProcessor::fnAudio↔  
ConvertFrom [protected]
```

Definition at line 319 of file [sound4cl.hpp](#).

10.14.5.4 fnAudioConvertTo

```
helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > sound4::CProcessor::fnAudio↔  
ConvertTo [protected]
```

Definition at line 320 of file [sound4cl.hpp](#).

10.14.5.5 fnAudioMonoFromLiveStereo

```
helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > sound4::CProcessor::fn↔  
AudioMonoFromLiveStereo [protected]
```

Definition at line 323 of file [sound4cl.hpp](#).

10.14.5.6 fnAudioMonoToLiveStereo

```
helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > sound4::CProcessor::fn↔  
AudioMonoToLiveStereo [protected]
```

Definition at line 324 of file [sound4cl.hpp](#).

10.14.5.7 fnDeleteClient

```
helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > sound4::CProcessor::fnDeleteClient  
[protected]
```

Definition at line 326 of file [sound4cl.hpp](#).

10.14.5.8 fnExitProcess

```
helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > sound4::CProcessor::fnExitProcess  
[protected]
```

Definition at line 302 of file [sound4cl.hpp](#).

10.14.5.9 fnFreeBus

```
helper::DynFuncHolder< decltype(sound4cl_FreeBus) > sound4::CProcessor::fnFreeBus [protected]
```

Definition at line 358 of file [sound4cl.hpp](#).

10.14.5.10 fnFreeJsonAnswer

```
helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer ) > sound4::CProcessor::fnFreeJson↔  
Answer [protected]
```

Definition at line 328 of file [sound4cl.hpp](#).

10.14.5.11 fnFreeMetadataInfos

```
helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos ) > sound4::CProcessor::fnFree↔  
MetadataInfos [protected]
```

Definition at line 345 of file [sound4cl.hpp](#).

10.14.5.12 fnFreeParameters

```
helper::DynFuncHolder< decltype(sound4cl_FreeParameters ) > sound4::CProcessor::fnFree↔  
Parameters [protected]
```

Definition at line 295 of file [sound4cl.hpp](#).

10.14.5.13 fnFreeParameterValue

```
helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue ) > sound4::CProcessor::fnFree↔  
ParameterValue [protected]
```

Definition at line 298 of file [sound4cl.hpp](#).

10.14.5.14 fnGetAudio

```
helper::DynFuncHolder< decltype(sound4cl_GetAudio ) > sound4::CProcessor::fnGetAudio [protected]
```

Definition at line 318 of file [sound4cl.hpp](#).

10.14.5.15 fnGetAudioInputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount ) > sound4::CProcessor::fnGet↔  
AudioInputCount [protected]
```

Definition at line 289 of file [sound4cl.hpp](#).

10.14.5.16 fnGetAudioOutputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount ) > sound4::CProcessor::fnGet↔  
AudioOutputCount [protected]
```

Definition at line 290 of file [sound4cl.hpp](#).

10.14.5.17 fnGetBufferIn

```
helper::DynFuncHolder< decltype(sound4cl_GetBufferIn ) > sound4::CProcessor::fnGetBufferIn  
[protected]
```

Definition at line 308 of file [sound4cl.hpp](#).

10.14.5.18 fnGetBufferOut

```
helper::DynFuncHolder< decltype(sound4cl_GetBufferOut ) > sound4::CProcessor::fnGetBufferOut  
[protected]
```

Definition at line 309 of file [sound4cl.hpp](#).

10.14.5.19 fnGetBytesFromFormat

```
helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat ) > sound4::CProcessor::fnGet↵  
BytesFromFormat [protected]
```

Definition at line 313 of file [sound4cl.hpp](#).

10.14.5.20 fnGetChannelCount

```
helper::DynFuncHolder< decltype(sound4cl_GetChannelCount ) > sound4::CProcessor::fnGetChannel↵  
Count [protected]
```

Definition at line 288 of file [sound4cl.hpp](#).

10.14.5.21 fnGetChunkSizeInFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames ) > sound4::CProcessor::fnGet↵  
ChunkSizeInFrames [protected]
```

Definition at line 287 of file [sound4cl.hpp](#).

10.14.5.22 fnGetEstimatedDelay

```
helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay ) > sound4::CProcessor::fnGet↵  
EstimatedDelay [protected]
```

Definition at line 310 of file [sound4cl.hpp](#).

10.14.5.23 fnGetFormatFromName

```
helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName ) > sound4::CProcessor::fnGet↵  
FormatFromName [protected]
```

Definition at line 312 of file [sound4cl.hpp](#).

10.14.5.24 fnGetFormatName

```
helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > sound4::CProcessor::fnGetFormatName  
[protected]
```

Definition at line 311 of file [sound4cl.hpp](#).

10.14.5.25 fnGetMaxPacketFrame

```
helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > sound4::CProcessor::fnGetMax↵  
PacketFrame [protected]
```

Definition at line 314 of file [sound4cl.hpp](#).

10.14.5.26 fnGetMetadataInfos

```
helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > sound4::CProcessor::fnGet↵  
MetadataInfos [protected]
```

Definition at line 344 of file [sound4cl.hpp](#).

10.14.5.27 fnGetOutputCount

```
helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > sound4::CProcessor::fnGetOutput↵  
Count [protected]
```

Definition at line 317 of file [sound4cl.hpp](#).

10.14.5.28 fnGetParameter

```
helper::DynFuncHolder< decltype(sound4cl_GetParameter) > sound4::CProcessor::fnGetParameter  
[protected]
```

Definition at line 297 of file [sound4cl.hpp](#).

10.14.5.29 fnGetPossibleChunkSizeInFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > sound4::CProcessor↵  
::fnGetPossibleChunkSizeInFrames [protected]
```

Definition at line 352 of file [sound4cl.hpp](#).

10.14.5.30 fnGetProcessChunkFrames

```
helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > sound4::CProcessor::fnGet↵  
ProcessChunkFrames [protected]
```

Definition at line 353 of file [sound4cl.hpp](#).

10.14.5.31 fnGetSampleRate

```
helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > sound4::CProcessor::fnGetSampleRate  
[protected]
```

Definition at line 291 of file [sound4cl.hpp](#).

10.14.5.32 fnGetVersion

```
helper::DynFuncHolder< decltype(sound4cl_GetVersion) > sound4::CProcessor::fnGetVersion  
[protected]
```

Definition at line 286 of file [sound4cl.hpp](#).

10.14.5.33 fnInitProcess

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess) > sound4::CProcessor::fnInitProcess  
[protected]
```

Definition at line 299 of file [sound4cl.hpp](#).

10.14.5.34 fnInitProcess2

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > sound4::CProcessor::fnInitProcess2  
[protected]
```

Definition at line 300 of file [sound4cl.hpp](#).

10.14.5.35 fnInitProcess3

```
helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > sound4::CProcessor::fnInitProcess3  
[protected]
```

Definition at line 354 of file [sound4cl.hpp](#).

10.14.5.36 fnMonoToStereo

```
helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > sound4::CProcessor::fnMonoToStereo  
[protected]
```

Definition at line 322 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 340 of file [sound4cl.hpp](#).

10.14.5.38 fnNewBus

```
helper::DynFuncHolder< decltype(sound4cl_NewBus ) > sound4::CProcessor::fnNewBus [protected]
```

Definition at line 357 of file [sound4cl.hpp](#).

10.14.5.39 fnNewClient

```
helper::DynFuncHolder< decltype(sound4cl_NewClient ) > sound4::CProcessor::fnNewClient [protected]
```

Definition at line 325 of file [sound4cl.hpp](#).

10.14.5.40 fnNewParameters

```
helper::DynFuncHolder< decltype(sound4cl_NewParameters ) > sound4::CProcessor::fnNewParameters  
[protected]
```

Definition at line 294 of file [sound4cl.hpp](#).

10.14.5.41 fnPresetManager_InformChange

```
helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange ) > sound4::CProcessor↵  
::fnPresetManager_InformChange [protected]
```

Definition at line 349 of file [sound4cl.hpp](#).

10.14.5.42 fnProcessAudio

```
helper::DynFuncHolder< decltype(sound4cl_ProcessAudio ) > sound4::CProcessor::fnProcessAudio  
[protected]
```

Definition at line 306 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 307 of file [sound4cl.hpp](#).

10.14.5.44 fnProcessJson

```
helper::DynFuncHolder< decltype(sound4cl_ProcessJson ) > sound4::CProcessor::fnProcessJson  
[protected]
```

Definition at line 327 of file [sound4cl.hpp](#).

10.14.5.45 fnSaveState

```
helper::DynFuncHolder< decltype(sound4cl_SaveState ) > sound4::CProcessor::fnSaveState [protected]
```

Definition at line 329 of file [sound4cl.hpp](#).

10.14.5.46 fnSetInstanceBus

```
helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus ) > sound4::CProcessor::fnSetInstance↔  
Bus [protected]
```

Definition at line 359 of file [sound4cl.hpp](#).

10.14.5.47 fnSetLoggerCallback

```
helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback ) > sound4::CProcessor::fnSet↔  
LoggerCallback [protected]
```

Definition at line 292 of file [sound4cl.hpp](#).

10.14.5.48 fnSetLogSeverity

```
helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity ) > sound4::CProcessor::fnSetLog↔  
Severity [protected]
```

Definition at line 293 of file [sound4cl.hpp](#).

10.14.5.49 fnSetMetadata

```
helper::DynFuncHolder< decltype(sound4cl_SetMetadata ) > sound4::CProcessor::fnSetMetadata  
[protected]
```

Definition at line 343 of file [sound4cl.hpp](#).

10.14.5.50 fnSetMetadataMulti

```
helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti ) > sound4::CProcessor::fnSet↔  
MetadataMulti [protected]
```

Definition at line 362 of file [sound4cl.hpp](#).

10.14.5.51 fnSetParameter

```
helper::DynFuncHolder< decltype(sound4cl_SetParameter ) > sound4::CProcessor::fnSetParameter  
[protected]
```

Definition at line 296 of file [sound4cl.hpp](#).

10.14.5.52 fnSetPresetManager

```
helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > sound4::CProcessor::fnSet↵  
PresetManager [protected]
```

Definition at line 348 of file [sound4cl.hpp](#).

10.14.5.53 fnStartUpdateThread

```
helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > sound4::CProcessor::fnStart↵  
UpdateThread [protected]
```

Definition at line 303 of file [sound4cl.hpp](#).

10.14.5.54 fnStereoToMono

```
helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > sound4::CProcessor::fnStereoToMono  
[protected]
```

Definition at line 321 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 339 of file [sound4cl.hpp](#).

10.14.5.56 fnStopUpdateThread

```
helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > sound4::CProcessor::fnStop↵  
UpdateThread [protected]
```

Definition at line 304 of file [sound4cl.hpp](#).

10.14.5.57 fnTerminateProcess

```
helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > sound4::CProcessor::fnTerminate↵  
Process [protected]
```

Definition at line 301 of file [sound4cl.hpp](#).

10.14.5.58 fnWaitUpdateThreadReady

```
helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > sound4::CProcessor::fn↵  
WaitUpdateThreadReady [protected]
```

Definition at line 305 of file [sound4cl.hpp](#).

10.14.5.59 fnWebserver

```
helper::DynFuncHolder< decltype(sound4cl_Webserver ) > sound4::CProcessor::fnWebserver [protected]
```

Definition at line 334 of file [sound4cl.hpp](#).

10.14.5.60 fnWebserver_FreeString

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString ) > sound4::CProcessor::fn↵  
Webserver_FreeString [protected]
```

Definition at line 367 of file [sound4cl.hpp](#).

10.14.5.61 fnWebserver_GetAppHealth

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth ) > sound4::CProcessor::fn↵  
Webserver_GetAppHealth [protected]
```

Definition at line 366 of file [sound4cl.hpp](#).

10.14.5.62 fnWebserver_SetAppHealth

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth ) > sound4::CProcessor::fn↵  
Webserver_SetAppHealth [protected]
```

Definition at line 365 of file [sound4cl.hpp](#).

10.14.5.63 fnWebserver_Status

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_Status ) > sound4::CProcessor::fnWebserver↵  
_Status [protected]
```

Definition at line 336 of file [sound4cl.hpp](#).

10.14.5.64 fnWebserver_Stop

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop ) > sound4::CProcessor::fnWebserver_↵  
Stop [protected]
```

Definition at line 335 of file [sound4cl.hpp](#).

10.14.5.65 fnWebserver_tcp

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp ) > sound4::CProcessor::fnWebserver_tcp  
[protected]
```

Definition at line 332 of file [sound4cl.hpp](#).

10.14.5.66 fnWebserver_tcp2

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2 ) > sound4::CProcessor::fnWebserver_tcp2 [protected]
```

Definition at line 333 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 121 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 126 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 128 of file [sound4cl.hpp](#).

10.15.3.2 operator bool()

```
template<typename T >
sound4::helper::DynFuncHolder< T >::operator bool ( ) const [inline]
```

Definition at line 127 of file [sound4cl.hpp](#).

10.15.3.3 operator T*()

```
template<typename T >
sound4::helper::DynFuncHolder< T >::operator T* ( ) const [inline]
```

Definition at line 129 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 92 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 97 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 99 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 98 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 100 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

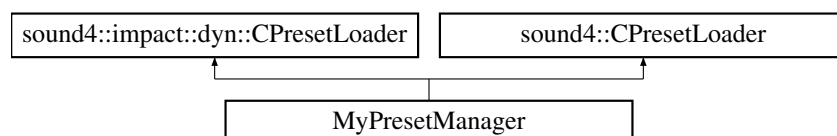
The documentation for this struct was generated from the following file:

- [sound4.impact.cl.h](#)

10.20 `MyPresetManager` Class Reference

[Custom preset manager]

Inheritance diagram for `MyPresetManager`:



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

Public Member Functions inherited from [sound4::impact::dyn::CPresetLoader](#)

- [CPresetLoader](#) ()=default
- virtual [~CPresetLoader](#) ()=default

Public Member Functions inherited from [sound4::CPresetLoader](#)

- [CPresetLoader](#) ()=default
- virtual [~CPresetLoader](#) ()=default

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 167 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 151 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 177 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 161 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

<i>T</i>	the C type for one audio sample
----------	---------------------------------

Definition at line 91 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

<i>T</i>	the C type for one audio sample
----------	---------------------------------

Definition at line 535 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

<i>T</i>	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 102 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 103 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 546 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 547 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 94 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 95 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 538 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 539 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 98 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 99 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 542 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 543 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 203 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 205 of file [sound4.impact.cl.h](#).

10.33.2.2 prefix

```
const char* SOUND4_ProcessInfo::prefix
```

Definition at line 206 of file [sound4.impact.cl.h](#).

10.33.2.3 shortname

```
const char* SOUND4_ProcessInfo::shortname
```

Definition at line 207 of file [sound4.impact.cl.h](#).

10.33.2.4 struct_size

```
size_t SOUND4_ProcessInfo::struct_size
```

Definition at line 204 of file [sound4.impact.cl.h](#).

10.33.2.5 version

```
const char* SOUND4_ProcessInfo::version
```

Definition at line 208 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

[Go to the documentation of this file.](#)

```

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
00058     while (!stopthread)
00059     {
00060         // Read a chunk from stdin
00061         if (stdin::fread(instance.GetBufferIn().data(), sizeof(float), sound4::impact::InputSampleSize,
00062             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,
00072             stdout) != sound4::impact::OutputSampleSize) {
00073             std::cerr << "Failed to write audio output" << std::endl;
00074             stopthread=true;
00075         }
00076     }
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 }
00103
00104 int main(int argc, const char **argv)
00105 {
00106     #if (IMPACT_HAS_WEBSERVER)
00107         int listen_port = 8080;
00108     #endif // (IMPACT_HAS_WEBSERVER)
00109     auto store_path_str=GetEnv("S4STORE_PATH");
00110
00111     // Get credentials from environment
00112     auto LoginKey=GetEnv("S4LOGINKEY");
00113     if (LoginKey.empty()) {
00114         LoginKey=GetEnv("S4USERID");
00115     }
00116     auto RadioName=GetEnv("RADIO_NAME");
00117     auto KeyId=GetEnv("ACCESS_KEY_ID");
00118     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00119
00120     const char *programe=argv[0];
00121
00122     if (argc>1) {
00123         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00124             Usage(programe);
00125             return 0;
00126         }
00127     }
00128     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00129         std::cerr << "Missing credentials in environment" << std::endl;
00130         Usage(programe);
00131         return 1;
00132     }
00133     std::filesystem::path store_path;
00134     if (!store_path_str.empty()) {
00135         std::error_code ec;
00136         store_path = std::filesystem::path(store_path_str);
00137         if (!std::filesystem::exists(store_path,ec) || !std::filesystem::is_directory(store_path,ec))
00138         {
00139             std::cerr << "Storage folder '"<store_path.c_str()<"' is not a valid directory" <<
00140             std::endl;
00141             Usage(programe);
00142             return 1;
00143         }
00144     }
00145     sound4::impact::SetLogSeverity(verbose5);
00146     sound4::impact::SetLoggerCallback(MyLogger);
00147
00148     sound4::impact::CInstance instance;
00149     instance.SetParam("ADMIN_USER","admin");
00150     instance.SetParam("ADMIN_SECRET","admin");
00151
00152     std::cerr << "Creating" << std::endl;
00153     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
00154         std::cerr << "Failed to create the processing instance" << std::endl;
00155         return 1;
00156     }
00157
00158     #if (IMPACT_HAS_WEBSERVER)
00159     std::cerr << "Starting web server" << std::endl;
00160     if (!instance.StartWebServer(listen_port,0)) {
00161         std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00162         return 1;
00163     }
00164     #endif // (IMPACT_HAS_WEBSERVER)
00165
00166     {
00167         auto client = instance.NewClient();
00168         auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
00169         std::cerr << "Request for processor name returned "<answer<std::endl;
00170     }
00171
00172     ProcessLoop(instance);
00173
00174     std::cerr << "Stopping now" << std::endl;
00175     instance.Stop();
00176
00177     std::cerr << "End" << std::endl;
00178     return 0;
00179 }

```

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 *progrname, [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 198 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

[Go to the documentation of this file.](#)

```

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 *programe, sound4::impact::dyn::CDynLib &dynlib)
00092 {
00093     std::cerr << "Usage: " << programe << 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;
00103         std::cerr << "Example:" << std::endl;
00104         std::cerr << "ffmpeg -re -i file.wav -ar "<<dynlib.GetSampleRate()<<" -ac
"<<dynlib.GetChannelCount()<<" -f f32le - | "<<programe<<" | ffplay -f f32le -ar
"<<dynlib.GetSampleRate()<<" -ac "<<dynlib.GetChannelCount()<<" -i -" << std::endl;;
00105         #if defined(_WIN32) || defined(WIN32)
00106             std::cerr << "ffmpeg -f dshow -i audio=\"Microphone\" -ar "<<dynlib.GetSampleRate()<<" -ac
"<<dynlib.GetChannelCount()<<" -f f32le - | "<<programe<<" | ffplay -f f32le -ar
"<<dynlib.GetSampleRate()<<" -ac "<<dynlib.GetChannelCount()<<" -i -" << std::endl;;
00107         #else
00108             std::cerr << "ffmpeg -f pulse -i default -ar "<<dynlib.GetSampleRate()<<" -ac
"<<dynlib.GetChannelCount()<<" -f f32le - | "<<programe<<" | ffplay -f f32le -ar
"<<dynlib.GetSampleRate()<<" -ac "<<dynlib.GetChannelCount()<<" -i -" << std::endl;;
00109         #endif

```

```

00110     }
00111 }
00112
00113 void MyLogger(sound4::impact::dyn::LogSeverity severity, const std::string& msg)
00114 {
00115     std::cerr << msg << std::endl;
00116 }
00117
00118 class MyPresetManager : public sound4::impact::dyn::CPresetLoader {
00119 public:
00120     MyPresetManager() = default;
00121     virtual ~MyPresetManager() = default;
00122
00123     bool SetupPath(const std::filesystem::path& a_preset_path)
00124     {
00125         m_preset_path=a_preset_path;
00126         std::error_code ec;
00127         if ( !std::filesystem::exists(m_preset_path,ec) ) {
00128             return false;
00129         }
00130         if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00131             return false;
00132         }
00133         return true;
00134     }
00135
00136     virtual bool IsReadOnly() override { return false; };
00137     virtual bool Exists(const std::filesystem::path &name) override
00138     {
00139         std::error_code ec;
00140         return std::filesystem::exists(GetRealPath(name), ec);
00141     }
00142     virtual bool Remove(const std::filesystem::path &name) override
00143     {
00144         std::error_code ec;
00145         return std::filesystem::remove(GetRealPath(name), ec);
00146     }
00147     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
00148     {
00149         std::error_code ec;
00150         std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
00151         return (!ec);
00152     }
00153     virtual std::vector<std::filesystem::path> GetAll() override
00154     {
00155         std::vector<std::filesystem::path> list;
00156         std::error_code ec;
00157         for (auto it = std::filesystem::directory_iterator(m_preset_path.c_str(),
00158             std::filesystem::directory_options::skip_permission_denied, ec);
00159             !ec && it != std::filesystem::directory_iterator{};
00160             it.increment(ec) )
00161         {
00162             if (it->is_regular_file(ec)) {
00163                 list.push_back(it->path().lexically_relative(m_preset_path));
00164             }
00165         }
00166         return list;
00167     }
00168     virtual std::string Read(const std::filesystem::path &filename) override
00169     {
00170         std::ifstream f(GetRealPath(filename));
00171         if ( f.fail() ) {
00172             return std::string();
00173         }
00174         std::ostringstream ss;
00175         ss << f.rdbuf();
00176         return ss.str();
00177     }
00178     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00179     {
00180         std::ofstream f(GetRealPath(filename));
00181         if ( f.fail() ) {
00182             return false;
00183         }
00184         f << content;
00185         if ( f.fail() ) {
00186             return false;
00187         }
00188         return true;
00189     }
00190 private:
00191     std::filesystem::path m_preset_path;
00192     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00193     {
00194         return m_preset_path / name;
00195     }
00196 };
00197

```

```

00198 int main(int argc, const char **argv)
00199 {
00200     #if (IMPACT_HAS_WEBSERVER)
00201         int listen_port = 8080;
00202     #endif // (IMPACT_HAS_WEBSERVER)
00203     auto store_path_str=GetEnv("S4STORE_PATH");
00204
00205     // Get credentials from environment
00206     auto LoginKey=GetEnv("S4LOGINKEY");
00207     if (LoginKey.empty()) {
00208         LoginKey=GetEnv("S4USERID");
00209     }
00210     auto RadioName=GetEnv("RADIO_NAME");
00211     auto KeyId=GetEnv("ACCESS_KEY_ID");
00212     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00213
00214     const char *programe=argv[0];
00215
00216     sound4::impact::dyn::CDynLib dynlib;
00217     if (!dynlib.Load()) {
00218         std::cerr << "Failed to load processing library" << std::endl;
00219         return 1;
00220     }
00221
00222     if (argc>1) {
00223         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00224             Usage(programe, dynlib);
00225             return 0;
00226         }
00227     }
00228     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00229         std::cerr << "Missing credentials in environment" << std::endl;
00230         Usage(programe, dynlib);
00231         return 1;
00232     }
00233     std::filesystem::path store_path;
00234     if (!store_path_str.empty()) {
00235         std::error_code ec;
00236         store_path = std::filesystem::path(store_path_str);
00237         if (!std::filesystem::exists(store_path,ec) || !std::filesystem::is_directory(store_path,ec))
00238         {
00239             std::cerr << "Storage folder '"<store_path.c_str()<"' is not a valid directory" <<
00240             std::endl;
00241             Usage(programe, dynlib);
00242             return 1;
00243         }
00244     }
00245
00246     sound4::impact::dyn::SetLogSeverity(dynlib, verbose5);
00247     sound4::impact::dyn::SetLoggerCallback(dynlib, MyLogger);
00248
00249     sound4::impact::dyn::CInstance instance(dynlib);
00250     instance.SetParam("ADMIN_USER", "admin");
00251     instance.SetParam("ADMIN_SECRET", "admin");
00252
00253     MyPresetManager preset_manager;
00254     if (!store_path.empty()) {
00255         if (!preset_manager.SetupPath(store_path)) {
00256             std::cerr << "Failed to setup the custom preset manager on path " << store_path.c_str() <<
00257             std::endl;
00258             return 1;
00259         }
00260         instance.SetPresetManager(&preset_manager);
00261     }
00262
00263     std::cerr << "Creating" << std::endl;
00264     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
00265         std::cerr << "Failed to create the processing instance" << std::endl;
00266         return 1;
00267     }
00268
00269     #if (IMPACT_HAS_WEBSERVER)
00270     std::cerr << "Starting web server" << std::endl;
00271     if (!instance.StartWebServer(listen_port,0)) {
00272         std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00273         return 1;
00274     }
00275     #endif // (IMPACT_HAS_WEBSERVER)
00276
00277     {
00278         auto client = instance.NewClient();
00279         auto answer = client->ProcessJson(R"JSON({{get":{"processorname":null}}})JSON");
00280         std::cerr << "Request for processor name returned "<answer<std::endl;
00281     }
00282
00283     ProcessLoop(instance);

```

```
00294
00295     std::cerr << "Stopping now" << std::endl;
00296     instance.Stop();
00297
00298     std::cerr << "End" << std::endl;
00299     return 0;
00300 }
```

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

- namespace [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 *programe)
- void [MyLogger](#) ([impact_LogSeverity](#) severe, 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,
    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

[Go to the documentation of this file.](#)

```
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 {
00033     bool stopthread=false;
00034     alignas(64) float audio_in[CHUNK_SIZE*CHANNEL_COUNT];
00035     alignas(64) float audio_out[CHUNK_SIZE*CHANNEL_COUNT];
00036
00037     #if defined(_WIN32) || defined(WIN32)
00038         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00039         _setmode(_fileno(stdin), _O_BINARY);
00040         _setmode(_fileno(stdout), _O_BINARY);
00041     #endif
00042
00043     while (!stopthread)
00044     {
00045         // Read a chunk from stdin
00046         if (std::fread(&audio_in[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdin) !=
00047             CHUNK_SIZE*CHANNEL_COUNT) {
00048             std::cerr << "ProcessLoop: input stopped" << std::endl;
00049             stopthread=true;
00050             break;
00051         }
00052         if (stopthread) break;
00053         // Process it
00054         impact_ProcessAudio(pInstance, audio_in, audio_out);
00055         // Write it to stdout
00056         if (fwrite(&audio_out[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdout) !=
00057             CHUNK_SIZE*CHANNEL_COUNT) {
00058             std::cerr << "Failed to write audio output" << std::endl;
00059             stopthread=true;
00060         }
00061     }
00062 }
00063
00064 void Usage(const char *progname)
00065 {
00066     std::cerr << "Usage: " << progname << std::endl;;
00067     std::cerr << "  Reads audio from stdin and return processed audio in stdout" << std::endl;
00068     std::cerr << "Environment:" << std::endl;;
00069     std::cerr << "  - S4STORE_PATH          : path to store config [default: ./state]" << std::endl;;
00070     std::cerr << "  - S4LOGINKEY           : SOUND4 Login key (or S4USERID)" << std::endl;;
```

```

00071     std::cerr << " - RADIO_NAME           : SOUND4 Radio Name" << std::endl;;
00072     std::cerr << " - ACCESS_KEY_ID       : SOUND4 Access Key ID" << std::endl;;
00073     std::cerr << " - ACCESS_KEY_SECRET    : SOUND4 Access Key Secret" << std::endl;;
00074     std::cerr << std::endl;
00075     std::cerr << "Example:" << std::endl;
00076     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;;
00077     #if defined(_WIN32) || defined(WIN32)
00078     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;;
00079     #else
00080     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;;
00081     #endif
00082 }
00083
00084 void MyLogger(impact_LogSeverity severe [[maybe_unused]], const char *str)
00085 {
00086     std::cerr << str << std::endl;
00087 }
00088
00089
00090 int main(int argc, char *const argv[])
00091 {
00092     struct impact_CInstance *pInstance = NULL;
00093     const char *store_path_str=getenv("S4STORE_PATH");
00094     // Get credentials from environment
00095     char *pLoginKey=getenv("S4LOGINKEY");
00096     if (!pLoginKey) {
00097         pLoginKey=getenv("S4USERID");
00098     }
00099     char *pRadioName=getenv("RADIO_NAME");
00100     char *pKeyId=getenv("ACCESS_KEY_ID");
00101     char *pAccessKey=getenv("ACCESS_KEY_SECRET");
00102
00103     const char *progname=argv[0];
00104
00105     if (argc>1) {
00106         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00107             Usage(progname);
00108             return 0;
00109         }
00110     }
00111     if (!pLoginKey || !pRadioName || !pKeyId || !pAccessKey) {
00112         std::cerr << "Missing credentials in environment" << std::endl;
00113         Usage(progname);
00114         return 1;
00115     }
00116     std::error_code ec;
00117     std::filesystem::path fs_store_path;
00118     if (!store_path_str || store_path_str[0]==0) {
00119         fs_store_path=std::filesystem::current_path(ec) / "state";
00120     } else {
00121         fs_store_path = std::filesystem::path(store_path_str);
00122     }
00123     if (!std::filesystem::exists(fs_store_path,ec) ||
!std::filesystem::is_directory(fs_store_path,ec)) {
00124         std::cerr << "Storage folder '" << fs_store_path.c_str() << "' is not a valid directory" << std::endl;
00125         Usage(progname);
00126         return 1;
00127     }
00128     std::string store_path;
00129     #ifdef _WIN32
00130     {
00131         // Convert path to UTF-8
00132         std::wstring wstr(fs_store_path.wstring());
00133         if (wstr.empty()) return {};
00134         int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
00135         std::string str(size, 0);
00136         WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00137         store_path=str;
00138     }
00139     #else // !_WIN32
00140         store_path=fs_store_path;
00141     #endif // !_WIN32
00142
00143     // Optional: Use our own logging, instead of default color output to stderr.
00144     //impact_SetLoggerCallback(&MyLogger);
00145
00146     //impact_SetLogSeverity(impact_LogSeverity::info);
00147     impact_SetLogSeverity(impact_LogSeverity::verbose5);
00148
00149     std::cerr << "Process version: " << impact_GetVersion() << std::endl;
00150     // Do some sanity checks
00151     if (impact_GetSampleRate() != SAMPLE_RATE) {
00152         std::cerr << "Bad library sampling rate, expecting '" << SAMPLE_RATE << "' but has " <<

```

```

        impact_GetSampleRate() << std::endl;
00154         return 1;
00155     }
00156     if (impact_GetChunkSizeInFrames() != CHUNK_SIZE) {
00157         std::cerr << "Bad library frame size, expecting '" << CHUNK_SIZE << "' but has " <<
        impact_GetChunkSizeInFrames() << std::endl;
00158         return 1;
00159     }
00160     if (impact_GetChannelCount() != CHANNEL_COUNT) {
00161         std::cerr << "Bad library channel count, expecting '" << CHANNEL_COUNT << "' but has " <<
        impact_GetChannelCount() << std::endl;
00162         return 1;
00163     }
00164
00165     // Setup parameters
00166     struct impact_CParameters *params = impact_NewParameters();
00167     // Optional: activate web authentication, set admin password
00168     // This creates the authentication only for a new storage path
00169     impact_SetParameter(params, "ADMIN_USER", "admin");
00170     impact_SetParameter(params, "ADMIN_SECRET", "admin");
00171
00172     // Create the processing instance
00173     pInstance = impact_InitProcess2(pLoginKey, pRadioName, pKeyId, pAccessKey, store_path.c_str(), params);
00174     impact_FreeParameters(params);
00175     if (!pInstance) {
00176         std::cerr << "Failed to create the process instance" << std::endl;
00177         return 1;
00178     }
00179
00180     // Start the update thread, without JSON server (low priority)
00181     std::thread updater([pInstance]() {
00182         // will returned when stopped or destroyed
00183         std::cerr << "impact_StartUpdateThread (no json port)" << std::endl;
00184         impact_StartUpdateThread(pInstance, 0);
00185         std::cerr << "impact_StartUpdateThread END" << std::endl;
00186     });
00187
00188     // This is not mandatory, but better be sure we are ready right now
00189     std::cerr << "Waiting update thread" << std::endl;
00190     if (impact_WaitUpdateThreadReady(pInstance, 5000) < 0) {
00191         std::cerr << "Timeout to create the process instance" << std::endl;
00192     }
00193     std::cerr << "Waiting update thread DONE" << std::endl;
00194
00195     #if (IMPACT_HAS_WEBSERVER)
00196     // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
00197     std::cerr << "Starting web server" << std::endl;
00198     int listenport = 8080;
00199     int listenport_secure = 8443;
00200     uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
00201     if (listenport || listenport_secure) {
00202         webserver = impact_Webserver(listenport, listenport_secure, pInstance);
00203         if (webserver == SOUND4_INVALID_WEBSERVER_ID) {
00204             std::cerr << "Starting web server FAILED" << std::endl;
00205         } else {
00206             int web_status = impact_Webserver_Status(webserver);
00207             if (listenport) {
00208                 if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
00209                     std::cerr << "Web server listening on port " << listenport;
00210                 } else {
00211                     std::cerr << "Web server failed to listen on port " << listenport;
00212                 }
00213             }
00214             if (listenport_secure) {
00215                 if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
00216                     std::cerr << "Web server listening on secure port " << listenport_secure;
00217                 } else {
00218                     std::cerr << "Web server failed to listen on secure port " << listenport_secure;
00219                 }
00220             }
00221             std::cerr << "Starting web server DONE" << std::endl;
00222         }
00223     }
00224     #endif // (IMPACT_HAS_WEBSERVER)
00225
00226     // Force load a factory preset now
00227     // Could be used instead of setting a store_path
00228     /*{
00229         impact_CClientInstance *pClient = impact_NewClient(pInstance);
00230         const char *answer = impact_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }",
00231         NULL);
00232         std::cerr << "Loaded factory preset answer: " << answer << std::endl;
00233         impact_FreeJsonAnswer(answer);
00234         impact_DeleteClient(pClient);
00235     }*/
00236
00237     std::cerr << "Estimated delay " << ( impact_GetEstimatedDelay(pInstance) * 1000.0 / SAMPLE_RATE ) << "

```

```

        ms" « std::endl;
00242
00243     ProcessLoop(pInstance);
00244
00245 #if (IMPACT_HAS_WEBSERVER)
00246     // Stopping the webserver
00247     std::cerr « "Stopping web server" « std::endl;
00248     impact_Webserver_Stop(webserver, 1000);
00249     std::cerr « "Stopping web server DONE" « std::endl;
00250 #endif // (IMPACT_HAS_WEBSERVER)
00251
00252     // Stopping the processing instance
00253     std::cerr « "Stopping Update thread" « std::endl;
00254     impact_StopUpdateThread(pInstance); // this will exit the update thread
00255     std::cerr « "Stopping Update thread DONE" « std::endl;
00256
00257     updater.join();
00258
00259     impact_ExitProcess(pInstance);
00260
00261     return 0;
00262 }
00263
00264 }
```

11.7 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]

11.7.1 Function Documentation

11.7.1.1 GetEnv()

```

std::string GetEnv (
    const std::string & name )
```

Definition at line 60 of file [sound4cl-pipe.cpp](#).

11.7.1.2 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 182 of file [sound4cl-pipe.cpp](#).

11.7.1.3 ProcessLoop()

```
void ProcessLoop (
    sound4::CInstance & instance,
    sound4::CProcessor & processor )
```

[Processing Loop]

[Processing Loop]

Definition at line 69 of file [sound4cl-pipe.cpp](#).

11.8 sound4cl-pipe.cpp

[Go to the documentation of this file.](#)

```

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)
00070 {
00071     bool stopthread=false;
00072
00073     #if defined(_WIN32) || defined(WIN32)
00074         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00075         _setmode(_fileno(stdin), _O_BINARY);
00076         _setmode(_fileno(stdout), _O_BINARY);
00077     #endif
00078
00080     size_t input_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00081     size_t output_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00082     while (!stopthread)
00083     {
00084         // Read a chunk from stdin
00085         if (std::fread(instance.GetBufferIn(), sizeof(float), input_sample_count, stdin) !=
input_sample_count) {
00086             std::cerr << "ProcessLoop: input stopped" << std::endl;
00087             stopthread=true;
00088             break;
00089         }
00090         if (stopthread) break;
00091         // Process it
00092         instance.ProcessAudio();
00093         // Write it to stdout
00094         if (fwrite(instance.GetBufferOut(), sizeof(float), output_sample_count, stdout) !=
output_sample_count) {
00095             std::cerr << "Failed to write audio output" << std::endl;
00096             stopthread=true;
00097         }
00098     }
00100 }
00101
00103 class MyPresetManager : public sound4::CPresetLoader {
00104 public:
00105     MyPresetManager() = default;
00106     virtual ~MyPresetManager() = default;
00107
00108     bool SetupPath(const std::filesystem::path& a_preset_path)
00109     {
00110         m_preset_path=a_preset_path;
00111         std::error_code ec;
00112         if ( !std::filesystem::exists(m_preset_path,ec) ) {
00113             return false;
00114         }
00115         if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00116             return false;
00117         }
00118         return true;
00119     }
00120     virtual bool IsReadOnly() override { return false; };
00121     virtual bool Exists(const std::filesystem::path &name) override
00122     {
00123         std::error_code ec;
00124         return std::filesystem::exists(GetRealPath(name), ec);
00125     }
00126     virtual bool Remove(const std::filesystem::path &name) override

```

```

00127     {
00128         std::error_code ec;
00129         return std::filesystem::remove(GetRealPath(name), ec);
00130     }
00131     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
00132     {
00133         std::error_code ec;
00134         std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
00135         return (!ec);
00136     }
00137     virtual std::vector<std::filesystem::path> GetAll() override
00138     {
00139         std::vector<std::filesystem::path> list;
00140         std::error_code ec;
00141         for (auto it = std::filesystem::directory_iterator(m_preset_path.c_str(),
std::filesystem::directory_options::skip_permission_denied, ec);
00142             !ec && it != std::filesystem::directory_iterator{};
00143             it.increment(ec) )
00144         {
00145             if (it->is_regular_file(ec)) {
00146                 list.push_back(it->path().lexically_relative(m_preset_path));
00147             }
00148         }
00149         return list;
00150     }
00151     virtual std::string Read(const std::filesystem::path &filename) override
00152     {
00153         std::ifstream f(GetRealPath(filename));
00154         if ( f.fail() ) {
00155             return std::string();
00156         }
00157         std::ostringstream ss;
00158         ss << f.rdbuf();
00159         return ss.str();
00160     }
00161     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00162     {
00163         std::ofstream f(GetRealPath(filename));
00164         if ( f.fail() ) {
00165             return false;
00166         }
00167         f << content;
00168         if ( f.fail() ) {
00169             return false;
00170         }
00171         return true;
00172     }
00173 private:
00174     std::filesystem::path m_preset_path;
00175     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00176     {
00177         return m_preset_path / name;
00178     }
00179 };
00180
00181
00182 int main(int argc, const char **argv)
00183 {
00184     int listen_port = 8080;
00185     auto store_path_str=GetEnv("S4STORE_PATH");
00187     std::string libname = "libsound4.impact.cl.so";
00188
00189
00190     // Get credentials from environment
00191     auto LoginKey=GetEnv("S4LOGINKEY");
00192     if (LoginKey.empty()) {
00193         LoginKey=GetEnv("S4USERID");
00194     }
00195     auto RadioName=GetEnv("RADIO_NAME");
00196     auto KeyId=GetEnv("ACCESS_KEY_ID");
00197     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00198
00199     const char *programe=argv[0];
00200     for (int n=1;n<argc;n++) {
00201         std::string arg = argv[n];
00202         if (arg=="-p" || arg=="--port") {
00203             if (argc<n+1) {
00204                 std::cerr << "--port needs a number" << std::endl;
00205                 return 1;
00206             }
00207             n++;
00208             listen_port=atoi(argv[n]);
00209         } else if (arg=="-l" || arg=="--lib") {
00210             if (argc<n+1) {
00211                 std::cerr << "--port needs a number" << std::endl;
00212                 return 1;
00213             }
00214             n++;
00215         }

```

```

00216         libname=argv[n];
00217     } else if (arg=="-h" || arg=="--help") {
00218         std::cerr << "Usage: " << progname << " [-p port] [-l libname.so]"<<std::endl;;
00219         std::cerr << "  Reads audio from stdin and return processed audio in stdout" << std::endl;
00220         std::cerr << "  -l, --lib      : library to load ["<<libname<<"]" << std::endl;
00221         std::cerr << "  -p, --port     : port for web server ["<<listen_port<<"]" << std::endl;
00222         std::cerr << std::endl;
00223         std::cerr << "Environment:"<< std::endl;;
00224         std::cerr << "  - S4STORE_PATH          : path to store config [default: (no storage)]"<<
std::endl;;
00225         std::cerr << "  - S4LOGINKEY           : SOUND4 Login key (or S4USERID)"<< std::endl;;
00226         std::cerr << "  - RADIO_NAME          : SOUND4 Radio Name"<< std::endl;;
00227         std::cerr << "  - ACCESS_KEY_ID       : SOUND4 Access Key ID"<< std::endl;;
00228         std::cerr << "  - ACCESS_KEY_SECRET   : SOUND4 Access Key Secret"<< std::endl;
00229         std::cerr << std::endl;
00230         std::cerr << "Example:"<< std::endl;
00231         std::cerr << "  ffmpeg -re -i file.wav -ar 48000 -ac 2 -f f32le - | "<<progname<< " | ffplay
-f f32le -ar 48000 -ac 2 -i -" << std::endl;;
00232         #if defined(_WIN32) || defined(WIN32)
00233         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;;
00234         #else
00235         std::cerr << "  ffmpeg -f pulse -i default -ar 48000 -ac 2 -f f32le - | "<<progname<< " |
ffplay -f f32le -ar 48000 -ac 2 -i -" << std::endl;;
00236         #endif
00237         return 0;
00238     } else {
00239         std::cerr << "unknown argument '"<<arg<< "'"<< std::endl;
00240         return 1;
00241     }
00242 }
00243
00244
00246     sound4::CProcessor processor;
00247     if (!processor.Load(libname)) {
00248         std::cerr << "Failed to load processing library" << std::endl;
00249         return 1;
00250     }
00252
00253     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00254         std::cerr << "Missing credentials in environment" << std::endl;
00255         return 1;
00256     }
00257     std::filesystem::path store_path;
00258     if (!store_path_str.empty()) {
00259         store_path = std::filesystem::path(store_path_str);
00260         std::error_code ec;
00261         if (!std::filesystem::exists(store_path,ec) || !std::filesystem::is_directory(store_path,ec))
{
00262             std::cerr << "Storage folder '"<<store_path.c_str()<< "'" is not a valid directory" <<
std::endl;
00263             return 1;
00264         }
00265     }
00266
00268     processor.SetLogSeverity(sound4::verbose5);
00269     processor.SetLoggerCallback([](sound4::LogSeverity severity, const std::string& msg) {
00270         std::cerr << msg << std::endl;
00271     });
00272
00273     sound4::CInstance instance(processor);
00274     instance.SetParam("ADMIN_USER", "admin");
00275     instance.SetParam("ADMIN_SECRET", "admin");
00277
00279     MyPresetManager preset_manager;
00280     if (!store_path.empty()) {
00281         if (!preset_manager.SetupPath(store_path)) {
00282             std::cerr << "Failed to setup the custom preset manager on path " << store_path.c_str() <<
std::endl;
00283             return 1;
00284         }
00285         instance.SetPresetManager(&preset_manager);
00286     }
00288
00289     std::cerr << "Creating" << std::endl;
00291     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
00292         std::cerr << "Failed to create the processing instance" << std::endl;
00293         return 1;
00294     }
00296
00297     std::cerr << "Starting web server" << std::endl;
00299     if (!instance.StartWebServer(listen_port,0)) {
00300         std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00301         return 1;
00302     }
00304
00305     {

```

```

00307         auto client = instance.NewClient();
00308         auto answer = client->ProcessJson(R"JSON({ "get": { "processorname": null } })JSON");
00309         std::cerr << "Request for processor name returned " << answer << std::endl;
00311     }
00312
00313     ProcessLoop(instance, processor);
00314
00315     std::cerr << "Stopping now" << std::endl;
00316     instance.Stop();
00317
00318     std::cerr << "End" << std::endl;
00319     return 0;
00320 }

```

11.9 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 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)
- 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_CCClientInstance](#) * [impact_NewClient](#) (struct [impact_CInstance](#) *instance)
- void [impact_DeleteClient](#) (struct [impact_CCClientInstance](#) *client)
- const char * [impact_ProcessJson](#) (struct [impact_CCClientInstance](#) *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)
- void [impact_Webserver_SetAppHealth](#) (struct [impact_CInstance](#) *instance, int httpcode, const char *contenttype, const char *content)
- void [impact_Webserver_GetAppHealth](#) (struct [impact_CInstance](#) *instance, int *httpcode, char **contenttype, char **content)
- void [impact_Webserver_FreeString](#) (char *str)

11.9.1 Macro Definition Documentation

11.9.1.1 IMPACT_DECL

```
#define IMPACT_DECL __declspec(dllimport)
```

Definition at line 14 of file [sound4.impact.cl.h](#).

11.9.1.2 IMPACT_HAS_CLOUDBUS

```
#define IMPACT_HAS_CLOUDBUS 0
```

Definition at line 22 of file [sound4.impact.cl.h](#).

11.9.1.3 IMPACT_HAS_WEBSERVER

```
#define IMPACT_HAS_WEBSERVER 1
```

Definition at line 21 of file [sound4.impact.cl.h](#).

11.10 sound4.impact.cl.h

[Go to the documentation of this file.](#)

```
00001
00004 #pragma once
00005
00006 #include <stdint.h>
00007 #ifdef __unix__
00008     #include <sys/types.h>
00009     #define IMPACT_DECL __attribute__((visibility ("default")))
00010 #elif defined(_WIN32) || defined(WIN32)
00011     #ifdef BUILDING_IMPACT_DLL
00012         #define IMPACT_DECL __declspec(dllexport)
00013     #else
00014         #define IMPACT_DECL __declspec(dllimport)
00015     #endif
00016 #elif defined(__APPLE__)
00017     #include <sys/types.h>
00018     #define IMPACT_DECL __attribute__((visibility ("default")))
00019 #endif
00020
00021 #define IMPACT_HAS_WEBSERVER 1
00022 #define IMPACT_HAS_CLOUDBUS 0
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif // __cplusplus
00027
00073 // -----
00092 // -----
00158 // -----
00159
00160 // -----
00172 #define IMPACT_AUDIOFRAME_COUNT 64
00178 #define IMPACT_CHANNEL_COUNT 2
00184 #define IMPACT_AUDIO_INPUT_COUNT 1
00190 #define IMPACT_AUDIO_OUTPUT_COUNT 1
00196 #define IMPACT_SAMPLE_RATE 48000
00197
00203 struct SOUND4_ProcessInfo {
00204     size_t struct_size;
00205     const char *name;
00206     const char *prefix;
00207     const char *shortname;
00208     const char *version;
00209 };
00210
00218 IMPACT_DECL const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo();
00219
00225 IMPACT_DECL const char *impact_GetVersion();
00226
00239 IMPACT_DECL unsigned int impact_GetChunkSizeInFrames();
00240
00249 IMPACT_DECL unsigned int* impact_GetPossibleChunkSizeInFrames();
00250
00258 IMPACT_DECL unsigned int impact_GetChannelCount();
00259
00268 IMPACT_DECL unsigned int impact_GetAudioInputCount();
00269
00278 IMPACT_DECL unsigned int impact_GetAudioOutputCount();
00279
```

```

00287 IMPACT_DECL unsigned int impact_GetSampleRate();
00288
00292 enum impact_LogSeverity {
00293     none = 0,
00294     fatal = 1,
00295     error = 2,
00296     warning = 3,
00297     info = 4,
00298     verbose = 5,
00299     verbose2 = 6,
00300     verbose3 = 7,
00301     verbose4 = 8,
00302     verbose5 = 9,
00303 };
00304
00311 typedef void (*impact_loggerfn) (enum impact_LogSeverity, const char *);
00312
00322 IMPACT_DECL void impact_SetLoggerCallback( impact_loggerfn logger );
00323
00331 IMPACT_DECL void impact_SetLogSeverity( enum impact_LogSeverity severity );
00332
00335 // -----
00341 // -----
00345 struct impact_CInstance;
00346
00350 struct impact_CParameters;
00351
00359 IMPACT_DECL struct impact_CParameters *impact_NewParameters();
00360
00364 IMPACT_DECL void impact_FreeParameters(struct impact_CParameters *params);
00365
00377 IMPACT_DECL void impact_SetParameter(struct impact_CParameters *params, const char *name, const char
    *value);
00378
00384 IMPACT_DECL const char *impact_GetParameter(struct impact_CParameters *params, const char *name);
00385
00391 IMPACT_DECL void impact_FreeParameterValue(const char *value);
00392
00400 IMPACT_DECL struct impact_CInstance *impact_InitProcess(
00401     const char *LoginKey,
00402     const char *RadioName,
00403     const char *Access_Key_ID,
00404     const char *Access_Key_Secret,
00405     const char *save_path
00406 );
00407
00415 IMPACT_DECL struct impact_CInstance *impact_InitProcess2(
00416     const char *LoginKey,
00417     const char *RadioName,
00418     const char *Access_Key_ID,
00419     const char *Access_Key_Secret,
00420     const char *save_path,
00421     const struct impact_CParameters *parameters
00422 );
00423
00468 IMPACT_DECL struct impact_CInstance *impact_InitProcess3(
00469     const char *LoginKey,
00470     const char *RadioName,
00471     const char *Access_Key_ID,
00472     const char *Access_Key_Secret,
00473     const char *save_path,
00474     const struct impact_CParameters *parameters,
00475     unsigned int frames_per_chunk
00476 );
00477
00489 IMPACT_DECL unsigned int impact_GetProcessChunkFrames(struct impact_CInstance *instance);
00490
00500 IMPACT_DECL int impact_TerminateProcess(struct impact_CInstance *instance);
00501
00507 IMPACT_DECL void impact_ExitProcess(struct impact_CInstance *instance);
00508
00511 #if IMPACT_HAS_CLOUDBUS
00512 // -----
00521 struct impact_CBus;
00522
00528 IMPACT_DECL struct impact_CBus* impact_NewBus();
00529
00535 IMPACT_DECL void impact_FreeBus(struct impact_CBus *bus);
00536
00546 IMPACT_DECL void impact_SetInstanceBus(struct impact_CParameters *params, struct impact_CBus *bus);
00547
00548
00550 #endif // IMPACT_HAS_CLOUDBUS
00551
00552 // -----
00575 // -----
00576

```

```

00577 // OS dependent filesystem path char type
00578 #ifdef _WIN32
00579     typedef wchar_t fs_char;
00580 #else // !_WIN32
00581     typedef char fs_char;
00582 #endif // !_WIN32
00583
00591 typedef char * (*impact_storage_reader) (const fs_char *filename, void* userdata);
00592
00599 typedef void (*impact_storage_reader_free) (char *content, void* userdata);
00600
00609 typedef int (*impact_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00610
00618 typedef int (*impact_storage_exists) (const fs_char *filename, void* userdata);
00619
00627 typedef fs_char** (*impact_storage_getall) (void* userdata);
00628
00635 typedef void (*impact_storage_getall_free) (fs_char**all, void* userdata);
00636
00644 typedef int (*impact_storage_remove) (const fs_char *filename, void* userdata);
00645
00654 typedef int (*impact_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00655
00673 IMPACT_DECL void impact_SetPresetManager(struct impact_CParameters *params
00674     , impact_storage_reader reader, impact_storage_reader_free
00675     , impact_storage_writer writer
00676     , impact_storage_exists exists
00677     , impact_storage_getall getall, impact_storage_getall_free getall_free
00678     , impact_storage_remove remove
00679     , impact_storage_rename rename
00680     , int IsReadOnly
00681     , void* userdata
00682 );
00683
00687 enum impact_PresetChange_Kind {
00688     change_kind_created = 1,
00689     change_kind_modified ,
00690     change_kind_deleted ,
00691 };
00692
00701 IMPACT_DECL void impact_PresetManager_InformChange(struct impact_CInstance *instance, const fs_char
00702 *relative_path, enum impact_PresetChange_Kind change_kind);
00703
00705 // -----
00713 // -----
00714
00731 IMPACT_DECL void impact_SetMetadata(struct impact_CInstance *instance, const char* key, const char*
00732 value);
00733
00743 IMPACT_DECL void impact_SetMetadataMulti(struct impact_CInstance *instance, const char** keyvalue);
00744
00752 IMPACT_DECL const char** impact_GetMetadataInfos(struct impact_CInstance *instance);
00753
00760 IMPACT_DECL void impact_FreeMetadataInfos(struct impact_CInstance *instance, const char** infos);
00761
00764 // -----
00769 // -----
00770
00783 IMPACT_DECL void impact_StartUpdateThread(struct impact_CInstance *instance, unsigned int port);
00784
00790 IMPACT_DECL void impact_StopUpdateThread(struct impact_CInstance *instance);
00791
00799 IMPACT_DECL int impact_WaitUpdateThreadReady(struct impact_CInstance *instance, int milliseconds);
00800
00803 // -----
00808 // -----
00830 IMPACT_DECL void impact_ProcessAudio(struct impact_CInstance *instance, const float *input, float
00831 *output );
00832
00850 IMPACT_DECL void impact_ProcessAudio_Planar(struct impact_CInstance *instance
00851     , float const * const *input
00852     , float * const *output);
00853
00866 IMPACT_DECL float *impact_GetBufferIn(struct impact_CInstance *instance);
00867
00880 IMPACT_DECL float *impact_GetBufferOut(struct impact_CInstance *instance);
00881
00889 IMPACT_DECL unsigned int impact_GetEstimatedDelay(struct impact_CInstance *instance);
00890
00893 // -----
00900 // -----
00901 #ifdef _MSC_VER
00902     #define __BYTE_ORDER__ REG_DWORD
00903     #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00904     #define __ORDER_BIG_ENDIAN__ REG_DWORD_BIG_ENDIAN
00905 #endif
00906

```

```

00910 enum impact_SampleFormat {
00911     INVALID_FORMAT=0,
00912     S16_LE,
00913     S16_BE,
00914     S24_LE,
00915     S24_BE,
00916     S32_LE,
00917     S32_BE,
00918     F32_LE,
00919     F32_BE,
00920
00921     // Native definitions, only links to the physical ones
00922     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00923         S16_NATIVE=S16_LE,
00924         S24_NATIVE=S24_LE,
00925         S32_NATIVE=S32_LE,
00926         F32_NATIVE=F32_LE,
00927     #else
00928         S16_NATIVE=S16_BE,
00929         S24_NATIVE=S24_BE,
00930         S32_NATIVE=S32_BE,
00931         F32_NATIVE=F32_BE,
00932     #endif
00933 };
00934
00941 IMPACT_DECL const char *impact_GetFormatName(const enum impact_SampleFormat fmt);
00942
00949 IMPACT_DECL enum impact_SampleFormat impact_GetFormatFromName(const char *name);
00950
00957 IMPACT_DECL unsigned int impact_GetBytesFromFormat(const enum impact_SampleFormat fmt);
00958
00959
00966 IMPACT_DECL unsigned int impact_GetMaxPacketFrame(struct impact_CInstance *instance);
00967
00978 IMPACT_DECL unsigned int impact_AddAudio(struct impact_CInstance *instance, const uint8_t *payload,
    unsigned int nFrame, enum impact_SampleFormat fmt );
00979
00986 IMPACT_DECL unsigned int impact_AddPadAudio(struct impact_CInstance *instance);
00987
00994 IMPACT_DECL unsigned int impact_GetOutputCount(struct impact_CInstance *instance);
00995
01005 IMPACT_DECL unsigned int impact_GetAudio(struct impact_CInstance *instance, uint8_t *payload, unsigned
    int max_nFrame, enum impact_SampleFormat fmt );
01006
01010 // -----
01014 // -----
01025 IMPACT_DECL void impact_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
    impact_SampleFormat fmt);
01026
01037 IMPACT_DECL void impact_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
    impact_SampleFormat fmt);
01038
01048 IMPACT_DECL void impact_StereoToMono(const float *input, float *output, size_t nFrame);
01049
01059 IMPACT_DECL void impact_MonoToStereo(const float *input, float *output, size_t nFrame);
01060
01071 IMPACT_DECL void impact_StereoToMono_Planar(const float *inputL, const float *inputR, float *output,
    size_t nFrame);
01072
01083 IMPACT_DECL void impact_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t
    nFrame);
01084
01091 IMPACT_DECL void impact_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
01092
01099 IMPACT_DECL void impact_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
01100
01103 // -----
01108 // -----
01112 struct impact_CClientInstance;
01113
01122 IMPACT_DECL struct impact_CClientInstance *impact_NewClient(struct impact_CInstance *instance);
01123
01129 IMPACT_DECL void impact_DeleteClient(struct impact_CClientInstance *client);
01130
01144 IMPACT_DECL const char *impact_ProcessJson(struct impact_CClientInstance *client, const char
    *json_str, int *need_save);
01145
01151 IMPACT_DECL void impact_FreeJsonAnswer(const char *json_str);
01152
01164 IMPACT_DECL int impact_SaveState(struct impact_CInstance *instance);
01165
01461 #if IMPACT_HAS_WEBSERVER
01462 // -----
01475 // -----
01494 IMPACT_DECL uint64_t impact_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure,
    const char *socket_ip, unsigned int socket_port);
01495

```

```

01496 // -----
01516 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);
01517
01535 IMPACT_DECL uint64_t impact_Webserver(unsigned int listenport, unsigned int listenport_secure, struct
impact_CInstance *instance);
01536
01546 IMPACT_DECL int impact_Webserver_Stop(uint64_t id, int timeout_ms);
01547
01556 IMPACT_DECL int impact_Webserver_Status(uint64_t id);
01557
01568 IMPACT_DECL void impact_Webserver_SetAppHealth(struct impact_CInstance *instance, int httpcode, const
char *contenttype, const char *content);
01569
01578 IMPACT_DECL void impact_Webserver_GetAppHealth(struct impact_CInstance *instance, int* httpcode,
char** contenttype, char** content);
01579
01585 IMPACT_DECL void impact_Webserver_FreeString(char *str);
01586
01591 #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
01592
01596 #define SOUND4_WEBSERVER_HTTP_OK (1<<0)
01600 #define SOUND4_WEBSERVER_HTTPS_OK (1<<1)
01601
01603 #endif // IMPACT_HAS_WEBSERVER
01604
01605 #ifndef __cplusplus
01606 } // extern "C"
01607 #endif // __cplusplus

```

11.11 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

- namespace [sound4](#)
- namespace [sound4::impact](#)
- namespace [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` = 64
- `const size_t sound4::impact::InputChannels` = 2 * 1
- `const size_t sound4::impact::OutputChannels` = 2 * 1
- `const size_t sound4::impact::InputSampleSize` = `ChunkFrames*InputChannels`
- `const size_t sound4::impact::OutputSampleSize` = `ChunkFrames*OutputChannels`
- `const size_t sound4::impact::SampleRate` = 48000

11.11.1 Macro Definition Documentation

11.11.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 25 of file `sound4.impact.cl.hpp`.

11.12 sound4.impact.cl.hpp

[Go to the documentation of this file.](#)

```
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,
00092             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);
00105             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00106             return wstr;
00107         }
00108         #endif // _WIN32
00109
00115         template<typename T>
00116         struct SampleFormat {
00117         };
00118         template<>
00119         struct SampleFormat<int16_t> {
00120             const impact_SampleFormat format = S16_NATIVE;
00121         };
00122         template<>
00123         struct SampleFormat<int32_t> {
00124             const impact_SampleFormat format = S32_NATIVE;
00125         };
00126         template<>
00127         struct SampleFormat<float> {
00128             const impact_SampleFormat format = F32_NATIVE;
00129         };
00130
00131
00136         static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
00137         impact_SampleFormat fmt) {
00137             impact_AudioConvertFrom(payload, output, nSpl, fmt);
00138         }
00139         static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl,
00140         impact_SampleFormat fmt) {
00140             impact_AudioConvertTo(input, payload, nSpl, fmt);
00141         }
00142         static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
00143             impact_StereoToMono(input, output, nSpl);
00144         }
00145         static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
00146             impact_MonoToStereo(input, output, nSpl);
00147         }
00148         static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float
00149         *output, size_t nSpl) {
00149             impact_StereoToMono_Planar(inputL, inputR, output, nSpl);
00150         }
00151         static inline void MonoToStereo_Planar(const float *input, float *outputL, float *outputR,
00152         size_t nSpl) {
00152             impact_MonoToStereo_Planar(input, outputL, outputR, nSpl);
00153         }
00154         static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
00155             impact_AudioMonoFromLiveStereo(payload, output);
00156         }
00157         static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
00158             impact_AudioMonoToLiveStereo(input, payload);
00159         }

```

```

00160
00161     }; // namespace helper
00162
00163     static inline std::string GetVersion()
00164     {
00165         return std::string(impact_GetVersion());
00166     }
00167
00168     static inline unsigned int GetChunkSizeInFrames()
00169     {
00170         return impact_GetChunkSizeInFrames();
00171     }
00172
00173     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00174     {
00175         std::vector<unsigned int> list;
00176         for (unsigned int* src=impact_GetPossibleChunkSizeInFrames();*src;*src++) {
00177             list.push_back(*src);
00178         }
00179         return list;
00180     }
00181
00182     static inline unsigned int GetChannelCount()
00183     {
00184         return impact_GetChannelCount();
00185     }
00186
00187     static inline unsigned int GetAudioInputCount()
00188     {
00189         return impact_GetAudioInputCount();
00190     }
00191
00192     static inline unsigned int GetAudioOutputCount()
00193     {
00194         return impact_GetAudioOutputCount();
00195     }
00196
00197     static inline unsigned int GetSampleRate()
00198     {
00199         return impact_GetSampleRate();
00200     }
00201
00202     static inline void SanityCheck(bool a_bCheckFrames = true)
00203     {
00204         // Do some sanity checks
00205         if (impact_GetSampleRate() != IMPACT_SAMPLE_RATE) {
00206             throw std::runtime_error("Bad library sampling rate");
00207         }
00208         if (a_bCheckFrames && impact_GetChunkSizeInFrames() != ChunkFrames) {
00209             throw std::runtime_error("Bad library frame size");
00210         }
00211         if (impact_GetChannelCount() != IMPACT_CHANNEL_COUNT) {
00212             throw std::runtime_error("Bad library channel count");
00213         }
00214         if (impact_GetAudioInputCount() != IMPACT_AUDIO_INPUT_COUNT) {
00215             throw std::runtime_error("Bad library input count");
00216         }
00217         if (impact_GetAudioOutputCount() != IMPACT_AUDIO_OUTPUT_COUNT) {
00218             throw std::runtime_error("Bad library output count");
00219         }
00220     }
00221
00222     static inline std::string GetFormatName(const SampleFormat fmt)
00223     {
00224         return std::string(impact_GetFormatName(fmt));
00225     }
00226
00227     static inline SampleFormat GetFormatFromName(const std::string& name)
00228     {
00229         return impact_GetFormatFromName(name.c_str());
00230     }
00231
00232     static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
00233     {
00234         return impact_GetBytesFromFormat(fmt);
00235     }
00236
00237     using LogSeverity = impact_LogSeverity;
00238     using log_cb_t = std::function<void(LogSeverity, const std::string&)>;
00239
00240     static log_cb_t _log_cb;
00241     extern "C" {
00242         static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
00243             _log_cb(severity, std::string(c_msg));
00244         }
00245     }
00246
00247     static inline void SetLogSeverity(LogSeverity severity)

```

```

00335     {
00336         impact_SetLogSeverity(severity);
00337     }
00341     static inline void SetLoggerCallback(log_cb_t cb)
00342     {
00343         _log_cb=cb;
00344         impact_SetLoggerCallback(&_log_cb_c);
00345     }
00346
00347 #if IMPACT_HAS_CLOUDBUS
00348     class CBus {
00349     public:
00350         CBus()
00351             : m_bus(impact_NewBus()) {}
00352         ~CBus()
00353         {
00354             impact_FreeBus(m_bus);
00355             m_bus = nullptr;
00356         }
00357         impact_CBus *Get() const { return m_bus; }
00358     private:
00359         impact_CBus *m_bus;
00360     };
00361 #endif // IMPACT_HAS_CLOUDBUS
00362
00363     class CPresetLoader {
00364     public:
00365         CPresetLoader() = default;
00370         virtual ~CPresetLoader() = default;
00371
00372         virtual bool IsReadOnly() = 0;
00373         virtual bool Exists(const std::filesystem::path &name) = 0;
00374         virtual bool Remove(const std::filesystem::path &name) = 0;
00375         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00376         virtual std::vector<std::filesystem::path> GetAll() = 0;
00377         virtual std::string Read(const std::filesystem::path &filename) = 0;
00378         virtual bool Write(const std::filesystem::path &filename, const std::string &content) = 0;
00379     };
00380     using PresetChange_Kind = impact_PresetChange_Kind;
00381
00382     class CInstance {
00383     public:
00384         CInstance()
00385         {
00386             params = impact_NewParameters();
00387             if (!params) throw std::bad_alloc();
00388         }
00389         // non-copyable
00390         CInstance( const CInstance& ) = delete;
00391         CInstance& operator=( const CInstance& ) = delete;
00392         virtual ~CInstance()
00393         {
00394             Stop();
00395             if (params) impact_FreeParameters(params);
00396         }
00397     bool IsOk() const { return instance!=nullptr; }
00400     void SetParam(const std::string& name, const std::string& value)
00401     {
00402         assert(params);
00403         impact_SetParameter(params, name.c_str(), value.c_str());
00404     }
00405     std::string GetParam(const std::string& name)
00406     {
00407         assert(params);
00408         auto c_value=impact_GetParameter(params, name.c_str());
00409         std::string ret(c_value);
00410         impact_FreeParameterValue(c_value);
00411         return ret;
00412     }
00413
00414     void SetPresetManager(CPresetLoader *preset_manager)
00415     {
00416         if (!preset_manager) return;
00417         assert(params);
00418         impact_SetPresetManager(params,
00419             impact_custom_reader,
00420             impact_custom_reader_free,
00421             impact_custom_writer,
00422             impact_custom_exists,
00423             impact_custom_getall,
00424             impact_custom_getall_free,
00425             impact_custom_remove,
00426             impact_custom_rename,
00427             preset_manager->IsReadOnly(),
00428             preset_manager
00429         );
00430     }
00431

```

```

00462
00463 #if IMPACT_HAS_CLOUDBUS
00464     void SetBus(const CBus& bus) {
00465         assert(params);
00466         impact_SetInstanceBus(params,bus.Get());
00467     }
00468 #endif // IMPACT_HAS_CLOUDBUS
00469
00470 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)
00471 {
00472     assert(!instance);
00473     std::string l_save_path;
00474     if (!save_path.empty()) {
00475         l_save_path = save_path.u8string();
00476     }
00477     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);
00478     if (!instance) return false;
00479     if (!init_metadata.empty()) {
00480         for (auto&& [key,value]: init_metadata) {
00481             impact_SetMetadata(instance, key.c_str(), value.c_str());
00482         }
00483         init_metadata.clear();
00484     }
00485     update_thread = std::thread([this, json_port]() {
00486         OnUpdateThreadStart();
00487         impact_StartUpdateThread(instance, json_port);
00488         OnUpdateThreadStop();
00489     });
00490     int timeout=5000;
00491     #ifdef DEBUG
00492         timeout=60*1000; // For debugger pauses
00493     #endif
00494     if (impact_WaitUpdateThreadReady(instance, timeout)<0) {
00495         return false;
00496     }
00497     return true;
00498 }
00499
00500 unsigned int GetChunkFrames() {
00501     if (instance) {
00502         return impact_GetProcessChunkFrames(instance);
00503     }
00504     return 0;
00505 }
00506
00507 void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00508 {
00509     impact_PresetManager_InformChange(instance, relative_path.c_str(),
(impact_PresetChange_Kind) change_kind);
00510 }
00511
00512 void SetMetadata(const std::string& key, const char* value)
00513 {
00514     if (instance) {
00515         impact_SetMetadata(instance, key.c_str(), value);
00516     } else {
00517         init_metadata.push_back( {key, value} );
00518     }
00519 }
00520
00521 void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00522 {
00523     if (instance) {
00524         typedef const char* pchar;
00525         pchar* keyvalue=new pchar[2*list.size()+1];
00526         size_t n=0;
00527         for (auto&& [key,value]: list) {
00528             keyvalue[2*n+0]=key.c_str();
00529             keyvalue[2*n+1]=value;
00530             n++;
00531         }
00532         keyvalue[2*n+0]=nullptr;
00533         impact_SetMetadataMulti(instance, keyvalue);
00534         delete[] keyvalue;
00535     } else {
00536         for (auto&& [key,value]: list) {
00537             if (value) {
00538                 init_metadata.push_back( {key, value} );
00539             }
00540         }
00541     }
00542 }

```

```

00595     }
00596
00602     std::vector< std::tuple<std::string, std::string> > GetMetadataInfos()
00603     {
00604         std::vector< std::tuple<std::string, std::string> > values;
00605         const char** c_values = impact_GetMetadataInfos(instance);
00606         if (c_values) {
00607             for (const char** c_value=c_values; *c_value; ) {
00608                 std::string key(*c_value);
00609                 c_value++;
00610                 if (*c_value) {
00611                     std::string descr(*c_value);
00612                     c_value++;
00613                     values.push_back( {key, descr} );
00614                 }
00615             }
00616             impact_FreeMetadataInfos(instance, c_values);
00617         }
00618         return values;
00619     }
00620
00621     // Can be overloaded, called at start of UpdateThread in the thread
00622     virtual void OnUpdateThreadStart() {}
00623     // Can be overloaded to call at end of UpdateThread in the thread
00624     virtual void OnUpdateThreadStop() {}
00625
00626 #if IMPACT_HAS_WEBSERVER
00632     bool StartWebServer(int http_port, int https_port=0)
00633     {
00634         assert(instance);
00635         assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
00636         webserver = impact_Webserver(http_port, https_port, instance);
00637         if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00638             return false;
00639         }
00640         int web_status=impact_Webserver_Status(webserver);
00641         if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
00642             return false;
00643         }
00644         if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
00645             return false;
00646         }
00647         return true;
00648     }
00654     void StopWebServer()
00655     {
00656         if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
00657             impact_Webserver_Stop(webserver, 1000);
00658             webserver = SOUND4_INVALID_WEBSERVER_ID;
00659         }
00660     }
00661
00665     void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
content)
00666     {
00667         assert(instance);
00668         impact_Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
content.c_str());
00669     }
00673     void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
00674     {
00675         assert(instance);
00676         char* c_contenttype=nullptr;
00677         char* c_content=nullptr;
00678         impact_Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
00679         contenttype=c_contenttype;
00680         content=c_content;
00681         impact_Webserver_FreeString(c_contenttype);
00682         impact_Webserver_FreeString(c_content);
00683     }
00684 #endif // IMPACT_HAS_WEBSERVER
00685
00692     int Terminate()
00693     {
00694         if (!instance) return 0;
00695         return impact_TerminateProcess(instance);
00696     }
00697
00703     void Stop()
00704     {
00705         if (!instance) return;
00706 #if IMPACT_HAS_WEBSERVER
00707         StopWebServer();
00708 #endif // IMPACT_HAS_WEBSERVER
00709         if (update_thread.joinable()) {
00710             impact_StopUpdateThread(instance); // this will exit the update thread
00711             update_thread.join();

```

```

00712         }
00713         impact_ExitProcess(instance);
00714         instance=nullptr;
00715     }
00716
00722     unsigned int GetEstimatedDelay()
00723     {
00724         assert(instance);
00725         return impact_GetEstimatedDelay(instance);
00726     }
00727
00738     std::array<float,InputSampleSize>& GetBufferIn()
00739     {
00740         assert(instance);
00741         float *buf=impact_GetBufferIn(instance);
00742         return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
00743     }
00744
00755     std::array<float,OutputSampleSize>& GetBufferOut()
00756     {
00757         assert(instance);
00758         float *buf= impact_GetBufferOut(instance);
00759         return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
00760     }
00766     void ProcessAudio()
00767     {
00768         assert(instance);
00769         impact_ProcessAudio(instance, impact_GetBufferIn(instance),
00770         impact_GetBufferOut(instance));
00771     }
00784     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
00785     InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
00786     void ProcessAudio_Planar(float const * const *input, float * const *output)
00787     {
00788         assert(instance);
00789         impact_ProcessAudio_Planar(instance, input, output);
00790     }
00802     template<typename T>
00803     std::vector<T> ProcessAnyAudio(const std::vector<T> input)
00804     {
00805         assert(instance);
00806         std::vector<T> output;
00807         unsigned int out_offset=0;
00808         unsigned int in_offset=0;
00809         unsigned int todo = input.size();
00810         while (todo>0) {
00811             unsigned int left = AddAudio(&input[in_offset], todo);
00812             unsigned int out_avail = impact_GetOutputCount(instance);
00813             output.resize(out_offset + out_avail);
00814             GetAudio(&output[out_offset], out_avail);
00815             out_offset+=out_avail;
00816             in_offset += todo-left;
00817             todo=left;
00818         }
00819         return output;
00820     }
00821
00827     class CClient {
00828     public:
00829         CClient(impact_CInstance* instance)
00830         {
00831             assert(instance);
00832             client=impact_NewClient(instance);
00833             if (!client) throw std::bad_alloc();
00834         }
00835         // non-copyable
00836         CClient( const CInstance& ) = delete;
00837         CClient& operator=( const CInstance& ) = delete;
00838         ~CClient()
00839         {
00840             if (client) {
00841                 impact_DeleteClient(client);
00842                 client=nullptr;
00843             }
00844         }
00852     std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
00853     {
00854         assert(client);
00855         int need_save=0;
00856         const char *canswer = impact_ProcessJson(client, request.c_str(), &need_save);
00857         if (!canswer) return {};
00858         std::string answer(canswer);
00859         impact_FreeJsonAnswer (canswer);
00860         if (NeedSave) {
00861             *NeedSave=(need_save!=0);
00862         }

```

```

00863         return answer;
00864     }
00865     private:
00866         impact_CClientInstance *client = nullptr;
00867     };
00875     std::shared_ptr<CClient> NewClient()
00876     {
00877         assert(instance);
00878         return std::make_shared<CClient>(instance);
00879     }
00880
00881     protected:
00882         template<typename T>
00883         unsigned int AddAudio(const T* payload, unsigned int nFrame)
00884         {
00885             assert(instance);
00886             return impact_AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
00887 helper::SampleFormat<T>::format);
00888         }
00889         template<typename T>
00889         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
00890         {
00891             assert(instance);
00892             return impact_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
00893 helper::SampleFormat<T>::format);
00894         }
00895     private:
00896         impact_CParameters* params = nullptr;
00897         impact_CInstance* instance = nullptr;
00898         std::thread update_thread;
00899         std::vector< std::pair<std::string, std::string> > init_metadata;
00900 #if IMPACT_HAS_WEBSERVER
00901         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
00902 #endif // IMPACT_HAS_WEBSERVER
00903     };
00904
00905 }; // namespace impact
00906 }; // namespace sound4
00907
00908 // bridge C callbacks to CPresetLoader
00909 extern "C" {
00910     static char *impact_custom_reader(const fs_char *filename, void* userdata) {
00911         sound4::impact::CPresetLoader
00912 *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00913         std::string content=preset_loader->Read(filename);
00914         return strdup(content.c_str());
00915     }
00916     static void impact_custom_reader_free(char *content, void* userdata) {
00917         if (!content) return;
00918         free(content);
00919     }
00919     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
00920         sound4::impact::CPresetLoader
00921 *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00922         auto res=preset_loader->Write(filename,content);
00923         return res?0:-1;
00924     }
00924     static int impact_custom_exists(const fs_char *filename, void* userdata) {
00925         sound4::impact::CPresetLoader
00926 *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00927         auto res=preset_loader->Exists(filename);
00928         return res?0:-1;
00929     }
00929     static fs_char** impact_custom_getall(void* userdata) {
00930         sound4::impact::CPresetLoader
00931 *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00932         auto res=preset_loader->GetAll();
00933         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
00934         for (size_t n=0;n<res.size();n++) {
00935             all[n]=fs_strdup(res[n].c_str());
00936         }
00937         all[res.size()]=nullptr;
00938         return all;
00939     }
00939     static void impact_custom_getall_free(fs_char** all, void* userdata) {
00940         if (!all) return;
00941         for (fs_char** one=all;*one;one++) {
00942             free(*one);
00943         }
00944         free(all);
00945     }
00946     static int impact_custom_remove(const fs_char *filename, void* userdata) {
00947         sound4::impact::CPresetLoader
00948 *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00949         auto res=preset_loader->Remove(filename);
00950         return res?0:-1;

```

```

00950     }
00951     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
00952         sound4::impact::CPresetLoader
    *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00953         auto res=preset_loader->Rename(from,to);
00954         return res?0:-1;
00955     }
00956
00957 };

```

11.13 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

- namespace [sound4](#)
- namespace [sound4::impact](#)
- namespace [dyn](#)
- namespace [sound4::impact::dyn](#)
- namespace [sound4::impact::dyn::helper](#)

Macros

- `#define UNIXLIKE 0`
- `#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 = 64`
- `const size_t sound4::impact::dyn::InputChannels = 2 * 1`
- `const size_t sound4::impact::dyn::OutputChannels = 2 * 1`
- `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 = 48000`

11.13.1 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](#).

11.13.2 Macro Definition Documentation

11.13.2.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 42 of file [sound4.impact.cl_dyn.hpp](#).

11.13.2.2 UNIXLIKE

```
#define UNIXLIKE 0
```

Definition at line 23 of file [sound4.impact.cl_dyn.hpp](#).

11.14 sound4.impact.cl_dyn.hpp

[Go to the documentation of this file.](#)

```

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 #if defined(__unix__) || defined(__APPLE__)
00021     #define UNIXLIKE 1
00022 #else
00023     #define UNIXLIKE 0
00024 #endif
00025
00026 #if UNIXLIKE
00027     #include <dlfcn.h>
00028 #elif defined(_WIN32)
00029     #include <libloaderapi.h>
00030 #else
00031     #error "Unsupported OS"
00032 #endif
00033
00034 // Needed to have function declarations
00035 #include "sound4.impact.cl.h"
00036
00037
00038 #ifdef _WIN32
00039     #include <stringapiset.h>
00040     #include <wchar.h>
00042     #define fs_strdup wcsdup
00043 #else // !_WIN32
00044     #define fs_strdup strdup
00046 #endif // !_WIN32
00047
00048 // bridge C callbacks to CPresetLoader
00049 extern "C" {
00050     static char *impact_custom_reader(const fs_char *filename, void* userdata);
00051     static void impact_custom_reader_free(char *content, void* userdata);
00052     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata);
00053     static int impact_custom_exists(const fs_char *filename, void* userdata);
00054     static fs_char** impact_custom_getall(void* userdata);
00055     static void impact_custom_getall_free(fs_char** all, void* userdata);
00056     static int impact_custom_remove(const fs_char *filename, void* userdata);
00057     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00058 };
00059
00063 namespace sound4 {
00067 namespace impact {
00071 namespace dyn {
00072
00076     namespace helper {
00077         // Storage definition, depending on OS
00078         #if UNIXLIKE
00079             using DynLib_t = void *;
00080             using DynFunc_t = void *;
00081         #else
00082             using DynLib_t = HMODULE;
00083             using DynFunc_t = FARPROC;
00084         #endif
00085
00091         template <typename T>
00092         class DynFuncHolder {
00093         private:
00094             DynFunc_t m_ptr = NULL;
00095         public:
00096             DynFuncHolder() = default;
00097             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00098             operator bool() const { return (m_ptr!=NULL); }
00099             bool IsOk() const { return m_ptr!=NULL; }
00100             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00101         };
00102
00107         class CDynLoader {
00108         private:
00109             DynLib_t m_lib {};
00110         public:

```

```

00111         CDynLoader() = default;
00112         CDynLoader(CDynLoader&&) = default;
00113         // No copy allowed
00114         CDynLoader(const CDynLoader&) = delete;
00115         CDynLoader& operator=(CDynLoader const&) = delete;
00116         operator bool() const {
00117             return (m_lib!=NULL);
00118         }
00119         bool IsOk() const {
00120             return (m_lib!=NULL);
00121         }
00122         #if UNIXLIKE
00123         void Close() {
00124             if (IsOk()) {
00125                 #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00126                     dlclose(m_lib);
00127                 #endif // __SANITIZE_ADDRESS__
00128                 m_lib=NULL;
00129             }
00130         }
00131         bool Load(const std::filesystem::path& dynlib)
00132         {
00133             Close();
00134             m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00135             return IsOk();
00136         }
00137         template <typename T>
00138         DynFuncHolder<T> GetSymbol(const std::string& name) {
00139             auto ptr=dlsym(m_lib, name.c_str());
00140             if (!ptr) {
00141                 throw std::runtime_error("Missing function in library");
00142             }
00143             return DynFuncHolder<T>(ptr);
00144         }
00145         template <typename T>
00146         DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00147             auto ptr=dlsym(m_lib, name.c_str());
00148             return DynFuncHolder<T>(ptr);
00149         }
00150         static std::filesystem::path GetThisLibraryPath(void) {
00151             static Dl_info info;
00152             if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00153             auto rp=realpath(info.dli_fname, NULL);
00154             std::filesystem::path p(rp);
00155             free(rp);
00156             return p;
00157         }
00158         #else
00159         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_
00160         {
00161             Close();
00162             m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00163             return IsOk();
00164         }
00165         void Close() {
00166             if (IsOk()) {
00167                 // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
FreeLibraryAndExitThread but...
00168                 // So define SOUND4_CALL_FREEELIBRARY before including this file if you
really want to unload the library dynamically
00169                 #if defined(SOUND4_CALL_FREEELIBRARYANDEXITTHREAD)
FreeLibraryAndExitThread(m_lib, 0);
00170                 #elif defined(SOUND4_CALL_FREEELIBRARY)
FreeLibrary(m_lib);
00171                 #endif // SOUND4_CALL_FREEELIBRARY
00172                 m_lib=NULL;
00173             }
00174         }
00175         }
00176         template <typename T>
00177         DynFuncHolder<T> GetSymbol(const std::string& name) {
00178             auto ptr=GetProcAddress(m_lib, name.c_str());
00179             if (!ptr) {
00180                 throw std::runtime_error("Missing function in library");
00181             }
00182             return DynFuncHolder<T>(ptr);
00183         }
00184         template <typename T>
00185         DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00186             auto ptr=GetProcAddress(m_lib, name.c_str());
00187             return DynFuncHolder<T>(ptr);
00188         }
00189         static std::filesystem::path GetThisLibraryPath(void) {
00190             static wchar_t path[MAX_PATH]={};
00191             HMODULE hm = NULL;
00192             if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00193

```

```

00195         (LPCWSTR) &GetThisLibraryPath, &hm))
00196     {
00197         if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00198             return std::filesystem::path(path);
00199         }
00200     }
00201     return {};
00202 }
00203 #endif
00204 virtual ~CDynLoader() {
00205     Close();
00206 }
00207 }; // class CDynLoader
00208 }; // namespace helper
00209
00210 class CDynLib {
00211 private:
00212     helper::CDynLoader m_lib;
00213     bool m_bOK = false;
00214 public:
00215     CDynLib() = default;
00216
00217     // All those wrapped functions have the same signature as there C impact_XXX
00218     helper::DynFuncHolder< decltype(impact_GetVersion) > > GetVersion ;
00219     helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) > > GetChunkSizeInFrames ;
00220     helper::DynFuncHolder< decltype(impact_GetChannelCount) > > GetChannelCount ;
00221     helper::DynFuncHolder< decltype(impact_GetAudioInputCount) > > GetAudioInputCount ;
00222     helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) > > GetAudioOutputCount ;
00223     helper::DynFuncHolder< decltype(impact_GetSampleRate) > > GetSampleRate ;
00224     helper::DynFuncHolder< decltype(impact_SetLoggerCallback) > > SetLoggerCallback ;
00225     helper::DynFuncHolder< decltype(impact_SetLogSeverity) > > SetLogSeverity ;
00226     helper::DynFuncHolder< decltype(impact_NewParameters) > > NewParameters ;
00227     helper::DynFuncHolder< decltype(impact_FreeParameters) > > FreeParameters ;
00228     helper::DynFuncHolder< decltype(impact_SetParameter) > > SetParameter ;
00229     helper::DynFuncHolder< decltype(impact_GetParameter) > > GetParameter ;
00230     helper::DynFuncHolder< decltype(impact_FreeParameterValue) > > FreeParameterValue ;
00231     helper::DynFuncHolder< decltype(impact_InitProcess) > > InitProcess ;
00232     helper::DynFuncHolder< decltype(impact_InitProcess2) > > InitProcess2 ;
00233     helper::DynFuncHolder< decltype(impact_TerminateProcess) > > TerminateProcess ;
00234     helper::DynFuncHolder< decltype(impact_ExitProcess) > > ExitProcess ;
00235     helper::DynFuncHolder< decltype(impact_StartUpdateThread) > > StartUpdateThread ;
00236     helper::DynFuncHolder< decltype(impact_StopUpdateThread) > > StopUpdateThread ;
00237     helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) > > WaitUpdateThreadReady ;
00238     helper::DynFuncHolder< decltype(impact_ProcessAudio) > > ProcessAudio ;
00239     helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) > > ProcessAudio_Planar ;
00240     helper::DynFuncHolder< decltype(impact_GetBufferIn) > > GetBufferIn ;
00241     helper::DynFuncHolder< decltype(impact_GetBufferOut) > > GetBufferOut ;
00242     helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) > > GetEstimatedDelay ;
00243     helper::DynFuncHolder< decltype(impact_GetFormatName) > > GetFormatName ;
00244     helper::DynFuncHolder< decltype(impact_GetFormatFromName) > > GetFormatFromName ;
00245     helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) > > GetBytesFromFormat ;
00246     helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) > > GetMaxPacketFrame ;
00247     helper::DynFuncHolder< decltype(impact_AddAudio) > > AddAudio ;
00248     helper::DynFuncHolder< decltype(impact_AddPadAudio) > > AddPadAudio ;
00249     helper::DynFuncHolder< decltype(impact_GetOutputCount) > > GetOutputCount ;
00250     helper::DynFuncHolder< decltype(impact_GetAudio) > > GetAudio ;
00251     helper::DynFuncHolder< decltype(impact_AudioConvertFrom) > > AudioConvertFrom ;
00252     helper::DynFuncHolder< decltype(impact_AudioConvertTo) > > AudioConvertTo ;
00253     helper::DynFuncHolder< decltype(impact_StereoToMono) > > StereoToMono ;
00254     helper::DynFuncHolder< decltype(impact_MonoToStereo) > > MonoToStereo ;
00255     helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > > AudioMonoFromLiveStereo ;
00256     helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) > > AudioMonoToLiveStereo ;
00257     helper::DynFuncHolder< decltype(impact_NewClient) > > NewClient ;
00258     helper::DynFuncHolder< decltype(impact_DeleteClient) > > DeleteClient ;
00259     helper::DynFuncHolder< decltype(impact_ProcessJson) > > ProcessJson ;
00260     helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) > > FreeJsonAnswer ;
00261     helper::DynFuncHolder< decltype(impact_SaveState) > > SaveState ;
00262
00263 #if IMPACT_HAS_WEBSERVER
00264     helper::DynFuncHolder< decltype(impact_Webserver_tcp) > > Webserver_tcp ;
00265     helper::DynFuncHolder< decltype(impact_Webserver_tcp2) > > Webserver_tcp2 ;
00266     helper::DynFuncHolder< decltype(impact_Webserver) > > Webserver ;
00267     helper::DynFuncHolder< decltype(impact_Webserver_Stop) > > Webserver_Stop ;
00268     helper::DynFuncHolder< decltype(impact_Webserver_Status) > > Webserver_Status ;
00269 #endif // IMPACT_HAS_WEBSERVER
00270
00271 // added 2023-03-22
00272     helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) > > StereoToMono_Planar ;
00273     helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) > > MonoToStereo_Planar ;
00274
00275 // added 2023-05-22
00276     helper::DynFuncHolder< decltype(impact_SetMetadata) > > SetMetadata ;
00277     helper::DynFuncHolder< decltype(impact_GetMetadataInfos) > > GetMetadataInfos ;
00278     helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) > > FreeMetadataInfos ;
00279
00280 // added 2023-06-20
00281     helper::DynFuncHolder< decltype(impact_SetPresetManager) > > SetPresetManager ;

```

```

00287     helper::DynFuncHolder< decltype(impact_PresetManager_InformChange    ) >
PresetManager_InformChange    ;
00288
00289     // added 2023-09-21
00290     helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames    ) >
GetPossibleChunkSizeInFrames    ;
00291     helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames    ) >
GetProcessChunkFrames    ;
00292     helper::DynFuncHolder< decltype(impact_InitProcess3    ) > InitProcess3
;
00293
00294 #if IMPACT_HAS_CLOUDBUS
00295     // added 2023-06-26
00296     helper::DynFuncHolder< decltype(impact_NewBus    ) > NewBus    ;
00297     helper::DynFuncHolder< decltype(impact_FreeBus    ) > FreeBus    ;
00298     helper::DynFuncHolder< decltype(impact_SetInstanceBus    ) > SetInstanceBus    ;
00299 #endif // IMPACT_HAS_CLOUDBUS
00300
00301     // added 2024-02-19
00302     helper::DynFuncHolder< decltype(impact_SetMetadataMulti    ) > SetMetadataMulti    ;
00303
00304 #if IMPACT_HAS_WEBSERVER
00305     // added 2024-05-23
00306     helper::DynFuncHolder< decltype(impact_Webserver_SetAppHealth    ) > Webserver_SetAppHealth
;
00307     helper::DynFuncHolder< decltype(impact_Webserver_GetAppHealth    ) > Webserver_GetAppHealth
;
00308     helper::DynFuncHolder< decltype(impact_Webserver_FreeString    ) > Webserver_FreeString
;
00309 #endif // IMPACT_HAS_WEBSERVER
00310
00311 bool Load(const std::filesystem::path& path = {}) {
00312     std::string filename;
00313     #if defined(__APPLE__)
00314         filename="libsound4.impact.cl.dylib";
00315     #elif defined(__unix__)
00316         filename="libsound4.impact.cl.so";
00317     #else
00318         filename="sound4.impact.cl.dll";
00319     #endif
00320     m_lib.Close();
00321     // If a path is given, use it directly and do not try other path
00322     if (!path.empty() && !m_lib.Load(path / filename)) {
00323         return false;
00324     } else if (!m_lib.IsOk()) {
00325         auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00326         #ifndef _WIN32
00327             // Linux: if in a bin directory, try ../lib first
00328             if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00329                 auto libdir = thisdir.parent_path() / "lib";
00330                 m_lib.Load(libdir / filename);
00331             }
00332         #endif // !_WIN32
00333         if (!m_lib.IsOk()) {
00334             // Search in the same directory this code is in
00335             m_lib.Load(thisdir / filename);
00336         }
00337         if (!m_lib.IsOk()) {
00338             // Try current path
00339             std::error_code ec;
00340             auto p = std::filesystem::current_path(ec);
00341             if (!ec) {
00342                 m_lib.Load(p / filename);
00343             }
00344         }
00345         if (!m_lib.IsOk()) {
00346             return false;
00347         }
00348     }
00349     // Load all C functions
00350     try {
00351         GetVersion                = m_lib.GetSymbol< decltype(impact_GetVersion    ) >
>("impact_GetVersion"    );
00352         GetChunkSizeInFrames      = m_lib.GetSymbol< decltype(impact_GetChunkSizeInFrames    ) >
>("impact_GetChunkSizeInFrames"    );
00353         GetChannelCount           = m_lib.GetSymbol< decltype(impact_GetChannelCount    ) >
>("impact_GetChannelCount"    );
00354         GetAudioInputCount        = m_lib.GetSymbol< decltype(impact_GetAudioInputCount    ) >
>("impact_GetAudioInputCount"    );
00355         GetAudioOutputCount       = m_lib.GetSymbol< decltype(impact_GetAudioOutputCount    ) >
>("impact_GetAudioOutputCount"    );
00356         GetSampleRate             = m_lib.GetSymbol< decltype(impact_GetSampleRate    ) >
>("impact_GetSampleRate"    );
00357         SetLoggerCallback          = m_lib.GetSymbol< decltype(impact_SetLoggerCallback    ) >
>("impact_SetLoggerCallback"    );
00358         SetLogSeverity            = m_lib.GetSymbol< decltype(impact_SetLogSeverity    ) >
>("impact_SetLogSeverity"    );

```

```

00366         NewParameters          = m_lib.GetSymbol< decltype(impact_NewParameters) )
>("impact_NewParameters" ) ;
00367         FreeParameters          = m_lib.GetSymbol< decltype(impact_FreeParameters) )
>("impact_FreeParameters" ) ;
00368         SetParameter           = m_lib.GetSymbol< decltype(impact_SetParameter) )
>("impact_SetParameter" ) ;
00369         GetParameter           = m_lib.GetSymbol< decltype(impact_GetParameter) )
>("impact_GetParameter" ) ;
00370         FreeParameterValue      = m_lib.GetSymbol< decltype(impact_FreeParameterValue) )
>("impact_FreeParameterValue" ) ;
00371         InitProcess            = m_lib.GetSymbol< decltype(impact_InitProcess) )
>("impact_InitProcess" ) ;
00372         InitProcess2           = m_lib.GetSymbol< decltype(impact_InitProcess2) )
>("impact_InitProcess2" ) ;
00373         TerminateProcess       = m_lib.GetSymbol< decltype(impact_TerminateProcess) )
>("impact_TerminateProcess" ) ;
00374         ExitProcess            = m_lib.GetSymbol< decltype(impact_ExitProcess) )
>("impact_ExitProcess" ) ;
00375         StartUpdateThread      = m_lib.GetSymbol< decltype(impact_StartUpdateThread) )
>("impact_StartUpdateThread" ) ;
00376         StopUpdateThread       = m_lib.GetSymbol< decltype(impact_StopUpdateThread) )
>("impact_StopUpdateThread" ) ;
00377         WaitUpdateThreadReady  = m_lib.GetSymbol< decltype(impact_WaitUpdateThreadReady) )
>("impact_WaitUpdateThreadReady" ) ;
00378         ProcessAudio           = m_lib.GetSymbol< decltype(impact_ProcessAudio) )
>("impact_ProcessAudio" ) ;
00379         ProcessAudio_Planar    = m_lib.GetSymbol< decltype(impact_ProcessAudio_Planar) )
>("impact_ProcessAudio_Planar" ) ;
00380         GetBufferIn            = m_lib.GetSymbol< decltype(impact_GetBufferIn) )
>("impact_GetBufferIn" ) ;
00381         GetBufferOut           = m_lib.GetSymbol< decltype(impact_GetBufferOut) )
>("impact_GetBufferOut" ) ;
00382         GetEstimatedDelay      = m_lib.GetSymbol< decltype(impact_GetEstimatedDelay) )
>("impact_GetEstimatedDelay" ) ;
00383         GetFormatName          = m_lib.GetSymbol< decltype(impact_GetFormatName) )
>("impact_GetFormatName" ) ;
00384         GetFormatFromName      = m_lib.GetSymbol< decltype(impact_GetFormatFromName) )
>("impact_GetFormatFromName" ) ;
00385         GetBytesFromFormat     = m_lib.GetSymbol< decltype(impact_GetBytesFromFormat) )
>("impact_GetBytesFromFormat" ) ;
00386         GetMaxPacketFrame     = m_lib.GetSymbol< decltype(impact_GetMaxPacketFrame) )
>("impact_GetMaxPacketFrame" ) ;
00387         AddAudio               = m_lib.GetSymbol< decltype(impact_AddAudio) )
>("impact_AddAudio" ) ;
00388         AddPadAudio            = m_lib.GetSymbol< decltype(impact_AddPadAudio) )
>("impact_AddPadAudio" ) ;
00389         GetOutputCount         = m_lib.GetSymbol< decltype(impact_GetOutputCount) )
>("impact_GetOutputCount" ) ;
00390         GetAudio               = m_lib.GetSymbol< decltype(impact_GetAudio) )
>("impact_GetAudio" ) ;
00391         AudioConvertFrom       = m_lib.GetSymbol< decltype(impact_AudioConvertFrom) )
>("impact_AudioConvertFrom" ) ;
00392         AudioConvertTo         = m_lib.GetSymbol< decltype(impact_AudioConvertTo) )
>("impact_AudioConvertTo" ) ;
00393         StereoToMono           = m_lib.GetSymbol< decltype(impact_StereoToMono) )
>("impact_StereoToMono" ) ;
00394         MonoToStereo           = m_lib.GetSymbol< decltype(impact_MonoToStereo) )
>("impact_MonoToStereo" ) ;
00395         AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoFromLiveStereo) )
>("impact_AudioMonoFromLiveStereo") ;
00396         AudioMonoToLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoToLiveStereo) )
>("impact_AudioMonoToLiveStereo" ) ;
00397         NewClient              = m_lib.GetSymbol< decltype(impact_NewClient) )
>("impact_NewClient" ) ;
00398         DeleteClient           = m_lib.GetSymbol< decltype(impact_DeleteClient) )
>("impact_DeleteClient" ) ;
00399         ProcessJson            = m_lib.GetSymbol< decltype(impact_ProcessJson) )
>("impact_ProcessJson" ) ;
00400         FreeJsonAnswer         = m_lib.GetSymbol< decltype(impact_FreeJsonAnswer) )
>("impact_FreeJsonAnswer" ) ;
00401         SaveState              = m_lib.GetSymbol< decltype(impact_SaveState) )
>("impact_SaveState" ) ;
00402 #if IMPACT_HAS_WEBSERVER
00403         Webserver_tcp          = m_lib.GetSymbol< decltype(impact_Webserver_tcp) )
>("impact_Webserver_tcp" ) ;
00404         Webserver_tcp2         = m_lib.GetSymbol< decltype(impact_Webserver_tcp2) )
>("impact_Webserver_tcp2" ) ;
00405         Webserver              = m_lib.GetSymbol< decltype(impact_Webserver) )
>("impact_Webserver" ) ;
00406         Webserver_Stop         = m_lib.GetSymbol< decltype(impact_Webserver_Stop) )
>("impact_Webserver_Stop" ) ;
00407         Webserver_Status       = m_lib.GetSymbol< decltype(impact_Webserver_Status) )
>("impact_Webserver_Status" ) ;
00408 #endif // IMPACT_HAS_WEBSERVER
00409
00410         // added 2023-03-22
00411         StereoToMono_Planar    = m_lib.GetSymbol< decltype(impact_StereoToMono_Planar) )

```

```

>("impact_StereoToMono_Planar" );
00412     MonoToStereo_Planar = m_lib.GetSymbol< decltype(impact_MonoToStereo_Planar )
>("impact_MonoToStereo_Planar" );
00413
00414     } catch (std::runtime_error& ) {
00415         return false;
00416     }
00417     // C functions allowed to be missing
00418     try {
00419         // added 2023-05-22
00420         SetMetadata = m_lib.GetSymbol< decltype(impact_SetMetadata )
>("impact_SetMetadata" );
00421         GetMetadataInfos = m_lib.GetSymbol< decltype(impact_GetMetadataInfos )
>("impact_GetMetadataInfos" );
00422         FreeMetadataInfos = m_lib.GetSymbol< decltype(impact_FreeMetadataInfos )
>("impact_FreeMetadataInfos" );
00423         // added 2023-06-20
00424         SetPresetManager = m_lib.GetSymbol< decltype(impact_SetPresetManager
>("impact_SetPresetManager" );
00425         PresetManager_InformChange = m_lib.GetSymbol<
decltype(impact_PresetManager_InformChange ) >("impact_PresetManager_InformChange" );
00426
00427         // added 2023-09-21
00428         GetPossibleChunkSizeInFrames = m_lib.GetSymbol<
decltype(impact_GetPossibleChunkSizeInFrames ) >("impact_GetPossibleChunkSizeInFrames" );
00429         GetProcessChunkFrames = m_lib.GetSymbol< decltype(impact_GetProcessChunkFrames
>("impact_GetProcessChunkFrames" );
00430         InitProcess3 = m_lib.GetSymbol< decltype(impact_InitProcess3
>("impact_InitProcess3" );
00431
00432 #if IMPACT_HAS_CLOUDBUS
00433     // added 2023-06-26
00434     NewBus = m_lib.GetSymbol< decltype(impact_NewBus )
>("impact_NewBus" );
00435     FreeBus = m_lib.GetSymbol< decltype(impact_FreeBus )
>("impact_FreeBus" );
00436     SetInstanceBus = m_lib.GetSymbol< decltype(impact_SetInstanceBus )
>("impact_SetInstanceBus" );
00437 #endif // IMPACT_HAS_CLOUDBUS
00438
00439     // added 2024-02-19
00440     SetMetadataMulti = m_lib.GetSymbol< decltype(impact_SetMetadataMulti )
>("impact_SetMetadataMulti" );
00441
00442 #if IMPACT_HAS_WEBSERVER
00443     // added 2024-05-23
00444     Webserver_SetAppHealth = m_lib.GetSymbol< decltype(impact_Webserver_SetAppHealth
>("Webserver_SetAppHealth" );
00445     Webserver_GetAppHealth = m_lib.GetSymbol< decltype(impact_Webserver_GetAppHealth
>("Webserver_GetAppHealth" );
00446     Webserver_FreeString = m_lib.GetSymbol< decltype(impact_Webserver_FreeString
>("Webserver_FreeString" );
00447 #endif // IMPACT_HAS_WEBSERVER
00448
00449     } catch (std::runtime_error& ) {
00450         // Ignored, handler will take care
00451     }
00452
00453     m_bOK=true;
00454     return true;
00455 }
00462 bool IsOk() const {
00463     return m_bOK;
00464 }
00465
00466 };
00467
00468
00469
00470 static constexpr const char *process_name = "SOUND4 IMPACT.CL";
00471 static constexpr const char *process_shortname = "impact";
00472 using SampleFormat = impact_SampleFormat;
00476 const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
00480 const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
00484 const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
00488 const size_t InputSampleSize=ChunkFrames*InputChannels;
00492 const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00496 const size_t SampleRate=IMPACT_SAMPLE_RATE;
00497
00501 namespace helper {
00502     #ifdef _WIN32
00503     static inline std::string WStringToUTF8(const std::wstring& wstr) {
00504         if (wstr.empty()) return {};
00505         int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
00511         NULL);
00511         std::string str(size, 0);
00512         WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);

```

```

00513         return str;
00514     }
00520     static inline std::wstring UTF8ToWString(const std::string& str) {
00521         if (str.empty()) return std::wstring();
00522         int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00523         std::wstring wstr(size, 0);
00524         MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00525         return wstr;
00526     }
00527 #endif // _WIN32
00528
00534     template<typename T>
00535     struct SampleFormat {
00536     };
00537     template<>
00538     struct SampleFormat<int16_t> {
00539         const impact_SampleFormat format = S16_NATIVE;
00540     };
00541     template<>
00542     struct SampleFormat<int32_t> {
00543         const impact_SampleFormat format = S32_NATIVE;
00544     };
00545     template<>
00546     struct SampleFormat<float> {
00547         const impact_SampleFormat format = F32_NATIVE;
00548     };
00549
00550
00555     static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output,
size_t nSpl, impact_SampleFormat fmt) {
00556         dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
00557     }
00558     static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload,
size_t nSpl, impact_SampleFormat fmt) {
00559         dynlib.AudioConvertTo(input, payload, nSpl, fmt);
00560     }
00561     static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t
nSpl) {
00562         dynlib.StereoToMono(input, output, nSpl);
00563     }
00564     static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t
nSpl) {
00565         dynlib.MonoToStereo(input, output, nSpl);
00566     }
00567     static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float
*inputR, float *output, size_t nSpl) {
00568         dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
00569     }
00570     static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL,
float *outputR, size_t nSpl) {
00571         dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
00572     }
00573     static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float
*output) {
00574         dynlib.AudioMonoFromLiveStereo(payload, output);
00575     }
00576     static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t
*payload) {
00577         dynlib.AudioMonoToLiveStereo(input, payload);
00578     }
00579
00580     }; // namespace helper
00581
00587     static inline std::string GetVersion(CDynLib& dynlib)
00588     {
00589         return std::string(dynlib.GetVersion());
00590     }
00591
00602     static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
00603     {
00604         return dynlib.GetChunkSizeInFrames();
00605     }
00606
00615     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
00616     {
00617         std::vector<unsigned int> list;
00618         if (dynlib.GetPossibleChunkSizeInFrames) {
00619             for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
00620                 list.push_back(*src);
00621             }
00622         } else {
00623             list.push_back(64);
00624         }
00625         return list;
00626     }
00627
00635     static inline unsigned int GetChannelCount(CDynLib& dynlib)

```

```

00636     {
00637         return dynlib.GetChannelCount();
00638     }
00639
00640     static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
00641     {
00642         return dynlib.GetAudioInputCount();
00643     }
00644
00645     static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
00646     {
00647         return dynlib.GetAudioOutputCount();
00648     }
00649
00650     static inline unsigned int GetSampleRate(CDynLib& dynlib)
00651     {
00652         return dynlib.GetSampleRate();
00653     }
00654
00655     static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
00656     {
00657         // Do some sanity checks
00658         if (dynlib.GetSampleRate() != IMPACT_SAMPLE_RATE) {
00659             throw std::runtime_error("Bad library sampling rate");
00660         }
00661         if (a_bCheckFrames && dynlib.GetChunkSizeInFrames() != ChunkFrames) {
00662             throw std::runtime_error("Bad library frame size");
00663         }
00664         if (dynlib.GetChannelCount() != IMPACT_CHANNEL_COUNT) {
00665             throw std::runtime_error("Bad library channel count");
00666         }
00667         if (dynlib.GetAudioInputCount() != IMPACT_AUDIO_INPUT_COUNT) {
00668             throw std::runtime_error("Bad library input count");
00669         }
00670         if (dynlib.GetAudioOutputCount() != IMPACT_AUDIO_OUTPUT_COUNT) {
00671             throw std::runtime_error("Bad library output count");
00672         }
00673     }
00674
00675     static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
00676     {
00677         return std::string(dynlib.GetFormatName(fmt));
00678     }
00679
00680     static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
00681     {
00682         return dynlib.GetFormatFromName(name.c_str());
00683     }
00684
00685     static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
00686     {
00687         return dynlib.GetBytesFromFormat(fmt);
00688     }
00689
00690     using LogSeverity = impact_LogSeverity;
00691     using log_cb_t = std::function<void(LogSeverity, const std::string&);>;
00692
00693     static log_cb_t _log_cb;
00694     extern "C" {
00695         static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
00696             _log_cb(severity, std::string(c_msg));
00697         }
00698     }
00699
00700     static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
00701     {
00702         dynlib.SetLogSeverity(severity);
00703     }
00704
00705     static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
00706     {
00707         _log_cb=cb;
00708         dynlib.SetLoggerCallback(&_log_cb_c);
00709     }
00710
00711     #if IMPACT_HAS_CLOUDBUS
00712     class CBus {
00713     public:
00714         CBus(CDynLib& dynlib)
00715             : m_dynlib(dynlib)
00716             {
00717             if (m_dynlib.NewBus) {
00718                 m_bus=m_dynlib.NewBus();
00719             }
00720         }
00721         ~CBus()
00722         {
00723             if (m_bus && m_dynlib.FreeBus) {
00724                 m_dynlib.FreeBus(m_bus);
00725             }
00726         }
00727     };
00728     #endif

```

```

00784         }
00785         m_bus = nullptr;
00786     }
00787     impact_CBus *Get() const { return m_bus; }
00788 private:
00789     CDynLib& m_dynlib;
00790     impact_CBus *m_bus;
00791 };
00792 #endif // IMPACT_HAS_CLOUDBUS
00793
00794 class CPresetLoader {
00795 public:
00800     CPresetLoader() = default;
00801     virtual ~CPresetLoader() = default;
00802
00803     virtual bool IsReadOnly() = 0;
00804     virtual bool Exists(const std::filesystem::path &name) = 0;
00805     virtual bool Remove(const std::filesystem::path &name) = 0;
00806     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00807     virtual std::vector<std::filesystem::path> GetAll() = 0;
00808     virtual std::string Read(const std::filesystem::path &filename) = 0;
00809     virtual bool Write(const std::filesystem::path &filename, const std::string &content) = 0;
00810 };
00811 using PresetChange_Kind = impact_PresetChange_Kind;
00812
00813 class CInstance {
00814 public:
00821     CInstance(CDynLib& dynlib)
00822     : m_dynlib(dynlib)
00823     {
00824         params = m_dynlib.NewParameters();
00825         if (!params) throw std::bad_alloc();
00826     }
00827     // non-copyable
00828     CInstance(const CInstance& ) = delete;
00829     CInstance& operator=( const CInstance& ) = delete;
00833     virtual ~CInstance()
00834     {
00835         Stop();
00836         if (params) m_dynlib.FreeParameters(params);
00837     }
00841     bool IsOk() const { return instance!=nullptr; }
00849     void SetParam(const std::string& name, const std::string& value)
00850     {
00851         assert(params);
00852         m_dynlib.SetParameter(params, name.c_str(), value.c_str());
00853     }
00859     std::string GetParam(const std::string& name)
00860     {
00861         assert(params);
00862         auto c_value=m_dynlib.GetParameter(params, name.c_str());
00863         std::string ret(c_value);
00864         m_dynlib.FreeParameterValue(c_value);
00865         return ret;
00866     }
00867 #if IMPACT_HAS_CLOUDBUS
00871     void SetBus(const CBus& bus) {
00872         assert(params);
00873         if (m_dynlib.SetInstanceBus) {
00874             m_dynlib.SetInstanceBus(params,bus.Get());
00875         }
00876     }
00877 #endif // IMPACT_HAS_CLOUDBUS
00878
00888     void SetPresetManager(CPresetLoader *preset_manager)
00889     {
00890         if (!preset_manager) return;
00891         if (!m_dynlib.SetPresetManager) return;
00892         assert(params);
00893         m_dynlib.SetPresetManager(params,
00894             impact_custom_reader,
00895             impact_custom_reader_free,
00896             impact_custom_writer,
00897             impact_custom_exists,
00898             impact_custom_getall,
00899             impact_custom_getall_free,
00900             impact_custom_remove,
00901             impact_custom_rename,
00902             preset_manager->IsReadOnly(),
00903             preset_manager
00904         );
00905     }
00906
00921     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)
00922     {

```

```

00923         assert(!instance);
00924         std::string l_save_path;
00925         if (!save_path.empty()) {
00926             l_save_path = save_path.u8string();
00927         }
00928         if (m_dynlib.InitProcess3) {
00929             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);
00930         } else if (frames_per_chunk!=64) {
00931             return false;
00932         } else {
00933             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);
00934         }
00935         if (!instance) return false;
00936         if (!init_metadata.empty()) {
00937             for (auto&& [key,value]: init_metadata) {
00938                 m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
00939             }
00940             init_metadata.clear();
00941         }
00942         update_thread = std::thread([this, json_port]() {
00943             OnUpdateThreadStart();
00944             m_dynlib.StartUpdateThread(instance, json_port);
00945             OnUpdateThreadStop();
00946         });
00947         int timeout=5000;
00948         #ifdef DEBUG
00949             timeout=60*1000; // For debugger pauses
00950         #endif
00951         if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
00952             return false;
00953         }
00954         return true;
00955     }
00956
00964     unsigned int GetChunkFrames() {
00965         if (instance && m_dynlib.GetProcessChunkFrames) {
00966             return m_dynlib.GetProcessChunkFrames(instance);
00967         }
00968         return 64;
00969     }
00970
00978     void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00979     {
00980         if (m_dynlib.PresetManager_InformChange) {
00981             m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(),
(impact_PresetChange_Kind) change_kind);
00982         }
00983     }
00984
00996     void SetMetadata(const std::string& key, const char* value)
00997     {
00998         if (m_dynlib.SetMetadata) {
00999             if (instance) {
01000                 m_dynlib.SetMetadata(instance, key.c_str(), value);
01001             } else {
01002                 init_metadata.push_back( {key, value} );
01003             }
01004         }
01005     }
01006
01017     void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
01018     {
01019         if (instance) {
01020             if (m_dynlib.SetMetadataMulti) {
01021                 typedef const char* pchar;
01022                 pchar* keyvalue=new pchar[2*list.size()+1];
01023                 size_t n=0;
01024                 for (auto&& [key,value]: list) {
01025                     keyvalue[2*n+0]=key.c_str();
01026                     keyvalue[2*n+1]=value;
01027                     n++;
01028                 }
01029                 keyvalue[2*n+0]=nullptr;
01030
01031                 impact_SetMetadataMulti(instance, keyvalue);
01032
01033                 delete[] keyvalue;
01034             } else {
01035                 for (auto&& [key,value]: list) {
01036                     SetMetadata(key, value);
01037                 }
01038             }
01039         } else {

```

```

01040         for (auto&& [key,value]: list) {
01041             if (value) {
01042                 init_metadata.push_back( {key, value} );
01043             }
01044         }
01045     }
01046 }
01047
01053 std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
01054 {
01055     std::vector< std::tuple<std::string,std::string> > values;
01056     if (m_dynlib.GetMetadataInfos) {
01057         const char** c_values = m_dynlib.GetMetadataInfos(instance);
01058         if (c_values) {
01059             for (const char** c_value=c_values; *c_value; ) {
01060                 std::string key(*c_value);
01061                 c_value++;
01062                 if (*c_value) {
01063                     std::string descr(*c_value);
01064                     c_value++;
01065                     values.push_back( {key,descr} );
01066                 }
01067             }
01068             m_dynlib.FreeMetadataInfos(instance, c_values);
01069         }
01070     }
01071     return values;
01072 }
01073
01074 // Can be overloaded, called at start of UpdateThread in the thread
01075 virtual void OnUpdateThreadStart() {}
01076 // Can be overloaded to call at end of UpdateThread in the thread
01077 virtual void OnUpdateThreadStop() {}
01078
01079 #if IMPACT_HAS_WEBSERVER
01085 bool StartWebServer(int http_port, int https_port=0)
01086 {
01087     assert(instance);
01088     assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01089     webserver = m_dynlib.Webserver(http_port,https_port,instance);
01090     if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01091         return false;
01092     }
01093     int web_status=m_dynlib.Webserver_Status(webserver);
01094     if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01095         return false;
01096     }
01097     if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01098         return false;
01099     }
01100     return true;
01101 }
01107 void StopWebServer()
01108 {
01109     if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01110         m_dynlib.Webserver_Stop(webserver, 1000);
01111         webserver = SOUND4_INVALID_WEBSERVER_ID;
01112     }
01113 }
01114
01118 void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
content)
01119 {
01120     assert(instance);
01121     if (!m_dynlib.Webserver_SetAppHealth) return;
01122     m_dynlib.Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
content.c_str());
01123 }
01127 void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01128 {
01129     assert(instance);
01130     if (!m_dynlib.Webserver_GetAppHealth) return;
01131     char* c_contenttype=nullptr;
01132     char* c_content=nullptr;
01133     m_dynlib.Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01134     contenttype=c_contenttype;
01135     content=c_content;
01136     m_dynlib.Webserver_FreeString(c_contenttype);
01137     m_dynlib.Webserver_FreeString(c_content);
01138 }
01139 #endif // IMPACT_HAS_WEBSERVER
01140
01147 int Terminate()
01148 {
01149     if (!instance) return 0;
01150     return m_dynlib.TerminateProcess(instance);
01151 }

```

```

01152
01158     void Stop()
01159     {
01160         if (!instance) return;
01161         #if IMPACT_HAS_WEBSERVER
01162             StopWebServer();
01163         #endif // IMPACT_HAS_WEBSERVER
01164         if (update_thread.joinable()) {
01165             m_dynlib.StopUpdateThread(instance); // this will exit the update thread
01166             update_thread.join();
01167         }
01168         m_dynlib.ExitProcess(instance);
01169         instance=nullptr;
01170     }
01171
01177     unsigned int GetEstimatedDelay()
01178     {
01179         assert(instance);
01180         return m_dynlib.GetEstimatedDelay(instance);
01181     }
01182
01193     std::array<float,InputSampleSize>& GetBufferIn()
01194     {
01195         assert(instance);
01196         float *buf=m_dynlib.GetBufferIn(instance);
01197         return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
01198     }
01199
01210     std::array<float,OutputSampleSize>& GetBufferOut()
01211     {
01212         assert(instance);
01213         float *buf= m_dynlib.GetBufferOut(instance);
01214         return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
01215     }
01221     void ProcessAudio()
01222     {
01223         assert(instance);
01224         m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance),
m_dynlib.GetBufferOut(instance));
01225     }
01239     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01240     void ProcessAudio_Planar(float const * const *input, float * const *output)
01241     {
01242         assert(instance);
01243         m_dynlib.ProcessAudio_Planar(instance, input, output);
01244     }
01245
01257     template<typename T>
01258     std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01259     {
01260         assert(instance);
01261         std::vector<T> output;
01262         unsigned int out_offset=0;
01263         unsigned int in_offset=0;
01264         unsigned int todo = input.size();
01265         while (todo>0) {
01266             unsigned int left = AddAudio(&input[in_offset], todo);
01267             unsigned int out_avail = m_dynlib.GetOutputCount(instance);
01268             output.resize(out_offset + out_avail);
01269             GetAudio(&output[out_offset], out_avail);
01270             out_offset+=out_avail;
01271             in_offset += todo-left;
01272             todo=left;
01273         }
01274         return output;
01275     }
01276
01282     class CClient {
01283     public:
01284         CClient(impact_CInstance* instance, CDynLib& dynlib)
01285         : m_dynlib(dynlib)
01286         {
01287             assert(instance);
01288             client=m_dynlib.NewClient(instance);
01289             if (!client) throw std::bad_alloc();
01290         }
01291         // non-copyable
01292         CClient( const CInstance& ) = delete;
01293         CClient& operator=( const CInstance& ) = delete;
01294         ~CClient()
01295         {
01296             if (client) {
01297                 m_dynlib.DeleteClient(client);
01298                 client=nullptr;
01299             }
01300         }

```

```

01308         std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01309         {
01310             assert(client);
01311             int need_save=0;
01312             const char *canswer = m_dynlib.ProcessJson(client, request.c_str(),
&need_save);
01313             if (!canswer) return {};
01314             std::string answer(canswer);
01315             m_dynlib.FreeJsonAnswer (canswer);
01316             if (NeedSave) {
01317                 *NeedSave=(need_save!=0);
01318             }
01319             return answer;
01320         }
01321     private:
01322         CDynLib& m_dynlib;
01323         impact_CClientInstance *client = nullptr;
01324     };
01332     std::shared_ptr<CClient> NewClient()
01333     {
01334         assert(instance);
01335         return std::make_shared<CClient>(instance, m_dynlib);
01336     }
01337
01338     protected:
01339         template<typename T>
01340         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01341         {
01342             assert(instance);
01343             return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
helper::SampleFormat<T>::format);
01344         }
01345         template<typename T>
01346         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01347         {
01348             assert(instance);
01349             return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
helper::SampleFormat<T>::format);
01350         }
01351
01352     private:
01353         CDynLib& m_dynlib;
01354         impact_CParameters* params = nullptr;
01355         impact_CInstance* instance = nullptr;
01356         std::thread update_thread;
01357         std::vector< std::pair<std::string, std::string> > init_metadata;
01358         #if IMPACT_HAS_WEBSERVER
01359         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01360         #endif // IMPACT_HAS_WEBSERVER
01361     };
01362
01363 }; // namespace dyn
01364 }; // namespace impact
01365 }; // namespace sound4
01366
01367 // bridge C callbacks to CPresetLoader
01368 extern "C" {
01369     static char *impact_custom_reader(const fs_char *filename, void* userdata) {
01370         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01372         std::string content=preset_loader->Read(filename);
01373         return strdup(content.c_str());
01374     }
01375     static void impact_custom_reader_free(char *content, void* userdata) {
01376         if (!content) return;
01377         free(content);
01378     }
01379     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01380         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01382         auto res=preset_loader->Write(filename,content);
01383         return res?0:-1;
01384     }
01385     static int impact_custom_exists(const fs_char *filename, void* userdata) {
01386         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01388         auto res=preset_loader->Exists(filename);
01389         return res?0:-1;
01390     }
01391     static fs_char** impact_custom_getall(void* userdata) {
01392         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01394         auto res=preset_loader->GetAll();
01395         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01396         for (size_t n=0;n<res.size();n++) {
01397             all[n]=fs_strdup(res[n].c_str());
01398         }
01399         all[n]=nullptr;
01400     }

```

```

01395     }
01396     all[res.size()]=nullptr;
01397     return all;
01398 }
01399 static void impact_custom_getall_free(fs_char** all, void* userdata) {
01400     if (!all) return;
01401     for (fs_char** one=all;*one;one++) {
01402         free(*one);
01403     }
01404     free(all);
01405 }
01406 static int impact_custom_remove(const fs_char *filename, void* userdata) {
01407     sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01408     auto res=preset_loader->Remove(filename);
01409     return res?0:-1;
01410 }
01411 static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01412     sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01413     auto res=preset_loader->Rename(from,to);
01414     return res?0:-1;
01415 }
01416
01417 };

```

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

- namespace [sound4](#)
- namespace [sound4::helper](#)

Macros

- `#define` [UNIXLIKE](#) 0

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](#).

11.15.2 Macro Definition Documentation

11.15.2.1 UNIXLIKE

```
#define UNIXLIKE 0
```

Definition at line 24 of file [sound4cl.hpp](#).

11.16 sound4cl.hpp

[Go to the documentation of this file.](#)

```

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 #if defined(__unix__) || defined(__APPLE__)
00022     #define UNIXLIKE 1
00023 #else
00024     #define UNIXLIKE 0
00025 #endif
00026
00027 #if UNIXLIKE
00028     #include <dlfcn.h>
00029 #elif defined(_WIN32)
00030     #include <libloaderapi.h>
00031 #else
00032     #error "Unsupported OS"
00033 #endif
00034
00035 #include "sound4cl_cdef.h"
00036
00037 // bridge C callbacks to CPresetLoader
00038 extern "C" {
00039     static char *sound4cl_custom_reader(const fs_char *filename, void* userdata);
00040     static void sound4cl_custom_reader_free(char *content, void* userdata);
00041     static int sound4cl_custom_writer(const fs_char *filename, const char *content, void*
userdata);
00042     static int sound4cl_custom_exists(const fs_char *filename, void* userdata);
00043     static fs_char** sound4cl_custom_getall(void* userdata);
00044     static void sound4cl_custom_getall_free(fs_char** all, void* userdata);
00045     static int sound4cl_custom_remove(const fs_char *filename, void* userdata);
00046     static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00047 };
00048
00052 namespace sound4 {
00053
00057     namespace helper {
00058         #ifdef _WIN32
00064         static inline std::wstring WStringToUTF8(const std::wstring& wstr) {
00065             if (wstr.empty()) return {};
00066             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
NULL);
00067             std::wstring str(size, 0);
00068             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00069             return str;
00070         }
00076         static inline std::wstring UTF8ToWString(const std::string& str) {
00077             if (str.empty()) return std::wstring();
00078             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00079             std::wstring wstr(size, 0);
00080             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00081             return wstr;
00082         }
00083         #endif // _WIN32
00084
00090         template<typename T>
00091         struct SampleFormat {
00092         };
00093         template<>
00094         struct SampleFormat<int16_t> {
00095             const sound4cl_SampleFormat format = S16_NATIVE;
00096         };
00097         template<>
00098         struct SampleFormat<int32_t> {
00099             const sound4cl_SampleFormat format = S32_NATIVE;
00100         };
00101         template<>
00102         struct SampleFormat<float> {
00103             const sound4cl_SampleFormat format = F32_NATIVE;
00104         };
00105
00106         // Storage definition, depending on OS
00107         #if UNIXLIKE

```

```

00108         using DynLib_t = void *;
00109         using DynFunc_t = void *;
00110     #else
00111         using DynLib_t = HMODULE;
00112         using DynFunc_t = FARPROC;
00113     #endif
00114
00120         template <typename T>
00121     class DynFuncHolder {
00122     private:
00123         DynFunc_t m_ptr = NULL;
00124     public:
00125         DynFuncHolder() = default;
00126         DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00127         operator bool() const { return (m_ptr!=NULL); }
00128         bool IsOk() const { return m_ptr!=NULL; }
00129         operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00130     };
00131
00136     class CDynLoader {
00137     private:
00138         DynLib_t m_lib {};
00139     public:
00140         CDynLoader() = default;
00141         CDynLoader(CDynLoader&&) = default;
00142         // No copy allowed
00143         CDynLoader(const CDynLoader&) = delete;
00144         CDynLoader& operator=(CDynLoader const&) = delete;
00145         operator bool() const {
00146             return (m_lib!=NULL);
00147         }
00148         bool IsOk() const {
00149             return (m_lib!=NULL);
00150         }
00151         #if UNIXLIKE
00152         void Close() {
00153             if (IsOk()) {
00154                 #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00155                     dlclose(m_lib);
00156                 #endif // __SANITIZE_ADDRESS__
00157                 m_lib=NULL;
00158             }
00159         }
00160         bool Load(const std::filesystem::path& dynlib)
00161         {
00162             Close();
00163             m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00164             return IsOk();
00165         }
00166         template <typename T>
00167         DynFuncHolder<T> GetSymbol(const std::string& name) {
00168             auto ptr=dlsym(m_lib, name.c_str());
00169             if (!ptr) {
00170                 throw std::runtime_error("Missing function in library");
00171             }
00172             return DynFuncHolder<T>(ptr);
00173         }
00174         template <typename T>
00175         DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00176             auto ptr=dlsym(m_lib, name.c_str());
00177             return DynFuncHolder<T>(ptr);
00178         }
00179         static std::filesystem::path GetThisLibraryPath(void) {
00180             static Dl_info info;
00181             if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00182             auto rp=realpath(info.dli_fname, NULL);
00183             std::filesystem::path p(rp);
00184             free(rp);
00185             return p;
00186         }
00187     #else
00188         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_
00189         {
00190             Close();
00191             m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00192             return IsOk();
00193         }
00194         void Close() {
00195             if (IsOk()) {
00196                 // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
FreeLibraryAndExitThread but...
00197                 // So define SOUND4_CALL_FREEELIBRARY before including this file if you
really want to unload the library dynamically
00198                 #if defined(SOUND4_CALL_FREEELIBRARYANDEXITTHREAD)
FreeLibraryAndExitThread(m_lib, 0);
00199                 #elif defined(SOUND4_CALL_FREEELIBRARY)

```

```

00201         FreeLibrary(m_lib);
00202     #endif // SOUND4_CALL_FREELIBRARY
00203     m_lib=NULL;
00204 }
00205 }
00206 template <typename T>
00207 DynFuncHolder<T> GetSymbol(const std::string& name) {
00208     auto ptr=GetProcAddress(m_lib, name.c_str());
00209     if (!ptr) {
00210         throw std::runtime_error("Missing function in library");
00211     }
00212     return DynFuncHolder<T>(ptr);
00213 }
00214 template <typename T>
00215 DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00216     auto ptr=GetProcAddress(m_lib, name.c_str());
00217     return DynFuncHolder<T>(ptr);
00218 }
00219 static std::filesystem::path GetThisLibraryPath(void) {
00220     static wchar_t path[MAX_PATH]={};
00221     HMODULE hm = NULL;
00222     if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00223         GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00224         (LPCWSTR) &GetThisLibraryPath, &hm))
00225     {
00226         if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00227             return std::filesystem::path(path);
00228         }
00229     }
00230     return {};
00231 }
00232 #endif
00233 virtual ~CDynLoader() {
00234     Close();
00235 }
00236 }; // class CDynLoader
00237 }; // namespace helper
00238
00239 using LogSeverity = sound4cl_LogSeverity;
00240 const LogSeverity none = (LogSeverity) sound4cl_none;
00241 const LogSeverity fatal = (LogSeverity) sound4cl_fatal;
00242 const LogSeverity error = (LogSeverity) sound4cl_error;
00243 const LogSeverity warning = (LogSeverity) sound4cl_warning;
00244 const LogSeverity info = (LogSeverity) sound4cl_info;
00245 const LogSeverity verbose = (LogSeverity) sound4cl_verbose;
00246 const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2;
00247 const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3;
00248 const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4;
00249 const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5;
00250
00251 using log_cb_t = std::function<void(LogSeverity,const std::string&);>;
00252
00253 static log_cb_t _log_cb;
00254 extern "C" {
00255     static inline void _log_cb_c(LogSeverity severity, const char *c_msg) {
00256         _log_cb(severity, std::string(c_msg));
00257     }
00258 }
00259
00260 class CProcessor {
00261 private:
00262     helper::CDynLoader m_lib;
00263     bool m_bOK = false;
00264     std::string m_prefix; // The prefix of functions, depends on the library (cloud_, impact_...)
00265 protected:
00266     friend class CBus;
00267     friend class CInstance;
00268
00269     // All those wrapped functions have the same signature as there C sound4cl_XXX
00270     helper::DynFuncHolder< decltype(sound4cl_GetVersion) > > fnGetVersion ;
00271     helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > > fnGetChunkSizeInFrames ;
00272     helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > > fnGetChannelCount ;
00273     helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > > fnGetAudioInputCount ;
00274     helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > > fnGetAudioOutputCount ;
00275     helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > > fnGetSampleRate ;
00276     helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > > fnSetLoggerCallback ;
00277     helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > > fnSetLogSeverity ;
00278     helper::DynFuncHolder< decltype(sound4cl_NewParameters) > > fnNewParameters ;
00279     helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > > fnFreeParameters ;
00280     helper::DynFuncHolder< decltype(sound4cl_SetParameter) > > fnSetParameter ;
00281     helper::DynFuncHolder< decltype(sound4cl_GetParameter) > > fnGetParameter ;
00282     helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > > fnFreeParameterValue ;
00283     helper::DynFuncHolder< decltype(sound4cl_InitProcess) > > fnInitProcess ;
00284     helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > > fnInitProcess2 ;
00285     helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > > fnTerminateProcess ;
00286     helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > > fnExitProcess ;

```

```

00303     helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) ) > fnStartUpdateThread ;
00304     helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) ) > fnStopUpdateThread ;
00305     helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) ) > fnWaitUpdateThreadReady ;
00306     helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) ) > fnProcessAudio ;
00307     helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) ) > fnProcessAudio_Planar ;
00308     helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) ) > fnGetBufferIn ;
00309     helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) ) > fnGetBufferOut ;
00310     helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) ) > fnGetEstimatedDelay ;
00311     helper::DynFuncHolder< decltype(sound4cl_GetFormatName) ) > fnGetFormatName ;
00312     helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) ) > fnGetFormatFromName ;
00313     helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) ) > fnGetBytesFromFormat ;
00314     helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) ) > fnGetMaxPacketFrame ;
00315     helper::DynFuncHolder< decltype(sound4cl_AddAudio) ) > fnAddAudio ;
00316     helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) ) > fnAddPadAudio ;
00317     helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) ) > fnGetOutputCount ;
00318     helper::DynFuncHolder< decltype(sound4cl_GetAudio) ) > fnGetAudio ;
00319     helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) ) > fnAudioConvertFrom ;
00320     helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) ) > fnAudioConvertTo ;
00321     helper::DynFuncHolder< decltype(sound4cl_StereoToMono) ) > fnStereoToMono ;
00322     helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) ) > fnMonoToStereo ;
00323     helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) ) > fnAudioMonoFromLiveStereo ;
00324     helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) ) > fnAudioMonoToLiveStereo ;
00325     helper::DynFuncHolder< decltype(sound4cl_NewClient) ) > fnNewClient ;
00326     helper::DynFuncHolder< decltype(sound4cl_DeleteClient) ) > fnDeleteClient ;
00327     helper::DynFuncHolder< decltype(sound4cl_ProcessJson) ) > fnProcessJson ;
00328     helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) ) > fnFreeJsonAnswer ;
00329     helper::DynFuncHolder< decltype(sound4cl_SaveState) ) > fnSaveState ;
00330
00331     // only for processes supporting webserver
00332     helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) ) > fnWebserver_tcp ;
00333     helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) ) > fnWebserver_tcp2 ;
00334     helper::DynFuncHolder< decltype(sound4cl_Webserver) ) > fnWebserver ;
00335     helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) ) > fnWebserver_Stop ;
00336     helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) ) > fnWebserver_Status ;
00337
00338     // added 2023-03-22
00339     helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) ) > fnStereoToMono_Planar ;
00340     helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) ) > fnMonoToStereo_Planar ;
00341
00342     // added 2023-05-22
00343     helper::DynFuncHolder< decltype(sound4cl_SetMetadata) ) > fnSetMetadata ;
00344     helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) ) > fnGetMetadataInfos ;
00345     helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) ) > fnFreeMetadataInfos ;
00346
00347     // added 2023-06-20
00348     helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) ) >
00349     fnSetPresetManager ;
00350     helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) ) >
00351     fnPresetManager_InformChange ;
00352
00353     // added 2023-09-21
00354     helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) ) >
00355     fnGetPossibleChunkSizeInFrames ;
00356     helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) ) >
00357     fnGetProcessChunkFrames ;
00358     helper::DynFuncHolder< decltype(sound4cl_InitProcess3) ) > fnInitProcess3 ;
00359
00360     // added 2023-06-26, only for processes supporting bus
00361     helper::DynFuncHolder< decltype(sound4cl_NewBus) ) > fnNewBus ;
00362     helper::DynFuncHolder< decltype(sound4cl_FreeBus) ) > fnFreeBus ;
00363     helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) ) > fnSetInstanceBus ;
00364
00365     // added 2024-02-19
00366     helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) ) > fnSetMetadataMulti ;
00367
00368     // added 2024-05-23
00369     helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) ) >
00370     fnWebserver_SetAppHealth ;
00371     helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) ) >
00372     fnWebserver_GetAppHealth ;
00373     helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) ) > fnWebserver_FreeString ;
00374
00375     template <typename T>
00376     helper::DynFuncHolder<T> GetPrefixSymbol(const std::string& name) {
00377         std::string prefix_name = m_prefix+name;
00378         return m_lib.GetSymbol<T>(prefix_name);
00379     }
00380
00381     public:
00382     CProcessor() = default;
00383
00384     bool Load(const std::filesystem::path& filepath) {
00385         m_lib.Close();
00386         // If a path is given, use it directly and do not try other path

```

```

00388         if (filepath.has_parent_path() && !m_lib.Load(filepath)) {
00389             return false;
00390         } else if (!m_lib.IsOk()) {
00391             auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00392             auto filename = filepath.filename();
00393             #ifndef _WIN32
00394                 // Linux: if in a bin directory, try ../lib first
00395             if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00396                 auto libdir = thisdir.parent_path() / "lib";
00397                 m_lib.Load(libdir / filename);
00398             }
00399             #endif // !_WIN32
00400             if (!m_lib.IsOk()) {
00401                 // Search in the same directory this code is in
00402                 m_lib.Load(thisdir / filename);
00403             }
00404             if (!m_lib.IsOk()) {
00405                 // Try current path
00406                 std::error_code ec;
00407                 auto p = std::filesystem::current_path(ec);
00408                 if (!ec) {
00409                     m_lib.Load(p / filename);
00410                 }
00411             }
00412             if (!m_lib.IsOk()) {
00413                 return false;
00414             }
00415         }
00416         // Get information from the library itself
00417         try {
00418             auto GetInfo = m_lib.GetSymbol< decltype(sound4cl_SOUND4_GetProcessInfo )
>("SOUND4_GetProcessInfo" );
00419             auto info = GetInfo();
00420             m_prefix = info->prefix;
00421         } catch (std::runtime_error& ) {
00422             // Compatibility: search in known libraries without this function
00423             #ifdef _WIN32
00424                 std::string filename = helper::WStringToUTF8(filepath.stem());
00425             #else // !_WIN32
00426                 std::string filename = filepath.stem();
00427                 // remove lib prefix
00428                 filename=filename.substr(3);
00429             #endif // !_WIN32
00430             if (filename=="sound4.xl.cloud") {
00431                 m_prefix="cloudxl_";
00432             } else if (filename=="sound4.impact.cl") {
00433                 m_prefix="impact_";
00434             } else if (filename=="sound4.bigvoice.cl") {
00435                 m_prefix="bigvoice_";
00436             } else {
00437                 return false;
00438             }
00439         }
00440
00441         // Load all C functions
00442         try {
00443             fnGetVersion = GetPrefixSymbol< decltype(sound4cl_GetVersion
00444 ) >("GetVersion" );
00445             fnGetChunkSizeInFrames = GetPrefixSymbol< decltype(sound4cl_GetChunkSizeInFrames
00446 ) >("GetChunkSizeInFrames" );
00447             fnGetChannelCount = GetPrefixSymbol< decltype(sound4cl_GetChannelCount
00448 ) >("GetChannelCount" );
00449             fnGetAudioInputCount = GetPrefixSymbol< decltype(sound4cl_GetAudioInputCount
00450 ) >("GetAudioInputCount" );
00451             fnGetAudioOutputCount = GetPrefixSymbol< decltype(sound4cl_GetAudioOutputCount
00452 ) >("GetAudioOutputCount" );
00453             fnGetSampleRate = GetPrefixSymbol< decltype(sound4cl_GetSampleRate
00454 ) >("GetSampleRate" );
00455             fnSetLoggerCallback = GetPrefixSymbol< decltype(sound4cl_SetLoggerCallback
00456 ) >("SetLoggerCallback" );
00457             fnSetLogSeverity = GetPrefixSymbol< decltype(sound4cl_SetLogSeverity
00458 ) >("SetLogSeverity" );
00459             fnNewParameters = GetPrefixSymbol< decltype(sound4cl_NewParameters
00460 ) >("NewParameters" );
00461             fnFreeParameters = GetPrefixSymbol< decltype(sound4cl_FreeParameters
00462 ) >("FreeParameters" );
00463             fnSetParameter = GetPrefixSymbol< decltype(sound4cl_SetParameter
00464 ) >("SetParameter" );
00465             fnGetParameter = GetPrefixSymbol< decltype(sound4cl_GetParameter
00466 ) >("GetParameter" );
00467             fnFreeParameterValue = GetPrefixSymbol< decltype(sound4cl_FreeParameterValue
00468 ) >("FreeParameterValue" );
00469             fnInitProcess = GetPrefixSymbol< decltype(sound4cl_InitProcess
00470 ) >("InitProcess" );
00471             fnInitProcess2 = GetPrefixSymbol< decltype(sound4cl_InitProcess2
00472 ) >("InitProcess2" );
00473             fnTerminateProcess = GetPrefixSymbol< decltype(sound4cl_TerminateProcess

```

```

    ) > ("TerminateProcess"      );
00459         fnExitProcess          = GetPrefixSymbol< decltype(sound4cl_ExitProcess)
    ) > ("ExitProcess"           );
00460         fnStartUpdateThread     = GetPrefixSymbol< decltype(sound4cl_StartUpdateThread)
    ) > ("StartUpdateThread"      );
00461         fnStopUpdateThread      = GetPrefixSymbol< decltype(sound4cl_StopUpdateThread)
    ) > ("StopUpdateThread"       );
00462         fnWaitUpdateThreadReady = GetPrefixSymbol< decltype(sound4cl_WaitUpdateThreadReady)
    ) > ("WaitUpdateThreadReady"  );
00463         fnProcessAudio          = GetPrefixSymbol< decltype(sound4cl_ProcessAudio)
    ) > ("ProcessAudio"           );
00464         fnProcessAudio_Planar   = GetPrefixSymbol< decltype(sound4cl_ProcessAudio_Planar)
    ) > ("ProcessAudio_Planar"    );
00465         fnGetBufferIn           = GetPrefixSymbol< decltype(sound4cl_GetBufferIn)
    ) > ("GetBufferIn"            );
00466         fnGetBufferOut          = GetPrefixSymbol< decltype(sound4cl_GetBufferOut)
    ) > ("GetBufferOut"           );
00467         fnGetEstimatedDelay     = GetPrefixSymbol< decltype(sound4cl_GetEstimatedDelay)
    ) > ("GetEstimatedDelay"      );
00468         fnGetFormatName         = GetPrefixSymbol< decltype(sound4cl_GetFormatName)
    ) > ("GetFormatName"          );
00469         fnGetFormatFromName     = GetPrefixSymbol< decltype(sound4cl_GetFormatFromName)
    ) > ("GetFormatFromName"      );
00470         fnGetBytesFromFormat    = GetPrefixSymbol< decltype(sound4cl_GetBytesFromFormat)
    ) > ("GetBytesFromFormat"     );
00471         fnGetMaxPacketFrame     = GetPrefixSymbol< decltype(sound4cl_GetMaxPacketFrame)
    ) > ("GetMaxPacketFrame"      );
00472         fnAddAudio              = GetPrefixSymbol< decltype(sound4cl_AddAudio)
    ) > ("AddAudio"               );
00473         fnAddPadAudio           = GetPrefixSymbol< decltype(sound4cl_AddPadAudio)
    ) > ("AddPadAudio"            );
00474         fnGetOutputCount        = GetPrefixSymbol< decltype(sound4cl_GetOutputCount)
    ) > ("GetOutputCount"         );
00475         fnGetAudio              = GetPrefixSymbol< decltype(sound4cl_GetAudio)
    ) > ("GetAudio"               );
00476         fnAudioConvertFrom      = GetPrefixSymbol< decltype(sound4cl_AudioConvertFrom)
    ) > ("AudioConvertFrom"       );
00477         fnAudioConvertTo        = GetPrefixSymbol< decltype(sound4cl_AudioConvertTo)
    ) > ("AudioConvertTo"         );
00478         fnStereoToMono          = GetPrefixSymbol< decltype(sound4cl_StereoToMono)
    ) > ("StereoToMono"           );
00479         fnMonoToStereo          = GetPrefixSymbol< decltype(sound4cl_MonoToStereo)
    ) > ("MonoToStereo"           );
00480         fnAudioMonoFromLiveStereo = GetPrefixSymbol<
decltype(sound4cl_AudioMonoFromLiveStereo) > ("AudioMonoFromLiveStereo");
00481         fnAudioMonoToLiveStereo = GetPrefixSymbol< decltype(sound4cl_AudioMonoToLiveStereo)
    ) > ("AudioMonoToLiveStereo" );
00482         fnNewClient             = GetPrefixSymbol< decltype(sound4cl_NewClient)
    ) > ("NewClient"              );
00483         fnDeleteClient          = GetPrefixSymbol< decltype(sound4cl_DeleteClient)
    ) > ("DeleteClient"           );
00484         fnProcessJson           = GetPrefixSymbol< decltype(sound4cl_ProcessJson)
    ) > ("ProcessJson"            );
00485         fnFreeJsonAnswer        = GetPrefixSymbol< decltype(sound4cl_FreeJsonAnswer)
    ) > ("FreeJsonAnswer"         );
00486         fnSaveState             = GetPrefixSymbol< decltype(sound4cl_SaveState)
    ) > ("SaveState"              );
00487     } catch (std::runtime_error& ) {
00488         return false;
00489     }
00490     // Webserver : not for all projects
00491     try {
00492         fnWebserver_tcp          = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp)
    ) > ("Webserver_tcp"           );
00493         fnWebserver_tcp2         = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp2)
    ) > ("Webserver_tcp2"          );
00494         fnWebserver              = GetPrefixSymbol< decltype(sound4cl_Webserver)
    ) > ("Webserver"              );
00495         fnWebserver_Stop         = GetPrefixSymbol< decltype(sound4cl_Webserver_Stop)
    ) > ("Webserver_Stop"          );
00496         fnWebserver_Status       = GetPrefixSymbol< decltype(sound4cl_Webserver_Status)
    ) > ("Webserver_Status"        );
00497     } catch (std::runtime_error& ) {
00498         // Ignored, handler will take care
00499     }
00500     // Stereo/Mono conversion : must be present
00501     try {
00502         // added 2023-03-22
00503         fnStereoToMono_Planar    = GetPrefixSymbol< decltype(sound4cl_StereoToMono_Planar)
    ) > ("StereoToMono_Planar"     );
00504         fnMonoToStereo_Planar    = GetPrefixSymbol< decltype(sound4cl_MonoToStereo_Planar)
    ) > ("MonoToStereo_Planar"     );
00505     } catch (std::runtime_error& ) {
00506         return false;
00507     }
00508     // C functions allowed to be missing
00509

```

```

00510         try {
00511             // added 2023-05-22
00512             fnSetMetadata                                     = GetPrefixSymbol< decltype(sound4cl_SetMetadata
00513         ) >("SetMetadata"                                     );
00514             fnGetMetadataInfos                             = GetPrefixSymbol< decltype(sound4cl_GetMetadataInfos
00515         ) >("GetMetadataInfos"                               );
00516             fnFreeMetadataInfos                             = GetPrefixSymbol< decltype(sound4cl_FreeMetadataInfos
00517         ) >("FreeMetadataInfos"                             );
00518             // added 2023-06-20
00519             fnSetPresetManager                             = GetPrefixSymbol< decltype(sound4cl_SetPresetManager
00520         ) >("SetPresetManager"                               );
00521             fnPresetManager_InformChange                   = GetPrefixSymbol<
00522         decltype(sound4cl_PresetManager_InformChange       ) >("PresetManager_InformChange" );
00523             // added 2023-09-21
00524             fnGetPossibleChunkSizeInFrames                 = GetPrefixSymbol<
00525         decltype(sound4cl_GetPossibleChunkSizeInFrames     ) >("GetPossibleChunkSizeInFrames" );
00526             fnGetProcessChunkFrames                         = GetPrefixSymbol<
00527         decltype(sound4cl_GetProcessChunkFrames           ) >("GetProcessChunkFrames" );
00528             fnInitProcess3                                  = GetPrefixSymbol< decltype(sound4cl_InitProcess3
00529         ) >("InitProcess3"                                   );
00530         } catch (std::runtime_error& ) {
00531             // Ignored, handler will take care
00532         }
00533         // bus: not for all projects
00534         try {
00535             // added 2023-06-26
00536             fnNewBus                                         = GetPrefixSymbol< decltype(sound4cl_NewBus
00537         ) >("NewBus"                                           );
00538             fnFreeBus                                         = GetPrefixSymbol< decltype(sound4cl_FreeBus
00539         ) >("FreeBus"                                           );
00540             fnSetInstanceBus                                 = GetPrefixSymbol< decltype(sound4cl_SetInstanceBus
00541         ) >("SetInstanceBus"                                   );
00542         } catch (std::runtime_error& ) {
00543             // Ignored, handler will take care
00544         }
00545         // Other optional
00546         try {
00547             // added 2024-02-19
00548             fnSetMetadataMulti                               = GetPrefixSymbol< decltype(sound4cl_SetMetadataMulti
00549         ) >("SetMetadataMulti"                                 );
00550         } catch (std::runtime_error& ) {
00551             // Ignored, handler will take care
00552         }
00553         // Other optional, web server only
00554         try {
00555             // added 2024-05-23
00556             fnWebserver_SetAppHealth                         = GetPrefixSymbol<
00557         decltype(sound4cl_Webserver_SetAppHealth           ) >("Webserver_SetAppHealth" );
00558             fnWebserver_GetAppHealth                         = GetPrefixSymbol<
00559         decltype(sound4cl_Webserver_GetAppHealth           ) >("Webserver_GetAppHealth" );
00560             fnWebserver_FreeString                           = GetPrefixSymbol< decltype(sound4cl_Webserver_FreeString
00561         ) >("Webserver_FreeString"                             );
00562         } catch (std::runtime_error& ) {
00563             // Ignored, handler will take care
00564         }
00565         m_bOK=true;
00566         return true;
00567     }
00568     bool IsOk() const {
00569         return m_bOK;
00570     }
00571     void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
00572         sound4cl_SampleFormat fmt)
00573     { fnAudioConvertFrom(payload, output, nSpl, fmt); }
00574     void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat
00575         fmt)
00576     { fnAudioConvertTo(input, payload, nSpl, fmt); }
00577     void StereoToMono(const float *input, float *output, size_t nSpl)
00578     { fnStereoToMono(input, output, nSpl); }
00579     void MonoToStereo(const float *input, float *output, size_t nSpl)
00580     { fnMonoToStereo(input, output, nSpl); }
00581     void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl)
00582     { fnStereoToMono_Planar(inputL, inputR, output, nSpl); }
00583     void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl)
00584     { fnMonoToStereo_Planar(input, outputL, outputR, nSpl); }
00585     void AudioMonoFromLiveStereo(const uint8_t *payload, float *output)
00586     { fnAudioMonoFromLiveStereo(payload, output); }
00587     void AudioMonoToLiveStereo(const float *input, uint8_t *payload)
00588     { fnAudioMonoToLiveStereo(input, payload); }
00589

```

```

00595     std::string GetVersion()
00596     { return std::string(fnGetVersion()); }
00597
00606     unsigned int GetChunkSizeInFrames()
00607     { return fnGetChunkSizeInFrames(); }
00608
00617     std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00618     {
00619         std::vector<unsigned int> list;
00620         if (fnGetPossibleChunkSizeInFrames) {
00621             for (unsigned int* src=fnGetPossibleChunkSizeInFrames();*src;src++) {
00622                 list.push_back(*src);
00623             }
00624         } else {
00625             list.push_back(fnGetChunkSizeInFrames());
00626         }
00627         return list;
00628     }
00629
00635     unsigned int GetChannelCount()
00636     { return fnGetChannelCount(); }
00637
00645     unsigned int GetAudioInputCount()
00646     { return fnGetAudioInputCount(); }
00647
00655     unsigned int GetAudioOutputCount()
00656     { return fnGetAudioOutputCount(); }
00657
00665     unsigned int GetSampleRate()
00666     { return fnGetSampleRate(); }
00667
00674     std::string GetFormatName( const sound4cl_SampleFormat fmt)
00675     { return std::string(fnGetFormatName(fmt)); }
00676
00683     sound4cl_SampleFormat GetFormatFromName( const std::string& name)
00684     { return fnGetFormatFromName(name.c_str()); }
00685
00692     unsigned int GetBytesFromFormat( const sound4cl_SampleFormat fmt)
00693     { return fnGetBytesFromFormat(fmt); }
00694
00698     void SetLogSeverity(LogSeverity severity)
00699     { fnSetLogSeverity(severity); }
00700
00704     void SetLoggerCallback(log_cb_t cb)
00705     { _log_cb=cb; fnSetLoggerCallback(&_log_cb_c); }
00706 };
00707
00713 class CBus {
00714 public:
00715     CBus(CProcessor& dynlib)
00716     : m_dynlib(dynlib)
00717     {
00718         if (m_dynlib.fnNewBus) {
00719             m_bus=m_dynlib.fnNewBus();
00720         }
00721     }
00722     ~CBus()
00723     {
00724         if (m_bus && m_dynlib.fnFreeBus) {
00725             m_dynlib.fnFreeBus(m_bus);
00726         }
00727         m_bus = nullptr;
00728     }
00729     sound4cl_CBus *Get() const { return m_bus; }
00730 private:
00731     CProcessor& m_dynlib;
00732     sound4cl_CBus *m_bus;
00733 };
00734
00740 class CPresetLoader {
00741 public:
00742     CPresetLoader() = default;
00743     virtual ~CPresetLoader() = default;
00744
00745     virtual bool IsReadOnly() = 0;
00746     virtual bool Exists(const std::filesystem::path &name) = 0;
00747     virtual bool Remove(const std::filesystem::path &name) = 0;
00748     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00749     virtual std::vector<std::filesystem::path> GetAll() = 0;
00750     virtual std::string Read(const std::filesystem::path &filename) = 0;
00751     virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00752 };
00753 using PresetChange_Kind = sound4cl_PresetChange_Kind;
00754
00760 class CInstance {
00761 public:
00765     CInstance(CProcessor& dynlib)

```

```

00766         : m_dynlib(dynlib)
00767         {
00768             params = m_dynlib.fnNewParameters();
00769             if (!params) throw std::bad_alloc();
00770         }
00771         // non-copyable
00772         CInstance( const CInstance& ) = delete;
00773         CInstance& operator=( const CInstance& ) = delete;
00774         virtual ~CInstance()
00775         {
00776             Stop();
00777             if (params) m_dynlib.fnFreeParameters(params);
00778         }
00779     bool IsOk() const { return instance!=nullptr; }
00780     void SetParam(const std::string& name, const std::string& value)
00781     {
00782         assert(params);
00783         m_dynlib.fnSetParameter(params, name.c_str(), value.c_str());
00784     }
00785     std::string GetParam(const std::string& name)
00786     {
00787         assert(params);
00788         auto c_value=m_dynlib.fnGetParameter(params, name.c_str());
00789         std::string ret(c_value);
00790         m_dynlib.fnFreeParameterValue(c_value);
00791         return ret;
00792     }
00793     void SetBus(const CBus& bus) {
00794         assert(params);
00795         if (m_dynlib.fnSetInstanceBus) {
00796             m_dynlib.fnSetInstanceBus(params,bus.Get());
00797         }
00798     }
00799     void SetPresetManager(CPresetLoader *preset_manager)
00800     {
00801         if (!preset_manager) return;
00802         if (!m_dynlib.fnSetPresetManager) return;
00803         assert(params);
00804         m_dynlib.fnSetPresetManager(params,
00805             sound4cl_custom_reader,
00806             sound4cl_custom_reader_free,
00807             sound4cl_custom_writer,
00808             sound4cl_custom_exists,
00809             sound4cl_custom_getall,
00810             sound4cl_custom_getall_free,
00811             sound4cl_custom_remove,
00812             sound4cl_custom_rename,
00813             preset_manager->IsReadOnly(),
00814             preset_manager
00815         );
00816     }
00817     bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
00818         Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
00819         json_port = 0, unsigned int frames_per_chunk=0)
00820     {
00821         assert(!instance);
00822         std::string l_save_path;
00823         if (!save_path.empty()) {
00824             l_save_path = save_path.u8string();
00825         }
00826         if (frames_per_chunk==0) {
00827             frames_per_chunk=m_dynlib.GetChunkSizeInFrames();
00828         }
00829         if (m_dynlib.fnInitProcess3) {
00830             instance = m_dynlib.fnInitProcess3(LoginKey.c_str(), RadioName.c_str(),
00831             Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
00832         } else if (frames_per_chunk!=m_dynlib.GetChunkSizeInFrames()) {
00833             return false;
00834         } else {
00835             instance = m_dynlib.fnInitProcess2(LoginKey.c_str(), RadioName.c_str(),
00836             Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
00837         }
00838         if (!instance) return false;
00839         if (!init_metadata.empty()) {
00840             for (auto&& [key,value]: init_metadata) {
00841                 m_dynlib.fnSetMetadata(instance, key.c_str(), value.c_str());
00842             }
00843             init_metadata.clear();
00844         }
00845         update_thread = std::thread([this, json_port]() {
00846             OnUpdateThreadStart();
00847             m_dynlib.fnStartUpdateThread(instance, json_port);
00848             OnUpdateThreadStop();
00849         });
00850     }

```

```

00893         int timeout=5000;
00894         #ifdef DEBUG
00895             timeout=60*1000; // For debugger pauses
00896         #endif
00897         if (m_dynlib.fnWaitUpdateThreadReady(instance, timeout)<0) {
00898             return false;
00899         }
00900         return true;
00901     }
00902
00903     unsigned int GetChunkFrames() {
00904         if (instance && m_dynlib.fnGetProcessChunkFrames) {
00905             return m_dynlib.fnGetProcessChunkFrames(instance);
00906         }
00907         return m_dynlib.fnGetChunkSizeInFrames();
00908     }
00909
00910     void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00911     {
00912         if (m_dynlib.fnPresetManager_InformChange) {
00913             m_dynlib.fnPresetManager_InformChange(instance, relative_path.c_str(),
(sound4cl_PresetChange_Kind) change_kind);
00914         }
00915     }
00916
00917     void SetMetadata(const std::string& key, const char* value)
00918     {
00919         if (m_dynlib.fnSetMetadata) {
00920             if (instance) {
00921                 m_dynlib.fnSetMetadata(instance, key.c_str(), value);
00922             } else {
00923                 init_metadata.push_back( {key, value} );
00924             }
00925         }
00926     }
00927
00928     void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00929     {
00930         if (instance) {
00931             if (m_dynlib.fnSetMetadataMulti) {
00932                 typedef const char* pchar;
00933                 pchar* keyvalue=new pchar[2*list.size()+1];
00934                 size_t n=0;
00935                 for (auto&& [key,value]: list) {
00936                     keyvalue[2*n+0]=key.c_str();
00937                     keyvalue[2*n+1]=value;
00938                     n++;
00939                 }
00940                 keyvalue[2*n+0]=nullptr;
00941
00942                 sound4cl_SetMetadataMulti(instance, keyvalue);
00943
00944                 delete[] keyvalue;
00945             } else {
00946                 for (auto&& [key,value]: list) {
00947                     SetMetadata(key, value);
00948                 }
00949             }
00950         } else {
00951             for (auto&& [key,value]: list) {
00952                 if (value) {
00953                     init_metadata.push_back( {key, value} );
00954                 }
00955             }
00956         }
00957     }
00958
00959     std::vector< std::tuple<std::string, std::string> > GetMetadataInfos()
00960     {
00961         std::vector< std::tuple<std::string, std::string> > values;
00962         if (m_dynlib.fnGetMetadataInfos) {
00963             const char** c_values = m_dynlib.fnGetMetadataInfos(instance);
00964             if (c_values) {
00965                 for (const char** c_value=c_values; *c_value; ) {
00966                     std::string key(*c_value);
00967                     c_value++;
00968                     if (*c_value) {
00969                         std::string descr(*c_value);
00970                         c_value++;
00971                         values.push_back( {key, descr} );
00972                     }
00973                 }
00974             }
00975             m_dynlib.fnFreeMetadataInfos(instance, c_values);
00976         }
00977     }
00978
00979     }
00980
00981     }
00982
00983     }
00984
00985     }
00986
00987     }
00988
00989     }
00990
00991     }
00992
00993     }
00994
00995     }
00996
00997     }
00998
00999     }
01000
01001     }
01002
01003     }
01004
01005     }
01006
01007     }
01008
01009     }
01010
01011     }
01012
01013     }
01014
01015     }
01016
01017     }

```

```

01017         return values;
01018     }
01019
01020     // Can be overloaded, called at start of UpdateThread in the thread
01021     virtual void OnUpdateThreadStart() {}
01022     // Can be overloaded to call at end of UpdateThread in the thread
01023     virtual void OnUpdateThreadStop() {}
01024
01030     bool StartWebServer(int http_port, int https_port=0)
01031     {
01032         assert(instance);
01033         assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01034         if (!m_dynlib.fnWebserver) {
01035             return false;
01036         }
01037         webserver = m_dynlib.fnWebserver(http_port,https_port,instance);
01038         if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01039             return false;
01040         }
01041         int web_status=m_dynlib.fnWebserver_Status(webserver);
01042         if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01043             return false;
01044         }
01045         if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01046             return false;
01047         }
01048         return true;
01049     }
01055     void StopWebServer()
01056     {
01057         if (!m_dynlib.fnWebserver_Stop) return;
01058         if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01059             m_dynlib.fnWebserver_Stop(webserver, 1000);
01060             webserver = SOUND4_INVALID_WEBSERVER_ID;
01061         }
01062     }
01063
01067     void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
content)
01068     {
01069         assert(instance);
01070         if (!m_dynlib.fnWebserver_SetAppHealth) return;
01071         m_dynlib.fnWebserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
content.c_str());
01072     }
01076     void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01077     {
01078         assert(instance);
01079         if (!m_dynlib.fnWebserver_GetAppHealth) return;
01080         char* c_contenttype=NULLPTR;
01081         char* c_content=NULLPTR;
01082         m_dynlib.fnWebserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01083         contenttype=c_contenttype;
01084         content=c_content;
01085         m_dynlib.fnWebserver_FreeString(c_contenttype);
01086         m_dynlib.fnWebserver_FreeString(c_content);
01087     }
01094     int Terminate()
01095     {
01096         if (!instance) return 0;
01097         return m_dynlib.fnTerminateProcess(instance);
01098     }
01099
01105     void Stop()
01106     {
01107         if (!instance) return;
01108         StopWebServer();
01109         if (update_thread.joinable()) {
01110             m_dynlib.fnStopUpdateThread(instance); // this will exit the update thread
01111             update_thread.join();
01112         }
01113         m_dynlib.fnExitProcess(instance);
01114         instance=NULLPTR;
01115     }
01116
01122     unsigned int GetEstimatedDelay()
01123     {
01124         assert(instance);
01125         return m_dynlib.fnGetEstimatedDelay(instance);
01126     }
01127
01138     float* GetBufferIn()
01139     {
01140         assert(instance);
01141         return m_dynlib.fnGetBufferIn(instance);
01142     }
01143

```

```

01154         float* GetBufferOut ()
01155         {
01156             assert(instance);
01157             return m_dynlib.fnGetBufferOut(instance);
01158         }
01164     void ProcessAudio()
01165     {
01166         assert(instance);
01167         m_dynlib.fnProcessAudio(instance, m_dynlib.fnGetBufferIn(instance),
m_dynlib.fnGetBufferOut(instance));
01168     }
01182     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01183     void ProcessAudio_Planar(float const * const *input, float * const *output)
01184     {
01185         assert(instance);
01186         m_dynlib.fnProcessAudio_Planar(instance, input, output);
01187     }
01188
01200     template<typename T>
01201     std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01202     {
01203         assert(instance);
01204         std::vector<T> output;
01205         unsigned int out_offset=0;
01206         unsigned int in_offset=0;
01207         unsigned int todo = input.size();
01208         while (todo>0) {
01209             unsigned int left = AddAudio(&input[in_offset], todo);
01210             unsigned int out_avail = m_dynlib.fnGetOutputCount(instance);
01211             output.resize(out_offset + out_avail);
01212             GetAudio(&output[out_offset], out_avail);
01213             out_offset+=out_avail;
01214             in_offset += todo-left;
01215             todo=left;
01216         }
01217         return output;
01218     }
01219
01225     class CClient {
01226     public:
01227         CClient(sound4cl_CInstance* instance, CProcessor& dynlib)
01228         : m_dynlib(dynlib)
01229         {
01230             assert(instance);
01231             client=m_dynlib.fnNewClient(instance);
01232             if (!client) throw std::bad_alloc();
01233         }
01234         // non-copyable
01235         CClient( const CInstance& ) = delete;
01236         CClient& operator=( const CInstance& ) = delete;
01237         ~CClient()
01238         {
01239             if (client) {
01240                 m_dynlib.fnDeleteClient(client);
01241                 client=nullptr;
01242             }
01243         }
01251         std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01252         {
01253             assert(client);
01254             int need_save=0;
01255             const char *canswer = m_dynlib.fnProcessJson(client, request.c_str(),
&need_save);
01256             if (!canswer) return {};
01257             std::string answer(canswer);
01258             m_dynlib.fnFreeJsonAnswer(canswer);
01259             if (NeedSave) {
01260                 *NeedSave=(need_save!=0);
01261             }
01262             return answer;
01263         }
01264     private:
01265         CProcessor& m_dynlib;
01266         sound4cl_CClientInstance *client = nullptr;
01267     };
01275     std::shared_ptr<CClient> NewClient()
01276     {
01277         assert(instance);
01278         return std::make_shared<CClient>(instance, m_dynlib);
01279     }
01280
01281     protected:
01282     template<typename T>
01283     unsigned int AddAudio(const T* payload, unsigned int nFrame)
01284     {
01285         assert(instance);

```

```

01286         return m_dynlib.fnAddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
helper::SampleFormat<T>::format);
01287     }
01288     template<typename T>
01289     unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01290     {
01291         assert(instance);
01292         return m_dynlib.fnGetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
helper::SampleFormat<T>::format);
01293     }
01294
01295     private:
01296         CProcessor& m_dynlib;
01297         sound4cl_CParameters* params = nullptr;
01298         sound4cl_CInstance* instance = nullptr;
01299         std::thread update_thread;
01300         std::vector< std::pair<std::string, std::string> > init_metadata;
01301         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01302     };
01303
01304 }; // namespace sound4
01305
01306 // bridge C callbacks to CPresetLoader
01307 extern "C" {
01308     static char *sound4cl_custom_reader(const fs_char *filename, void* userdata) {
01309         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01310         std::string content=preset_loader->Read(filename);
01311         return strdup(content.c_str());
01312     }
01313     static void sound4cl_custom_reader_free(char *content, void* userdata) {
01314         if (!content) return;
01315         free(content);
01316     }
01317     static int sound4cl_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01318         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01319         auto res=preset_loader->Write(filename, content);
01320         return res?0:-1;
01321     }
01322     static int sound4cl_custom_exists(const fs_char *filename, void* userdata) {
01323         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01324         auto res=preset_loader->Exists(filename);
01325         return res?0:-1;
01326     }
01327     static fs_char** sound4cl_custom_getall(void* userdata) {
01328         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01329         auto res=preset_loader->GetAll();
01330         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01331         for (size_t n=0;n<res.size();n++) {
01332             all[n]=fs_strdup(res[n].c_str());
01333         }
01334         all[res.size()]=nullptr;
01335         return all;
01336     }
01337     static void sound4cl_custom_getall_free(fs_char** all, void* userdata) {
01338         if (!all) return;
01339         for (fs_char** one=all;*one;one++) {
01340             free(*one);
01341         }
01342         free(all);
01343     }
01344     static int sound4cl_custom_remove(const fs_char *filename, void* userdata) {
01345         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01346         auto res=preset_loader->Remove(filename);
01347         return res?0:-1;
01348     }
01349     static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01350         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01351         auto res=preset_loader->Rename(from, to);
01352         return res?0:-1;
01353     }
01354 };
01355
01356 };

```

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 lsReadOnly, 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_C←Instance *instance)
- int [sound4cl_Webserver_Stop](#) (uint64_t id, int timeout_ms)
- int [sound4cl_Webserver_Status](#) (uint64_t id)
- void [sound4cl_Webserver_SetAppHealth](#) (struct sound4cl_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void [sound4cl_Webserver_GetAppHealth](#) (struct sound4cl_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void [sound4cl_Webserver_FreeString](#) (char *str)

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
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_FreeString()

```
void sound4cl_Webserver_FreeString (
    char * str )
```

11.17.4.62 sound4cl_Webserver_GetAppHealth()

```
void sound4cl_Webserver_GetAppHealth (
    struct sound4cl_CInstance * instance,
    int * httpcode,
    char ** contenttype,
    char ** content )
```

11.17.4.63 sound4cl_Webserver_SetAppHealth()

```
void sound4cl_Webserver_SetAppHealth (
    struct sound4cl_CInstance * instance,
    int httpcode,
    const char * contenttype,
    const char * content )
```

11.17.4.64 sound4cl_Webserver_Status()

```
int sound4cl_Webserver_Status (
    uint64_t id )
```

11.17.4.65 sound4cl_Webserver_Stop()

```
int sound4cl_Webserver_Stop (
    uint64_t id,
    int timeout_ms )
```

11.17.4.66 sound4cl_Webserver_tcp()

```
uint64_t sound4cl_Webserver_tcp (
    unsigned int listenport,
    unsigned int listenport_secure,
    const char * socket_ip,
    unsigned int socket_port )
```

11.17.4.67 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

[Go to the documentation of this file.](#)

```
00001
00006 #pragma once
00007
00008 #include <stdint.h>
00009 #ifdef __unix__
00010     #include <sys/types.h>
00011 #endif
00012
00013 #ifdef _WIN32
00014     #include <stringapiset.h>
00015     #include <wchar.h>
00017     #define fs_strdup wcsdup
00018     typedef wchar_t fs_char;
00019     #ifdef _MSC_VER
```

```

00020         #define __BYTE_ORDER__ REG_DWORD
00021         #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00022         #define __ORDER_BIG_ENDIAN__ REG_DWORD_BIG_ENDIAN
00023     #endif
00024 #else // !_WIN32
00026     #define fs_strdup strdup
00027     typedef char fs_char;
00028 #endif // !_WIN32
00029
00030 #ifdef __cplusplus
00031 extern "C" {
00032 #endif // __cplusplus
00033
00034 struct sound4cl_SOUND4_ProcessInfo {
00035     size_t struct_size;
00036     const char *name;
00037     const char *prefix;
00038     const char *shortname;
00039     const char *version;
00040 };
00041 enum sound4cl_LogSeverity {
00042     sound4cl_none = 0,
00043     sound4cl_fatal = 1,
00044     sound4cl_error = 2,
00045     sound4cl_warning = 3,
00046     sound4cl_info = 4,
00047     sound4cl_verbose = 5,
00048     sound4cl_verbose2 = 6,
00049     sound4cl_verbose3 = 7,
00050     sound4cl_verbose4 = 8,
00051     sound4cl_verbose5 = 9,
00052 };
00053 struct sound4cl_CInstance;
00054 struct sound4cl_CParameters;
00055 struct sound4cl_CBus;
00056 struct sound4cl_CClientInstance;
00057 enum sound4cl_PresetChange_Kind {
00058     change_kind_created = 1,
00059     change_kind_modified ,
00060     change_kind_deleted ,
00061 };
00062 #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
00063 #define SOUND4_WEBSERVER_HTTP_OK (1<0)
00064 #define SOUND4_WEBSERVER_HTTPS_OK (1<1)
00065
00066 enum sound4cl_SampleFormat {
00067     INVALID_FORMAT=0,
00068     S16_LE,
00069     S16_BE,
00070     S24_LE,
00071     S24_BE,
00072     S32_LE,
00073     S32_BE,
00074     F32_LE,
00075     F32_BE,
00076
00077     // Native definitions, only links to the physical ones
00078     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00079         S16_NATIVE=S16_LE,
00080         S24_NATIVE=S24_LE,
00081         S32_NATIVE=S32_LE,
00082         F32_NATIVE=F32_LE,
00083     #else
00084         S16_NATIVE=S16_BE,
00085         S24_NATIVE=S24_BE,
00086         S32_NATIVE=S32_BE,
00087         F32_NATIVE=F32_BE,
00088     #endif
00089 };
00090
00091 // Native definitions, only links to the physical ones
00092 #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00093     S16_NATIVE=S16_LE,
00094     S24_NATIVE=S24_LE,
00095     S32_NATIVE=S32_LE,
00096     F32_NATIVE=F32_LE,
00097 #else
00098     S16_NATIVE=S16_BE,
00099     S24_NATIVE=S24_BE,
00100     S32_NATIVE=S32_BE,
00101     F32_NATIVE=F32_BE,
00102 #endif
00103
00104 const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo();
00105 const char *sound4cl_GetVersion();
00106 unsigned int sound4cl_GetChunkSizeInFrames();
00107 unsigned int* sound4cl_GetPossibleChunkSizeInFrames();
00108 unsigned int sound4cl_GetChannelCount();
00109 unsigned int sound4cl_GetAudioInputCount();
00110 unsigned int sound4cl_GetAudioOutputCount();
00111 unsigned int sound4cl_GetSampleRate();
00112 typedef void (*sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *);
00113 void sound4cl_SetLoggerCallback( sound4cl_loggerfn logger );
00114 void sound4cl_SetLogSeverity( enum sound4cl_LogSeverity severity );
00115 struct sound4cl_CParameters *sound4cl_NewParameters();
00116 void sound4cl_FreeParameters(struct sound4cl_CParameters *params);
00117 void sound4cl_SetParameter(struct sound4cl_CParameters *params, const char *name, const char *value);
00118 const char *sound4cl_GetParameter(struct sound4cl_CParameters *params, const char *name);
00119 void sound4cl_FreeParameterValue(const char *value);
00120 struct sound4cl_CInstance *sound4cl_InitProcess(

```

```

00111     const char *LoginKey,
00112     const char *RadioName,
00113     const char *Access_Key_ID,
00114     const char *Access_Key_Secret,
00115     const char *save_path
00116 );
00117 struct sound4cl_CInstance *sound4cl_InitProcess2(
00118     const char *LoginKey,
00119     const char *RadioName,
00120     const char *Access_Key_ID,
00121     const char *Access_Key_Secret,
00122     const char *save_path,
00123     const struct sound4cl_CParameters *parameters
00124 );
00125 struct sound4cl_CInstance *sound4cl_InitProcess3(
00126     const char *LoginKey,
00127     const char *RadioName,
00128     const char *Access_Key_ID,
00129     const char *Access_Key_Secret,
00130     const char *save_path,
00131     const struct sound4cl_CParameters *parameters,
00132     unsigned int frames_per_chunk
00133 );
00134 unsigned int sound4cl_GetProcessChunkFrames(struct sound4cl_CInstance *instance);
00135 int sound4cl_TerminateProcess(struct sound4cl_CInstance *instance);
00136 void sound4cl_ExitProcess(struct sound4cl_CInstance *instance);
00137 struct sound4cl_CBus* sound4cl_NewBus();
00138 void sound4cl_FreeBus(struct sound4cl_CBus *bus);
00139 void sound4cl_SetInstanceBus(struct sound4cl_CParameters *params, struct sound4cl_CBus *bus);
00140 typedef char * (*sound4cl_storage_reader) (const fs_char *filename, void* userdata);
00141 typedef void (*sound4cl_storage_reader_free) (char *content, void* userdata);
00142 typedef int (*sound4cl_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00143 typedef int (*sound4cl_storage_exists) (const fs_char *filename, void* userdata);
00144 typedef fs_char** (*sound4cl_storage_getall) (void* userdata);
00145 typedef void (*sound4cl_storage_getall_free) (fs_char**all, void* userdata);
00146 typedef int (*sound4cl_storage_remove) (const fs_char *filename, void* userdata);
00147 typedef int (*sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00148 void sound4cl_SetPresetManager(struct sound4cl_CParameters *params
00149     , sound4cl_storage_reader reader, sound4cl_storage_reader_free
00150     , sound4cl_storage_writer writer
00151     , sound4cl_storage_exists exists
00152     , sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free
00153     , sound4cl_storage_remove remove
00154     , sound4cl_storage_rename rename
00155     , int IsReadOnly
00156     , void* userdata
00157 );
00158 void sound4cl_PresetManager_InformChange(struct sound4cl_CInstance *instance, const fs_char
00159     *relative_path, enum sound4cl_PresetChange_Kind change_kind);
00160 void sound4cl_SetMetadata(struct sound4cl_CInstance *instance, const char* key, const char* value);
00161 void sound4cl_SetMetadataMulti(struct sound4cl_CInstance *instance, const char** keyvalue);
00162 const char** sound4cl_GetMetadataInfos(struct sound4cl_CInstance *instance);
00163 void sound4cl_FreeMetadataInfos(struct sound4cl_CInstance *instance, const char** infos);
00164 void sound4cl_StartUpdateThread(struct sound4cl_CInstance *instance, unsigned int port);
00165 void sound4cl_StopUpdateThread(struct sound4cl_CInstance *instance);
00166 int sound4cl_WaitUpdateThreadReady(struct sound4cl_CInstance *instance, int milliseconds);
00167 void sound4cl_ProcessAudio(struct sound4cl_CInstance *instance, const float *input, float *output );
00168 void sound4cl_ProcessAudio_Planar(struct sound4cl_CInstance *instance
00169     , float const * const *input
00170     , float * const *output);
00171 float *sound4cl_GetBufferIn(struct sound4cl_CInstance *instance);
00172 float *sound4cl_GetBufferOut(struct sound4cl_CInstance *instance);
00173 unsigned int sound4cl_GetEstimatedDelay(struct sound4cl_CInstance *instance);
00174 const char *sound4cl_GetFormatName(const enum sound4cl_SampleFormat fmt);
00175 enum sound4cl_SampleFormat sound4cl_GetFormatFromName(const char *name);
00176 unsigned int sound4cl_GetBytesFromFormat(const enum sound4cl_SampleFormat fmt);
00177 unsigned int sound4cl_GetMaxPacketFrame(struct sound4cl_CInstance *instance);
00178 unsigned int sound4cl_AddAudio(struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned
00179     int nFrame, enum sound4cl_SampleFormat fmt );
00180 unsigned int sound4cl_AddPadAudio(struct sound4cl_CInstance *instance);
00181 unsigned int sound4cl_GetOutputCount(struct sound4cl_CInstance *instance);
00182 unsigned int sound4cl_GetAudio(struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int
00183     max_nFrame, enum sound4cl_SampleFormat fmt );
00184 void sound4cl_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
00185     sound4cl_SampleFormat fmt);
00186 void sound4cl_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
00187     sound4cl_SampleFormat fmt);
00188 void sound4cl_StereoToMono(const float *input, float *output, size_t nFrame);
00189 void sound4cl_MonoToStereo(const float *input, float *output, size_t nFrame);
00190 void sound4cl_StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t
00191     nFrame);
00192 void sound4cl_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nFrame);
00193 void sound4cl_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
00194 void sound4cl_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
00195 struct sound4cl_CClientInstance *sound4cl_NewClient(struct sound4cl_CInstance *instance);
00196 void sound4cl_DeleteClient(struct sound4cl_CClientInstance *client);
00197 const char *sound4cl_ProcessJson(struct sound4cl_CClientInstance *client, const char *json_str, int

```

```
    *need_save);
00192 void sound4cl_FreeJsonAnswer(const char *json_str);
00193 int sound4cl_SaveState(struct sound4cl_CInstance *instance);
00194 uint64_t sound4cl_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure, const char
    *socket_ip, unsigned int socket_port);
00195 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);
00196 uint64_t sound4cl_Webserver(unsigned int listenport, unsigned int listenport_secure, struct
    sound4cl_CInstance *instance);
00197 int sound4cl_Webserver_Stop(uint64_t id, int timeout_ms);
00198 int sound4cl_Webserver_Status(uint64_t id);
00199 void sound4cl_Webserver_SetAppHealth(struct sound4cl_CInstance *instance, int httpcode, const char*
    contenttype, const char* content);
00200 void sound4cl_Webserver_GetAppHealth(struct sound4cl_CInstance *instance, int* httpcode, char**
    contenttype, char** content);
00201 void sound4cl_Webserver_FreeString(char* str);
00202
00203 #ifdef __cplusplus
00204 } // extern "C"
00205 #endif // __cplusplus
```


Index

- ~CBus
 - sound4::CBus, [81](#)
- ~CClient
 - sound4::CInstance::CClient, [83](#)
 - sound4::impact::CInstance::CClient, [84](#)
 - sound4::impact::dyn::CInstance::CClient, [86](#)
- ~CDynLoader
 - sound4::helper::CDynLoader, [99](#)
 - sound4::impact::dyn::helper::CDynLoader, [102](#)
- ~CInstance
 - sound4::CInstance, [105](#)
 - sound4::impact::CInstance, [115](#)
 - sound4::impact::dyn::CInstance, [124](#)
- ~CPresetLoader
 - sound4::CPresetLoader, [133](#)
 - sound4::impact::CPresetLoader, [135](#)
 - sound4::impact::dyn::CPresetLoader, [137](#)
- ~MyPresetManager
 - MyPresetManager, [160](#), [161](#)
- AddAudio
 - sound4::CInstance, [105](#)
 - sound4::impact::CInstance, [115](#)
 - sound4::impact::dyn::CDynLib, [89](#)
 - sound4::impact::dyn::CInstance, [124](#)
- AddPadAudio
 - sound4::impact::dyn::CDynLib, [89](#)
- Audio convert helpers, [47](#)
 - impact_AudioConvertFrom, [47](#)
 - impact_AudioConvertTo, [48](#)
 - impact_AudioMonoFromLiveStereo, [48](#)
 - impact_AudioMonoToLiveStereo, [49](#)
 - impact_MonoToStereo, [49](#)
 - impact_MonoToStereo_Planar, [49](#)
 - impact_StereoToMono, [50](#)
 - impact_StereoToMono_Planar, [50](#)
- Audio managing (any audio), [43](#)
 - F32_BE, [44](#)
 - F32_LE, [44](#)
 - F32_NATIVE, [44](#)
 - impact_AddAudio, [44](#)
 - impact_AddPadAudio, [45](#)
 - impact_GetAudio, [45](#)
 - impact_GetBytesFromFormat, [45](#)
 - impact_GetFormatFromName, [46](#)
 - impact_GetFormatName, [46](#)
 - impact_GetMaxPacketFrame, [46](#)
 - 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, [140](#)
 - sound4::impact::dyn::CDynLib, [89](#)
- AudioConvertTo
 - sound4::CProcessor, [140](#)
 - sound4::impact::dyn::CDynLib, [90](#)
- AudioMonoFromLiveStereo
 - sound4::CProcessor, [141](#)
 - sound4::impact::dyn::CDynLib, [90](#)
- AudioMonoToLiveStereo
 - sound4::CProcessor, [141](#)
 - sound4::impact::dyn::CDynLib, [90](#)
- CBus
 - sound4::CBus, [81](#)
 - sound4::CProcessor, [146](#)
- CClient
 - sound4::CInstance::CClient, [82](#)
 - sound4::impact::CInstance::CClient, [84](#)
 - sound4::impact::dyn::CInstance::CClient, [86](#)
- CDynLib
 - sound4::impact::dyn::CDynLib, [88](#)
- CDynLoader
 - sound4::helper::CDynLoader, [99](#)
 - sound4::impact::dyn::helper::CDynLoader, [102](#)
- change_kind_created
 - Custom preset management, [36](#)
 - sound4cl_cdef.h, [245](#)
- change_kind_deleted
 - Custom preset management, [36](#)
 - sound4cl_cdef.h, [245](#)
- change_kind_modified
 - Custom preset management, [36](#)
 - sound4cl_cdef.h, [245](#)
- CHANNEL_COUNT
 - sound4.impact.cl-pipe.cpp, [184](#)
- CHUNK_SIZE
 - sound4.impact.cl-pipe.cpp, [184](#)
- ChunkFrames
 - sound4::impact, [76](#)
 - sound4::impact::dyn, [78](#)

- CInstance
 - sound4::CInstance, [105](#)
 - sound4::CProcessor, [146](#)
 - sound4::impact::CInstance, [115](#)
 - sound4::impact::dyn::CInstance, [124](#)
- Client's JSON config interfacing, [51](#)
 - impact_DeleteClient, [51](#)
 - impact_FreeJsonAnswer, [51](#)
 - impact_NewClient, [51](#)
 - impact_ProcessJson, [52](#)
 - impact_SaveState, [52](#)
- Close
 - sound4::helper::CDynLoader, [100](#)
 - sound4::impact::dyn::helper::CDynLoader, [102](#)
- CPresetLoader
 - sound4::CPresetLoader, [133](#)
 - sound4::impact::CPresetLoader, [135](#)
 - sound4::impact::dyn::CPresetLoader, [137](#)
- CProcessor
 - sound4::CProcessor, [140](#)
- Create
 - sound4::CInstance, [105](#)
 - sound4::impact::CInstance, [115](#)
 - sound4::impact::dyn::CInstance, [124](#)
- Custom preset management, [32](#)
 - change_kind_created, [36](#)
 - change_kind_deleted, [36](#)
 - change_kind_modified, [36](#)
 - fs_char, [33](#)
 - impact_PresetChange_Kind, [36](#)
 - impact_PresetManager_InformChange, [36](#)
 - impact_SetPresetManager, [37](#)
 - impact_storage_exists, [33](#)
 - impact_storage_getall, [33](#)
 - impact_storage_getall_free, [34](#)
 - impact_storage_reader, [34](#)
 - impact_storage_reader_free, [34](#)
 - impact_storage_remove, [35](#)
 - impact_storage_rename, [35](#)
 - impact_storage_writer, [35](#)
- DeleteClient
 - sound4::impact::dyn::CDynLib, [90](#)
- Deprecated List, [3](#)
- 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, [74](#)
 - sound4::impact::dyn::helper, [80](#)
- DynFuncHolder
 - sound4::helper::DynFuncHolder< T >, [156](#)
 - sound4::impact::dyn::helper::DynFuncHolder< T >, [158](#)
- DynLib_t
 - sound4::helper, [74](#)
 - sound4::impact::dyn::helper, [80](#)
- Environment variables and parameters, [19](#)
- error
 - Global setup, [23](#)
 - sound4, [72](#)
- Exists
 - MyPresetManager, [161](#)
 - sound4::CPresetLoader, [133](#)
 - sound4::impact::CPresetLoader, [135](#)
 - sound4::impact::dyn::CPresetLoader, [137](#)
- ExitProcess
 - sound4::impact::dyn::CDynLib, [90](#)
- F32_BE
 - Audio managing (any audio), [44](#)
 - sound4cl_cdef.h, [245](#)
- F32_LE
 - Audio managing (any audio), [44](#)
 - sound4cl_cdef.h, [245](#)
- F32_NATIVE
 - Audio managing (any audio), [44](#)
 - sound4cl_cdef.h, [245](#)
- fatal
 - Global setup, [23](#)
 - sound4, [72](#)
- fnAddAudio
 - sound4::CProcessor, [146](#)
- fnAddPadAudio
 - sound4::CProcessor, [146](#)
- fnAudioConvertFrom
 - sound4::CProcessor, [146](#)
- fnAudioConvertTo
 - sound4::CProcessor, [147](#)
- fnAudioMonoFromLiveStereo
 - sound4::CProcessor, [147](#)
- fnAudioMonoToLiveStereo
 - sound4::CProcessor, [147](#)
- fnDeleteClient
 - sound4::CProcessor, [147](#)
- fnExitProcess
 - sound4::CProcessor, [147](#)
- fnFreeBus
 - sound4::CProcessor, [147](#)
- fnFreeJsonAnswer
 - sound4::CProcessor, [147](#)
- fnFreeMetadataInfos
 - sound4::CProcessor, [148](#)
- fnFreeParameters
 - sound4::CProcessor, [148](#)
- fnFreeParameterValue
 - sound4::CProcessor, [148](#)
- fnGetAudio
 - sound4::CProcessor, [148](#)
- fnGetAudioInputCount
 - sound4::CProcessor, [148](#)
- fnGetAudioOutputCount
 - sound4::CProcessor, [148](#)

- fnGetBufferIn
 - sound4::CProcessor, [148](#)
- fnGetBufferOut
 - sound4::CProcessor, [149](#)
- fnGetBytesFromFormat
 - sound4::CProcessor, [149](#)
- fnGetChannelCount
 - sound4::CProcessor, [149](#)
- fnGetChunkSizeInFrames
 - sound4::CProcessor, [149](#)
- fnGetEstimatedDelay
 - sound4::CProcessor, [149](#)
- fnGetFormatFromName
 - sound4::CProcessor, [149](#)
- fnGetFormatName
 - sound4::CProcessor, [149](#)
- fnGetMaxPacketFrame
 - sound4::CProcessor, [150](#)
- fnGetMetadataInfos
 - sound4::CProcessor, [150](#)
- fnGetOutputCount
 - sound4::CProcessor, [150](#)
- fnGetParameter
 - sound4::CProcessor, [150](#)
- fnGetPossibleChunkSizeInFrames
 - sound4::CProcessor, [150](#)
- fnGetProcessChunkFrames
 - sound4::CProcessor, [150](#)
- fnGetSampleRate
 - sound4::CProcessor, [150](#)
- fnGetVersion
 - sound4::CProcessor, [151](#)
- fnInitProcess
 - sound4::CProcessor, [151](#)
- fnInitProcess2
 - sound4::CProcessor, [151](#)
- fnInitProcess3
 - sound4::CProcessor, [151](#)
- fnMonoToStereo
 - sound4::CProcessor, [151](#)
- fnMonoToStereo_Planar
 - sound4::CProcessor, [151](#)
- fnNewBus
 - sound4::CProcessor, [151](#)
- fnNewClient
 - sound4::CProcessor, [152](#)
- fnNewParameters
 - sound4::CProcessor, [152](#)
- fnPresetManager_InformChange
 - sound4::CProcessor, [152](#)
- fnProcessAudio
 - sound4::CProcessor, [152](#)
- fnProcessAudio_Planar
 - sound4::CProcessor, [152](#)
- fnProcessJson
 - sound4::CProcessor, [152](#)
- fnSaveState
 - sound4::CProcessor, [152](#)
- fnSetInstanceBus
 - sound4::CProcessor, [153](#)
- fnSetLoggerCallback
 - sound4::CProcessor, [153](#)
- fnSetLogSeverity
 - sound4::CProcessor, [153](#)
- fnSetMetadata
 - sound4::CProcessor, [153](#)
- fnSetMetadataMulti
 - sound4::CProcessor, [153](#)
- fnSetParameter
 - sound4::CProcessor, [153](#)
- fnSetPresetManager
 - sound4::CProcessor, [153](#)
- fnStartUpdateThread
 - sound4::CProcessor, [154](#)
- fnStereoToMono
 - sound4::CProcessor, [154](#)
- fnStereoToMono_Planar
 - sound4::CProcessor, [154](#)
- fnStopUpdateThread
 - sound4::CProcessor, [154](#)
- fnTerminateProcess
 - sound4::CProcessor, [154](#)
- fnWaitUpdateThreadReady
 - sound4::CProcessor, [154](#)
- fnWebserver
 - sound4::CProcessor, [154](#)
- fnWebserver_FreeString
 - sound4::CProcessor, [155](#)
- fnWebserver_GetAppHealth
 - sound4::CProcessor, [155](#)
- fnWebserver_SetAppHealth
 - sound4::CProcessor, [155](#)
- fnWebserver_Status
 - sound4::CProcessor, [155](#)
- fnWebserver_Stop
 - sound4::CProcessor, [155](#)
- fnWebserver_tcp
 - sound4::CProcessor, [155](#)
- fnWebserver_tcp2
 - sound4::CProcessor, [155](#)
- format
 - sound4::helper::SampleFormat< float >, [166](#)
 - sound4::helper::SampleFormat< int16_t >, [167](#)
 - sound4::helper::SampleFormat< int32_t >, [169](#)
 - sound4::impact::dyn::helper::SampleFormat< float >, [166](#)
 - sound4::impact::dyn::helper::SampleFormat< int16_t >, [168](#)
 - sound4::impact::dyn::helper::SampleFormat< int32_t >, [170](#)
 - sound4::impact::helper::SampleFormat< float >, [167](#)
 - sound4::impact::helper::SampleFormat< int16_t >, [168](#)
 - sound4::impact::helper::SampleFormat< int32_t >, [170](#)

- FreeJsonAnswer
 - sound4::impact::dyn::CDynLib, 90
- FreeMetadataInfos
 - sound4::impact::dyn::CDynLib, 90
- FreeParameters
 - sound4::impact::dyn::CDynLib, 91
- FreeParameterValue
 - sound4::impact::dyn::CDynLib, 91
- fs_char
 - Custom preset managagement, 33
 - sound4cl_cdef.h, 243
- fs_strdup
 - sound4.impact.cl.hpp, 202
 - sound4.impact.cl_dyn.hpp, 211
 - sound4cl_cdef.h, 242
- Get
 - sound4::CBus, 82
- GetAll
 - MyPresetManager, 161
 - sound4::CPresetLoader, 133
 - sound4::impact::CPresetLoader, 135
 - sound4::impact::dyn::CPresetLoader, 137
- GetAudio
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CDynLib, 91
 - sound4::impact::dyn::CInstance, 125
- GetAudioInputCount
 - sound4::CProcessor, 141
 - sound4::impact::dyn::CDynLib, 91
- GetAudioOutputCount
 - sound4::CProcessor, 141
 - sound4::impact::dyn::CDynLib, 91
- GetBufferIn
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CDynLib, 91
 - sound4::impact::dyn::CInstance, 125
- GetBufferOut
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CDynLib, 91
 - sound4::impact::dyn::CInstance, 125
- GetBytesFromFormat
 - sound4::CProcessor, 141
 - sound4::impact::dyn::CDynLib, 92
- GetChannelCount
 - sound4::CProcessor, 142
 - sound4::impact::dyn::CDynLib, 92
- GetChunkFrames
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CInstance, 126
- GetChunkSizeInFrames
 - sound4::CProcessor, 142
 - sound4::impact::dyn::CDynLib, 92
- GetEnv
 - sound4.impact.cl-pipe-cxx.cpp, 175
 - sound4.impact.cl-pipe-dyn.cpp, 179
 - sound4cl-pipe.cpp, 189
- GetEstimatedDelay
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CDynLib, 92
 - sound4::impact::dyn::CInstance, 126
- GetFormatFromName
 - sound4::CProcessor, 142
 - sound4::impact::dyn::CDynLib, 92
- GetFormatName
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 92
- GetMaxPacketFrame
 - sound4::impact::dyn::CDynLib, 92
- GetMetadataInfos
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CDynLib, 93
 - sound4::impact::dyn::CInstance, 126
- GetOutputCount
 - sound4::impact::dyn::CDynLib, 93
- GetParam
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CInstance, 126
- GetParameter
 - sound4::impact::dyn::CDynLib, 93
- GetPossibleChunkSizeInFrames
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 93
- GetPrefixSymbol
 - sound4::CProcessor, 143
- GetProcessChunkFrames
 - sound4::impact::dyn::CDynLib, 93
- GetSampleRate
 - sound4::CProcessor, 144
 - sound4::impact::dyn::CDynLib, 93
- GetSymbol
 - sound4::helper::CDynLoader, 100
 - sound4::impact::dyn::helper::CDynLoader, 102
- GetSymbol_safe
 - sound4::helper::CDynLoader, 100
 - sound4::impact::dyn::helper::CDynLoader, 102
- GetThisLibraryPath
 - sound4::helper::CDynLoader, 100
 - sound4::impact::dyn::helper::CDynLoader, 102
- GetVersion
 - sound4::CProcessor, 144
 - sound4::impact::dyn::CDynLib, 93
- GetWebServerAppHealth
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CInstance, 127
- Global setup, 20
 - error, 23
 - fatal, 23
 - IMPACT_AUDIO_INPUT_COUNT, 21

- IMPACT_AUDIO_OUTPUT_COUNT, 21
- IMPACT_AUDIOFRAME_COUNT, 21
- IMPACT_CHANNEL_COUNT, 22
- 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, 27
- verbose, 23
- verbose2, 23
- verbose3, 23
- verbose4, 23
- verbose5, 23
- warning, 23
- impact_AddAudio
 - Audio managing (any audio), 44
- 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, 47
- impact_AudioConvertTo
 - Audio convert helpers, 48
- IMPACT_AUDIOFRAME_COUNT
 - Global setup, 21
- impact_AudioMonoFromLiveStereo
 - Audio convert helpers, 48
- impact_AudioMonoToLiveStereo
 - Audio convert helpers, 49
- impact_CClientInstance, 159
- IMPACT_CHANNEL_COUNT
 - Global setup, 22
- impact_CInstance, 159
- impact_CParameters, 159
- IMPACT_DECL
 - sound4.impact.cl.h, 196
- impact_DeleteClient
 - Client's JSon config interfacing, 51
- impact_ExitProcess
 - Process creation/destruction, 27
- impact_FreeJsonAnswer
 - Client's JSon config interfacing, 51
- impact_FreeMetadataInfos
 - Metadata Management, 38
- impact_FreeParameters
 - Process creation/destruction, 28
- impact_FreeParameterValue
 - Process creation/destruction, 28
- 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), 45
- 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), 46
- impact_GetMaxPacketFrame
 - Audio managing (any audio), 46
- impact_GetMetadataInfos
 - Metadata Management, 38
- impact_GetOutputCount
 - Audio managing (any audio), 47
- impact_GetParameter
 - Process creation/destruction, 28
- 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, 196
- IMPACT_HAS_WEBSERVER
 - sound4.impact.cl.h, 197
- impact_InitProcess
 - Process creation/destruction, 29
- impact_InitProcess2
 - Process creation/destruction, 29
- impact_InitProcess3
 - Process creation/destruction, 29
- impact_loggerfn
 - Global setup, 22
- impact_LogSeverity
 - Global setup, 22
- impact_MonoToStereo
 - Audio convert helpers, 49
- impact_MonoToStereo_Planar

- Audio convert helpers, [49](#)
- impact_NewClient
 - Client's JSon config interfacing, [51](#)
- impact_NewParameters
 - Process creation/destruction, [31](#)
- impact_PresetChange_Kind
 - Custom preset management, [36](#)
- impact_PresetManager_InformChange
 - Custom preset management, [36](#)
- impact_ProcessAudio
 - Direct processing, [42](#)
- impact_ProcessAudio_Planar
 - Direct processing, [43](#)
- impact_ProcessJson
 - Client's JSon config interfacing, [52](#)
- IMPACT_SAMPLE_RATE
 - Global setup, [22](#)
- impact_SampleFormat
 - Audio managing (any audio), [44](#)
- impact_SaveState
 - Client's JSon config interfacing, [52](#)
- impact_SetLoggerCallback
 - Global setup, [25](#)
- impact_SetLogSeverity
 - Global setup, [25](#)
- impact_SetMetadata
 - Metadata Management, [38](#)
- impact_SetMetadataMulti
 - Metadata Management, [39](#)
- impact_SetParameter
 - Process creation/destruction, [31](#)
- impact_SetPresetManager
 - Custom preset management, [37](#)
- impact_StartUpdateThread
 - Update Management, [39](#)
- impact_StereoToMono
 - Audio convert helpers, [50](#)
- impact_StereoToMono_Planar
 - Audio convert helpers, [50](#)
- impact_StopUpdateThread
 - Update Management, [40](#)
- impact_storage_exists
 - Custom preset management, [33](#)
- impact_storage_getall
 - Custom preset management, [33](#)
- impact_storage_getall_free
 - Custom preset management, [34](#)
- impact_storage_reader
 - Custom preset management, [34](#)
- impact_storage_reader_free
 - Custom preset management, [34](#)
- impact_storage_remove
 - Custom preset management, [35](#)
- impact_storage_rename
 - Custom preset management, [35](#)
- impact_storage_writer
 - Custom preset management, [35](#)
- impact_TerminateProcess
 - Process creation/destruction, [31](#)
- impact_WaitUpdateThreadReady
 - Update Management, [40](#)
- impact_Webserver
 - Web server, [60](#)
- impact_Webserver_FreeString
 - Web server, [61](#)
- impact_Webserver_GetAppHealth
 - Web server, [61](#)
- impact_Webserver_SetAppHealth
 - Web server, [61](#)
- impact_Webserver_Status
 - Web server, [63](#)
- impact_Webserver_Stop
 - Web server, [63](#)
- impact_Webserver_tcp
 - Web server, [64](#)
- impact_Webserver_tcp2
 - Web server, [64](#)
- info
 - Global setup, [23](#)
 - sound4, [73](#)
- InitProcess
 - sound4::impact::dyn::CDynLib, [94](#)
- InitProcess2
 - sound4::impact::dyn::CDynLib, [94](#)
- InitProcess3
 - sound4::impact::dyn::CDynLib, [94](#)
- InputChannels
 - sound4::impact, [76](#)
 - sound4::impact::dyn, [78](#)
- InputSampleSize
 - sound4::impact, [76](#)
 - sound4::impact::dyn, [78](#)
- INVALID_FORMAT
 - Audio managing (any audio), [44](#)
 - sound4cl_cdef.h, [245](#)
- IsOk
 - sound4::CInstance, [108](#)
 - sound4::CProcessor, [144](#)
 - sound4::helper::CDynLoader, [100](#)
 - sound4::helper::DynFuncHolder< T >, [157](#)
 - sound4::impact::CInstance, [118](#)
 - sound4::impact::dyn::CDynLib, [89](#)
 - sound4::impact::dyn::CInstance, [127](#)
 - sound4::impact::dyn::helper::CDynLoader, [103](#)
 - sound4::impact::dyn::helper::DynFuncHolder< T >, [158](#)
- IsReadOnly
 - MyPresetManager, [161](#), [162](#)
 - sound4::CPresetLoader, [133](#)
 - sound4::impact::CPresetLoader, [135](#)
 - sound4::impact::dyn::CPresetLoader, [137](#)
- JSON syntax, [53](#)
- JSON syntax for internal setup, [56](#)
- Load
 - sound4::CProcessor, [144](#)

- sound4::helper::CDynLoader, 100
 - sound4::impact::dyn::CDynLib, 89
 - sound4::impact::dyn::helper::CDynLoader, 103
- log_cb_t
 - sound4, 72
 - sound4::impact, 75
 - sound4::impact::dyn, 78
- LogSeverity
 - sound4, 72
 - sound4::impact, 75
 - sound4::impact::dyn, 78
- main
 - sound4.impact.cl-pipe-cxx.cpp, 175
 - sound4.impact.cl-pipe-dyn.cpp, 179
 - sound4.impact.cl-pipe.cpp, 185
 - sound4cl-pipe.cpp, 189
- Metadata Management, 37
 - impact_FreeMetadataInfos, 38
 - impact_GetMetadataInfos, 38
 - impact_SetMetadata, 38
 - impact_SetMetadataMulti, 39
- MonoToStereo
 - sound4::CProcessor, 145
 - sound4::impact::dyn::CDynLib, 94
- MonoToStereo_Planar
 - sound4::CProcessor, 145
 - sound4::impact::dyn::CDynLib, 94
- MyLogger
 - sound4.impact.cl-pipe-cxx.cpp, 176
 - sound4.impact.cl-pipe-dyn.cpp, 180
 - sound4.impact.cl-pipe.cpp, 185
- MyPresetManager, 159
 - ~MyPresetManager, 160, 161
 - Exists, 161
 - GetAll, 161
 - IsReadOnly, 161, 162
 - MyPresetManager, 160
 - Read, 162
 - Remove, 162
 - Rename, 162, 163
 - SetupPath, 163
 - Write, 163
- name
 - SOUND4_ProcessInfo, 171
 - sound4cl_SOUND4_ProcessInfo, 172
- NewClient
 - sound4::CInstance, 108
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CDynLib, 94
 - sound4::impact::dyn::CInstance, 127
- NewParameters
 - sound4::impact::dyn::CDynLib, 94
- none
 - Global setup, 23
 - sound4, 73
- OnUpdateThreadStart
 - sound4::CInstance, 108
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CInstance, 127
- OnUpdateThreadStop
 - sound4::CInstance, 108
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CInstance, 128
- operator bool
 - sound4::helper::CDynLoader, 100
 - sound4::helper::DynFuncHolder< T >, 157
 - sound4::impact::dyn::helper::CDynLoader, 103
 - sound4::impact::dyn::helper::DynFuncHolder< T >, 158
- operator T*
 - sound4::helper::DynFuncHolder< T >, 157
 - sound4::impact::dyn::helper::DynFuncHolder< T >, 158
- operator=
 - sound4::CInstance, 108
 - sound4::CInstance::CClient, 83
 - sound4::helper::CDynLoader, 101
 - sound4::impact::CInstance, 118
 - sound4::impact::CInstance::CClient, 84
 - sound4::impact::dyn::CInstance, 128
 - sound4::impact::dyn::CInstance::CClient, 86
 - sound4::impact::dyn::helper::CDynLoader, 103
- OutputChannels
 - sound4::impact, 76
 - sound4::impact::dyn, 79
- OutputSampleSize
 - sound4::impact, 76
 - sound4::impact::dyn, 79
- prefix
 - SOUND4_ProcessInfo, 171
 - sound4cl_SOUND4_ProcessInfo, 172
- PresetChange_Kind
 - sound4, 72
 - sound4::impact, 75
 - sound4::impact::dyn, 78
- PresetManager_InformChange
 - sound4::CInstance, 108
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CDynLib, 95
 - sound4::impact::dyn::CInstance, 128
- Process creation/destruction, 27
 - impact_ExitProcess, 27
 - impact_FreeParameters, 28
 - impact_FreeParameterValue, 28
 - impact_GetParameter, 28
 - impact_GetProcessChunkFrames, 28
 - impact_InitProcess, 29
 - impact_InitProcess2, 29
 - impact_InitProcess3, 29
 - impact_NewParameters, 31
 - impact_SetParameter, 31
 - impact_TerminateProcess, 31
- ProcessAnyAudio
 - sound4::CInstance, 109

- sound4::impact::CInstance, 119
 - sound4::impact::dyn::CInstance, 128
- ProcessAudio
 - sound4::CInstance, 109
 - sound4::impact::CInstance, 119
 - sound4::impact::dyn::CDynLib, 95
 - sound4::impact::dyn::CInstance, 129
- ProcessAudio_Planar
 - sound4::CInstance, 109
 - sound4::impact::CInstance, 119
 - sound4::impact::dyn::CDynLib, 95
 - sound4::impact::dyn::CInstance, 129
- ProcessJson
 - sound4::CInstance::CClient, 83
 - sound4::impact::CInstance::CClient, 84
 - sound4::impact::dyn::CDynLib, 95
 - sound4::impact::dyn::CInstance::CClient, 86
- ProcessLoop
 - sound4.impact.cl-pipe-cxx.cpp, 176
 - sound4.impact.cl-pipe-dyn.cpp, 180
 - sound4.impact.cl-pipe.cpp, 185
 - sound4cl-pipe.cpp, 190
- Read
 - MyPresetManager, 162
 - sound4::CPresetLoader, 133
 - sound4::impact::CPresetLoader, 135
 - sound4::impact::dyn::CPresetLoader, 137
- Remove
 - MyPresetManager, 162
 - sound4::CPresetLoader, 134
 - sound4::impact::CPresetLoader, 135
 - sound4::impact::dyn::CPresetLoader, 137
- Rename
 - MyPresetManager, 162, 163
 - sound4::CPresetLoader, 134
 - sound4::impact::CPresetLoader, 135
 - sound4::impact::dyn::CPresetLoader, 137
- S16_BE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S16_LE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S16_NATIVE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S24_BE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S24_LE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S24_NATIVE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S32_BE
 - Audio managing (any audio), 44
- sound4cl_cdef.h, 245
- S32_LE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- S32_NATIVE
 - Audio managing (any audio), 44
 - sound4cl_cdef.h, 245
- Sample using C interface, 17
- Sample using C++ interface wrapper, 65
- Sample using C++ interface wrapper with dynamic library loading, 66
- Sample using Generic C++ interface (multi-process compatible), 68
- SAMPLE_RATE
 - sound4.impact.cl-pipe.cpp, 185
- SampleFormat
 - sound4::impact, 76
 - sound4::impact::dyn, 78
- SampleRate
 - sound4::impact, 77
 - sound4::impact::dyn, 79
- SaveState
 - sound4::impact::dyn::CDynLib, 95
- SetBus
 - sound4::CInstance, 110
- SetLoggerCallback
 - sound4::CProcessor, 145
 - sound4::impact::dyn::CDynLib, 95
- SetLogSeverity
 - sound4::CProcessor, 145
 - sound4::impact::dyn::CDynLib, 95
- SetMetadata
 - sound4::CInstance, 110
 - sound4::impact::CInstance, 120
 - sound4::impact::dyn::CDynLib, 96
 - sound4::impact::dyn::CInstance, 130
- SetMetadataMulti
 - sound4::CInstance, 111
 - sound4::impact::CInstance, 120
 - sound4::impact::dyn::CDynLib, 96
 - sound4::impact::dyn::CInstance, 130
- SetParam
 - sound4::CInstance, 111
 - sound4::impact::CInstance, 121
 - sound4::impact::dyn::CInstance, 130
- SetParameter
 - sound4::impact::dyn::CDynLib, 96
- SetPresetManager
 - sound4::CInstance, 111
 - sound4::impact::CInstance, 121
 - sound4::impact::dyn::CDynLib, 96
 - sound4::impact::dyn::CInstance, 131
- SetupPath
 - MyPresetManager, 163
- SetWebServerAppHealth
 - sound4::CInstance, 112
 - sound4::impact::CInstance, 121
 - sound4::impact::dyn::CInstance, 131

- shortname
 - SOUND4_ProcessInfo, 171
 - sound4cl_SOUND4_ProcessInfo, 172
- sound4, 71
 - error, 72
 - fatal, 72
 - info, 73
 - log_cb_t, 72
 - LogSeverity, 72
 - none, 73
 - PresetChange_Kind, 72
 - verbose, 73
 - verbose2, 73
 - verbose3, 73
 - verbose4, 73
 - verbose5, 73
 - warning, 73
- SOUND4 IMPACT.CL Library, 1
- sound4.impact.cl-pipe-cxx.cpp, 175, 177
 - GetEnv, 175
 - main, 175
 - MyLogger, 176
 - ProcessLoop, 176
 - Usage, 176
- sound4.impact.cl-pipe-dyn.cpp, 179, 181
 - GetEnv, 179
 - main, 179
 - MyLogger, 180
 - ProcessLoop, 180
 - Usage, 180
- sound4.impact.cl-pipe.cpp, 184, 186
 - CHANNEL_COUNT, 184
 - CHUNK_SIZE, 184
 - main, 185
 - MyLogger, 185
 - ProcessLoop, 185
 - SAMPLE_RATE, 185
 - Usage, 185
- sound4.impact.cl.h, 194, 197
 - IMPACT_DECL, 196
 - IMPACT_HAS_CLOUDBUS, 196
 - IMPACT_HAS_WEBSERVER, 197
- sound4.impact.cl.hpp, 201, 202
 - fs_strdup, 202
- sound4.impact.cl_dyn.hpp, 210, 212
 - fs_strdup, 211
 - UNIXLIKE, 211
- sound4::CBus, 81
 - ~CBus, 81
 - CBus, 81
 - Get, 82
- sound4::CInstance, 103
 - ~CInstance, 105
 - AddAudio, 105
 - CInstance, 105
 - Create, 105
 - GetAudio, 106
 - GetBufferIn, 106
 - GetBufferOut, 106
 - GetChunkFrames, 106
 - GetEstimatedDelay, 107
 - GetMetadataInfos, 107
 - GetParam, 107
 - GetWebServerAppHealth, 107
 - IsOk, 108
 - NewClient, 108
 - OnUpdateThreadStart, 108
 - OnUpdateThreadStop, 108
 - operator=, 108
 - PresetManager_InformChange, 108
 - ProcessAnyAudio, 109
 - ProcessAudio, 109
 - ProcessAudio_Planar, 109
 - SetBus, 110
 - SetMetadata, 110
 - SetMetadataMulti, 111
 - SetParam, 111
 - SetPresetManager, 111
 - SetWebServerAppHealth, 112
 - StartWebServer, 112
 - Stop, 112
 - StopWebServer, 112
 - Terminate, 113
- sound4::CInstance::CClient, 82
 - ~CClient, 83
 - CClient, 82
 - operator=, 83
 - ProcessJson, 83
- sound4::CPresetLoader, 132
 - ~CPresetLoader, 133
 - CPresetLoader, 133
 - Exists, 133
 - GetAll, 133
 - IsReadOnly, 133
 - Read, 133
 - Remove, 134
 - Rename, 134
 - Write, 134
- sound4::CProcessor, 138
 - AudioConvertFrom, 140
 - AudioConvertTo, 140
 - AudioMonoFromLiveStereo, 141
 - AudioMonoToLiveStereo, 141
 - CBus, 146
 - CInstance, 146
 - CProcessor, 140
 - fnAddAudio, 146
 - fnAddPadAudio, 146
 - fnAudioConvertFrom, 146
 - fnAudioConvertTo, 147
 - fnAudioMonoFromLiveStereo, 147
 - fnAudioMonoToLiveStereo, 147
 - fnDeleteClient, 147
 - fnExitProcess, 147
 - fnFreeBus, 147
 - fnFreeJsonAnswer, 147

- fnFreeMetadataInfos, 148
- fnFreeParameters, 148
- fnFreeParameterValue, 148
- fnGetAudio, 148
- fnGetAudioInputCount, 148
- fnGetAudioOutputCount, 148
- fnGetBufferIn, 148
- fnGetBufferOut, 149
- fnGetBytesFromFormat, 149
- fnGetChannelCount, 149
- fnGetChunkSizeInFrames, 149
- fnGetEstimatedDelay, 149
- fnGetFormatFromName, 149
- fnGetFormatName, 149
- fnGetMaxPacketFrame, 150
- fnGetMetadataInfos, 150
- fnGetOutputCount, 150
- fnGetParameter, 150
- fnGetPossibleChunkSizeInFrames, 150
- fnGetProcessChunkFrames, 150
- fnGetSampleRate, 150
- fnGetVersion, 151
- fnInitProcess, 151
- fnInitProcess2, 151
- fnInitProcess3, 151
- fnMonoToStereo, 151
- fnMonoToStereo_Planar, 151
- fnNewBus, 151
- fnNewClient, 152
- fnNewParameters, 152
- fnPresetManager_InformChange, 152
- fnProcessAudio, 152
- fnProcessAudio_Planar, 152
- fnProcessJson, 152
- fnSaveState, 152
- fnSetInstanceBus, 153
- fnSetLoggerCallback, 153
- fnSetLogSeverity, 153
- fnSetMetadata, 153
- fnSetMetadataMulti, 153
- fnSetParameter, 153
- fnSetPresetManager, 153
- fnStartUpdateThread, 154
- fnStereoToMono, 154
- fnStereoToMono_Planar, 154
- fnStopUpdateThread, 154
- fnTerminateProcess, 154
- fnWaitUpdateThreadReady, 154
- fnWebserver, 154
- fnWebserver_FreeString, 155
- fnWebserver_GetAppHealth, 155
- fnWebserver_SetAppHealth, 155
- fnWebserver_Status, 155
- fnWebserver_Stop, 155
- fnWebserver_tcp, 155
- fnWebserver_tcp2, 155
- GetAudioInputCount, 141
- GetAudioOutputCount, 141
- GetBytesFromFormat, 141
- GetChannelCount, 142
- GetChunkSizeInFrames, 142
- GetFormatFromName, 142
- GetFormatName, 143
- GetPossibleChunkSizeInFrames, 143
- GetPrefixSymbol, 143
- GetSampleRate, 144
- GetVersion, 144
- IsOk, 144
- Load, 144
- MonoToStereo, 145
- MonoToStereo_Planar, 145
- SetLoggerCallback, 145
- SetLogSeverity, 145
- StereoToMono, 145
- StereoToMono_Planar, 146
- sound4::helper, 74
 - DynFunc_t, 74
 - DynLib_t, 74
- sound4::helper::CDynLoader, 98
 - ~CDynLoader, 99
 - CDynLoader, 99
 - Close, 100
 - GetSymbol, 100
 - GetSymbol_safe, 100
 - GetThisLibraryPath, 100
 - IsOk, 100
 - Load, 100
 - operator bool, 100
 - operator=, 101
- sound4::helper::DynFuncHolder< T >, 156
 - DynFuncHolder, 156
 - IsOk, 157
 - operator bool, 157
 - operator T*, 157
- sound4::helper::SampleFormat< float >, 165
 - format, 166
- sound4::helper::SampleFormat< int16_t >, 167
 - format, 167
- sound4::helper::SampleFormat< int32_t >, 169
 - format, 169
- sound4::helper::SampleFormat< T >, 164
- sound4::impact, 75
 - ChunkFrames, 76
 - InputChannels, 76
 - InputSampleSize, 76
 - log_cb_t, 75
 - LogSeverity, 75
 - OutputChannels, 76
 - OutputSampleSize, 76
 - PresetChange_Kind, 75
 - SampleFormat, 76
 - SampleRate, 77
- sound4::impact::CInstance, 113
 - ~CInstance, 115
 - AddAudio, 115
 - CInstance, 115

- Create, [115](#)
- GetAudio, [116](#)
- GetBufferIn, [116](#)
- GetBufferOut, [116](#)
- GetChunkFrames, [116](#)
- GetEstimatedDelay, [117](#)
- GetMetadataInfos, [117](#)
- GetParam, [117](#)
- GetWebServerAppHealth, [117](#)
- IsOk, [118](#)
- NewClient, [118](#)
- OnUpdateThreadStart, [118](#)
- OnUpdateThreadStop, [118](#)
- operator=, [118](#)
- PresetManager_InformChange, [118](#)
- ProcessAnyAudio, [119](#)
- ProcessAudio, [119](#)
- ProcessAudio_Planar, [119](#)
- SetMetadata, [120](#)
- SetMetadataMulti, [120](#)
- SetParam, [121](#)
- SetPresetManager, [121](#)
- SetWebServerAppHealth, [121](#)
- StartWebServer, [122](#)
- Stop, [122](#)
- StopWebServer, [122](#)
- Terminate, [122](#)
- sound4::impact::CInstance::CClient, [84](#)
 - ~CClient, [84](#)
 - CClient, [84](#)
 - operator=, [84](#)
 - ProcessJson, [84](#)
- sound4::impact::CPresetLoader, [134](#)
 - ~CPresetLoader, [135](#)
 - CPresetLoader, [135](#)
 - Exists, [135](#)
 - GetAll, [135](#)
 - IsReadOnly, [135](#)
 - Read, [135](#)
 - Remove, [135](#)
 - Rename, [135](#)
 - Write, [136](#)
- sound4::impact::dyn, [77](#)
 - ChunkFrames, [78](#)
 - InputChannels, [78](#)
 - InputSampleSize, [78](#)
 - log_cb_t, [78](#)
 - LogSeverity, [78](#)
 - OutputChannels, [79](#)
 - OutputSampleSize, [79](#)
 - PresetChange_Kind, [78](#)
 - SampleFormat, [78](#)
 - SampleRate, [79](#)
- sound4::impact::dyn::CDynLib, [87](#)
 - AddAudio, [89](#)
 - AddPadAudio, [89](#)
 - AudioConvertFrom, [89](#)
 - AudioConvertTo, [90](#)
 - AudioMonoFromLiveStereo, [90](#)
 - AudioMonoToLiveStereo, [90](#)
 - CDynLib, [88](#)
 - DeleteClient, [90](#)
 - ExitProcess, [90](#)
 - FreeJsonAnswer, [90](#)
 - FreeMetadataInfos, [90](#)
 - FreeParameters, [91](#)
 - FreeParameterValue, [91](#)
 - GetAudio, [91](#)
 - GetAudioInputCount, [91](#)
 - GetAudioOutputCount, [91](#)
 - GetBufferIn, [91](#)
 - GetBufferOut, [91](#)
 - GetBytesFromFormat, [92](#)
 - GetChannelCount, [92](#)
 - GetChunkSizeInFrames, [92](#)
 - GetEstimatedDelay, [92](#)
 - GetFormatFromName, [92](#)
 - GetFormatName, [92](#)
 - GetMaxPacketFrame, [92](#)
 - GetMetadataInfos, [93](#)
 - GetOutputCount, [93](#)
 - GetParameter, [93](#)
 - GetPossibleChunkSizeInFrames, [93](#)
 - GetProcessChunkFrames, [93](#)
 - GetSampleRate, [93](#)
 - GetVersion, [93](#)
 - InitProcess, [94](#)
 - InitProcess2, [94](#)
 - InitProcess3, [94](#)
 - IsOk, [89](#)
 - Load, [89](#)
 - MonoToStereo, [94](#)
 - MonoToStereo_Planar, [94](#)
 - NewClient, [94](#)
 - NewParameters, [94](#)
 - PresetManager_InformChange, [95](#)
 - ProcessAudio, [95](#)
 - ProcessAudio_Planar, [95](#)
 - ProcessJson, [95](#)
 - SaveState, [95](#)
 - SetLoggerCallback, [95](#)
 - SetLogSeverity, [95](#)
 - SetMetadata, [96](#)
 - SetMetadataMulti, [96](#)
 - SetParameter, [96](#)
 - SetPresetManager, [96](#)
 - StartUpdateThread, [96](#)
 - StereoToMono, [96](#)
 - StereoToMono_Planar, [96](#)
 - StopUpdateThread, [97](#)
 - TerminateProcess, [97](#)
 - WaitUpdateThreadReady, [97](#)
 - Webserver, [97](#)
 - Webserver_FreeString, [97](#)
 - Webserver_GetAppHealth, [97](#)
 - Webserver_SetAppHealth, [97](#)

- Webserver_Status, 98
- Webserver_Stop, 98
- Webserver_tcp, 98
- Webserver_tcp2, 98
- sound4::impact::dyn::CInstance, 123
 - ~CInstance, 124
 - AddAudio, 124
 - CInstance, 124
 - Create, 124
 - GetAudio, 125
 - GetBufferIn, 125
 - GetBufferOut, 125
 - GetChunkFrames, 126
 - GetEstimatedDelay, 126
 - GetMetadataInfos, 126
 - GetParam, 126
 - GetWebServerAppHealth, 127
 - IsOk, 127
 - NewClient, 127
 - OnUpdateThreadStart, 127
 - OnUpdateThreadStop, 128
 - operator=, 128
 - PresetManager_InformChange, 128
 - ProcessAnyAudio, 128
 - ProcessAudio, 129
 - ProcessAudio_Planar, 129
 - SetMetadata, 130
 - SetMetadataMulti, 130
 - SetParam, 130
 - SetPresetManager, 131
 - SetWebServerAppHealth, 131
 - StartWebServer, 131
 - Stop, 131
 - StopWebServer, 132
 - Terminate, 132
- sound4::impact::dyn::CInstance::CClient, 85
 - ~CClient, 86
 - CClient, 86
 - operator=, 86
 - ProcessJson, 86
- sound4::impact::dyn::CPresetLoader, 136
 - ~CPresetLoader, 137
 - CPresetLoader, 137
 - Exists, 137
 - GetAll, 137
 - IsReadOnly, 137
 - Read, 137
 - Remove, 137
 - Rename, 137
 - Write, 138
- sound4::impact::dyn::helper, 79
 - DynFunc_t, 80
 - DynLib_t, 80
- sound4::impact::dyn::helper::CDynLoader, 101
 - ~CDynLoader, 102
 - CDynLoader, 102
 - Close, 102
 - GetSymbol, 102
 - GetSymbol_safe, 102
 - GetThisLibraryPath, 102
 - IsOk, 103
 - Load, 103
 - operator bool, 103
 - operator=, 103
- sound4::impact::dyn::helper::DynFuncHolder< T >, 157
 - DynFuncHolder, 158
 - IsOk, 158
 - operator bool, 158
 - operator T*, 158
- sound4::impact::dyn::helper::SampleFormat< float >, 166
 - format, 166
- sound4::impact::dyn::helper::SampleFormat< int16_t >, 167
 - format, 168
- sound4::impact::dyn::helper::SampleFormat< int32_t >, 169
 - format, 170
- sound4::impact::dyn::helper::SampleFormat< T >, 164
- sound4::impact::helper, 80
- sound4::impact::helper::SampleFormat< float >, 166
 - format, 167
- sound4::impact::helper::SampleFormat< int16_t >, 168
 - format, 168
- sound4::impact::helper::SampleFormat< int32_t >, 170
 - format, 170
- sound4::impact::helper::SampleFormat< T >, 165
- SOUND4_GetProcessInfo
 - Global setup, 27
- SOUND4_INVALID_WEBSERVER_ID
 - sound4cl_cdef.h, 242
 - Web server, 60
- SOUND4_ProcessInfo, 170
 - name, 171
 - prefix, 171
 - shortname, 171
 - struct_size, 171
 - version, 171
- SOUND4_WEBSERVER_HTTP_OK
 - sound4cl_cdef.h, 242
 - Web server, 60
- SOUND4_WEBSERVER_HTTPS_OK
 - sound4cl_cdef.h, 243
 - Web server, 60
- sound4cl-pipe.cpp, 189, 191
 - GetEnv, 189
 - main, 189
 - ProcessLoop, 190
- sound4cl.hpp, 225, 227
 - UNIXLIKE, 226
- sound4cl_AddAudio
 - sound4cl_cdef.h, 245
- sound4cl_AddPadAudio
 - sound4cl_cdef.h, 245
- sound4cl_AudioConvertFrom

- sound4cl_cdef.h, 246
- sound4cl_AudioConvertTo
 - sound4cl_cdef.h, 246
- sound4cl_AudioMonoFromLiveStereo
 - sound4cl_cdef.h, 246
- sound4cl_AudioMonoToLiveStereo
 - sound4cl_cdef.h, 246
- sound4cl_cdef.h, 239, 254
 - change_kind_created, 245
 - change_kind_deleted, 245
 - change_kind_modified, 245
 - F32_BE, 245
 - F32_LE, 245
 - F32_NATIVE, 245
 - fs_char, 243
 - fs_strdup, 242
 - INVALID_FORMAT, 245
 - S16_BE, 245
 - S16_LE, 245
 - S16_NATIVE, 245
 - S24_BE, 245
 - S24_LE, 245
 - S24_NATIVE, 245
 - S32_BE, 245
 - S32_LE, 245
 - S32_NATIVE, 245
 - SOUND4_INVALID_WEBSERVER_ID, 242
 - SOUND4_WEBSERVER_HTTP_OK, 242
 - SOUND4_WEBSERVER_HTTPS_OK, 243
- sound4cl_AddAudio, 245
- sound4cl_AddPadAudio, 245
- sound4cl_AudioConvertFrom, 246
- sound4cl_AudioConvertTo, 246
- sound4cl_AudioMonoFromLiveStereo, 246
- sound4cl_AudioMonoToLiveStereo, 246
- sound4cl_DeleteClient, 246
- sound4cl_error, 244
- sound4cl_ExitProcess, 246
- sound4cl_fatal, 244
- sound4cl_FreeBus, 246
- sound4cl_FreeJsonAnswer, 246
- sound4cl_FreeMetadataInfos, 247
- sound4cl_FreeParameters, 247
- sound4cl_FreeParameterValue, 247
- sound4cl_GetAudio, 247
- sound4cl_GetAudioInputCount, 247
- sound4cl_GetAudioOutputCount, 247
- sound4cl_GetBufferIn, 247
- sound4cl_GetBufferOut, 247
- sound4cl_GetBytesFromFormat, 248
- sound4cl_GetChannelCount, 248
- sound4cl_GetChunkSizeInFrames, 248
- sound4cl_GetEstimatedDelay, 248
- sound4cl_GetFormatFromName, 248
- sound4cl_GetFormatName, 248
- sound4cl_GetMaxPacketFrame, 248
- sound4cl_GetMetadataInfos, 248
- sound4cl_GetOutputCount, 248
- sound4cl_GetParameter, 249
- sound4cl_GetPossibleChunkSizeInFrames, 249
- sound4cl_GetProcessChunkFrames, 249
- sound4cl_GetSampleRate, 249
- sound4cl_GetVersion, 249
- sound4cl_info, 244
- sound4cl_InitProcess, 249
- sound4cl_InitProcess2, 249
- sound4cl_InitProcess3, 249
- sound4cl_loggerfn, 243
- sound4cl_LogSeverity, 244
- sound4cl_MonoToStereo, 250
- sound4cl_MonoToStereo_Planar, 250
- sound4cl_NewBus, 250
- sound4cl_NewClient, 250
- sound4cl_NewParameters, 250
- sound4cl_none, 244
- sound4cl_PresetChange_Kind, 245
- sound4cl_PresetManager_InformChange, 250
- sound4cl_ProcessAudio, 250
- sound4cl_ProcessAudio_Planar, 251
- sound4cl_ProcessJson, 251
- sound4cl_SampleFormat, 245
- sound4cl_SaveState, 251
- sound4cl_SetInstanceBus, 251
- sound4cl_SetLoggerCallback, 251
- sound4cl_SetLogSeverity, 251
- sound4cl_SetMetadata, 251
- sound4cl_SetMetadataMulti, 251
- sound4cl_SetParameter, 252
- sound4cl_SetPresetManager, 252
- sound4cl_SOUND4_GetProcessInfo, 252
- sound4cl_StartUpdateThread, 252
- sound4cl_StereoToMono, 252
- sound4cl_StereoToMono_Planar, 252
- sound4cl_StopUpdateThread, 253
- sound4cl_storage_exists, 243
- sound4cl_storage_getall, 243
- sound4cl_storage_getall_free, 243
- sound4cl_storage_reader, 243
- sound4cl_storage_reader_free, 244
- sound4cl_storage_remove, 244
- sound4cl_storage_rename, 244
- sound4cl_storage_writer, 244
- sound4cl_TerminateProcess, 253
- sound4cl_verbose, 244
- sound4cl_verbose2, 244
- sound4cl_verbose3, 244
- sound4cl_verbose4, 244
- sound4cl_verbose5, 244
- sound4cl_WaitUpdateThreadReady, 253
- sound4cl_warning, 244
- sound4cl_Webserver, 253
- sound4cl_Webserver_FreeString, 253
- sound4cl_Webserver_GetAppHealth, 253
- sound4cl_Webserver_SetAppHealth, 253
- sound4cl_Webserver_Status, 254
- sound4cl_Webserver_Stop, 254

- sound4cl_Webserver_tcp, [254](#)
 - sound4cl_Webserver_tcp2, [254](#)
- sound4cl_DeleteClient
 - sound4cl_cdef.h, [246](#)
- sound4cl_error
 - sound4cl_cdef.h, [244](#)
- sound4cl_ExitProcess
 - sound4cl_cdef.h, [246](#)
- sound4cl_fatal
 - sound4cl_cdef.h, [244](#)
- sound4cl_FreeBus
 - sound4cl_cdef.h, [246](#)
- sound4cl_FreeJsonAnswer
 - sound4cl_cdef.h, [246](#)
- sound4cl_FreeMetadataInfos
 - sound4cl_cdef.h, [247](#)
- sound4cl_FreeParameters
 - sound4cl_cdef.h, [247](#)
- sound4cl_FreeParameterValue
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetAudio
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetAudioInputCount
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetAudioOutputCount
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetBufferIn
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetBufferOut
 - sound4cl_cdef.h, [247](#)
- sound4cl_GetBytesFromFormat
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetChannelCount
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetChunkSizeInFrames
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetEstimatedDelay
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetFormatFromName
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetFormatName
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetMaxPacketFrame
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetMetadataInfos
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetOutputCount
 - sound4cl_cdef.h, [248](#)
- sound4cl_GetParameter
 - sound4cl_cdef.h, [249](#)
- sound4cl_GetPossibleChunkSizeInFrames
 - sound4cl_cdef.h, [249](#)
- sound4cl_GetProcessChunkFrames
 - sound4cl_cdef.h, [249](#)
- sound4cl_GetSampleRate
 - sound4cl_cdef.h, [249](#)
- sound4cl_GetVersion
 - sound4cl_cdef.h, [249](#)
- sound4cl_info
 - sound4cl_cdef.h, [244](#)
- sound4cl_InitProcess
 - sound4cl_cdef.h, [249](#)
- sound4cl_InitProcess2
 - sound4cl_cdef.h, [249](#)
- sound4cl_InitProcess3
 - sound4cl_cdef.h, [249](#)
- sound4cl_loggerfn
 - sound4cl_cdef.h, [243](#)
- sound4cl_LogSeverity
 - sound4cl_cdef.h, [244](#)
- sound4cl_MonoToStereo
 - sound4cl_cdef.h, [250](#)
- sound4cl_MonoToStereo_Planar
 - sound4cl_cdef.h, [250](#)
- sound4cl_NewBus
 - sound4cl_cdef.h, [250](#)
- sound4cl_NewClient
 - sound4cl_cdef.h, [250](#)
- sound4cl_NewParameters
 - sound4cl_cdef.h, [250](#)
- sound4cl_none
 - sound4cl_cdef.h, [244](#)
- sound4cl_PresetChange_Kind
 - sound4cl_cdef.h, [245](#)
- sound4cl_PresetManager_InformChange
 - sound4cl_cdef.h, [250](#)
- sound4cl_ProcessAudio
 - sound4cl_cdef.h, [250](#)
- sound4cl_ProcessAudio_Planar
 - sound4cl_cdef.h, [251](#)
- sound4cl_ProcessJson
 - sound4cl_cdef.h, [251](#)
- sound4cl_SampleFormat
 - sound4cl_cdef.h, [245](#)
- sound4cl_SaveState
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetInstanceBus
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetLoggerCallback
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetLogSeverity
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetMetadata
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetMetadataMulti
 - sound4cl_cdef.h, [251](#)
- sound4cl_SetParameter
 - sound4cl_cdef.h, [252](#)
- sound4cl_SetPresetManager
 - sound4cl_cdef.h, [252](#)
- sound4cl_SOUND4_GetProcessInfo
 - sound4cl_cdef.h, [252](#)
- sound4cl_SOUND4_ProcessInfo, [172](#)
 - name, [172](#)
 - prefix, [172](#)
 - shortname, [172](#)

- struct_size, [173](#)
- version, [173](#)
- sound4cl_StartUpdateThread
 - sound4cl_cdef.h, [252](#)
- sound4cl_StereoToMono
 - sound4cl_cdef.h, [252](#)
- sound4cl_StereoToMono_Planar
 - sound4cl_cdef.h, [252](#)
- sound4cl_StopUpdateThread
 - sound4cl_cdef.h, [253](#)
- sound4cl_storage_exists
 - sound4cl_cdef.h, [243](#)
- sound4cl_storage_getall
 - sound4cl_cdef.h, [243](#)
- sound4cl_storage_getall_free
 - sound4cl_cdef.h, [243](#)
- sound4cl_storage_reader
 - sound4cl_cdef.h, [243](#)
- sound4cl_storage_reader_free
 - sound4cl_cdef.h, [244](#)
- sound4cl_storage_remove
 - sound4cl_cdef.h, [244](#)
- sound4cl_storage_rename
 - sound4cl_cdef.h, [244](#)
- sound4cl_storage_writer
 - sound4cl_cdef.h, [244](#)
- sound4cl_TerminateProcess
 - sound4cl_cdef.h, [253](#)
- sound4cl_verbose
 - sound4cl_cdef.h, [244](#)
- sound4cl_verbose2
 - sound4cl_cdef.h, [244](#)
- sound4cl_verbose3
 - sound4cl_cdef.h, [244](#)
- sound4cl_verbose4
 - sound4cl_cdef.h, [244](#)
- sound4cl_verbose5
 - sound4cl_cdef.h, [244](#)
- sound4cl_WaitUpdateThreadReady
 - sound4cl_cdef.h, [253](#)
- sound4cl_warning
 - sound4cl_cdef.h, [244](#)
- sound4cl_Webserver
 - sound4cl_cdef.h, [253](#)
- sound4cl_Webserver_FreeString
 - sound4cl_cdef.h, [253](#)
- sound4cl_Webserver_GetAppHealth
 - sound4cl_cdef.h, [253](#)
- sound4cl_Webserver_SetAppHealth
 - sound4cl_cdef.h, [253](#)
- sound4cl_Webserver_Status
 - sound4cl_cdef.h, [254](#)
- sound4cl_Webserver_Stop
 - sound4cl_cdef.h, [254](#)
- sound4cl_Webserver_tcp
 - sound4cl_cdef.h, [254](#)
- sound4cl_Webserver_tcp2
 - sound4cl_cdef.h, [254](#)
- StartUpdateThread
 - sound4::impact::dyn::CDynLib, [96](#)
- StartWebServer
 - sound4::CInstance, [112](#)
 - sound4::impact::CInstance, [122](#)
 - sound4::impact::dyn::CInstance, [131](#)
- std, [80](#)
- StereoToMono
 - sound4::CProcessor, [145](#)
 - sound4::impact::dyn::CDynLib, [96](#)
- StereoToMono_Planar
 - sound4::CProcessor, [146](#)
 - sound4::impact::dyn::CDynLib, [96](#)
- Stop
 - sound4::CInstance, [112](#)
 - sound4::impact::CInstance, [122](#)
 - sound4::impact::dyn::CInstance, [131](#)
- StopUpdateThread
 - sound4::impact::dyn::CDynLib, [97](#)
- StopWebServer
 - sound4::CInstance, [112](#)
 - sound4::impact::CInstance, [122](#)
 - sound4::impact::dyn::CInstance, [132](#)
- struct_size
 - SOUND4_ProcessInfo, [171](#)
 - sound4cl_SOUND4_ProcessInfo, [173](#)
- Terminate
 - sound4::CInstance, [113](#)
 - sound4::impact::CInstance, [122](#)
 - sound4::impact::dyn::CInstance, [132](#)
- TerminateProcess
 - sound4::impact::dyn::CDynLib, [97](#)
- UNIXLIKE
 - sound4.impact.cl_dyn.hpp, [211](#)
 - sound4cl.hpp, [226](#)
- Update Management, [39](#)
 - impact_StartUpdateThread, [39](#)
 - impact_StopUpdateThread, [40](#)
 - impact_WaitUpdateThreadReady, [40](#)
- Usage
 - sound4.impact.cl-pipe-cxx.cpp, [176](#)
 - sound4.impact.cl-pipe-dyn.cpp, [180](#)
 - sound4.impact.cl-pipe.cpp, [185](#)
- verbose
 - Global setup, [23](#)
 - sound4, [73](#)
- verbose2
 - Global setup, [23](#)
 - sound4, [73](#)
- verbose3
 - Global setup, [23](#)
 - sound4, [73](#)
- verbose4
 - Global setup, [23](#)
 - sound4, [73](#)
- verbose5

- Global setup, [23](#)
- sound4, [73](#)
- version
 - SOUND4_ProcessInfo, [171](#)
 - sound4cl_SOUND4_ProcessInfo, [173](#)
- WaitUpdateThreadReady
 - sound4::impact::dyn::CDynLib, [97](#)
- warning
 - Global setup, [23](#)
 - sound4, [73](#)
- Web server, [59](#)
 - impact_Webserver, [60](#)
 - impact_Webserver_FreeString, [61](#)
 - impact_Webserver_GetAppHealth, [61](#)
 - impact_Webserver_SetAppHealth, [61](#)
 - impact_Webserver_Status, [63](#)
 - impact_Webserver_Stop, [63](#)
 - impact_Webserver_tcp, [64](#)
 - impact_Webserver_tcp2, [64](#)
 - SOUND4_INVALID_WEBSERVER_ID, [60](#)
 - SOUND4_WEBSERVER_HTTP_OK, [60](#)
 - SOUND4_WEBSERVER_HTTPS_OK, [60](#)
- Webserver
 - sound4::impact::dyn::CDynLib, [97](#)
- Webserver_FreeString
 - sound4::impact::dyn::CDynLib, [97](#)
- Webserver_GetAppHealth
 - sound4::impact::dyn::CDynLib, [97](#)
- Webserver_SetAppHealth
 - sound4::impact::dyn::CDynLib, [97](#)
- Webserver_Status
 - sound4::impact::dyn::CDynLib, [98](#)
- Webserver_Stop
 - sound4::impact::dyn::CDynLib, [98](#)
- Webserver_tcp
 - sound4::impact::dyn::CDynLib, [98](#)
- Webserver_tcp2
 - sound4::impact::dyn::CDynLib, [98](#)
- Write
 - MyPresetManager, [163](#)
 - sound4::CPresetLoader, [134](#)
 - sound4::impact::CPresetLoader, [136](#)
 - sound4::impact::dyn::CPresetLoader, [138](#)