

SOUND4 IMPACT.CL Library [1.3.2]

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	21
8.3.2.4 IMPACT_CHANNEL_COUNT	21
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()	23
8.3.5.4 impact_GetChunkSizeInFrames()	24
8.3.5.5 impact_GetPossibleChunkSizeInFrames()	24
8.3.5.6 impact_GetSampleRate()	24
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()	25
8.4 Process creation/destruction	26
8.4.1 Detailed Description	26
8.4.2 Function Documentation	26
8.4.2.1 impact_ExitProcess()	26
8.4.2.2 impact_FreeParameters()	27
8.4.2.3 impact_FreeParameterValue()	27
8.4.2.4 impact_GetParameter()	27
8.4.2.5 impact_GetProcessChunkFrames()	27
8.4.2.6 impact_InitProcess()	28
8.4.2.7 impact_InitProcess2()	28
8.4.2.8 impact_InitProcess3()	28
8.4.2.9 impact_NewParameters()	30
8.4.2.10 impact_SetParameter()	30
8.4.2.11 impact_TerminateProcess()	30
8.5 Custom preset management	30
8.5.1 Detailed Description	31
8.5.2 Typedef Documentation	32
8.5.2.1 fs_char	32
8.5.2.2 impact_storage_exists	32
8.5.2.3 impact_storage_getall	32
8.5.2.4 impact_storage_getall_free	32
8.5.2.5 impact_storage_reader	33
8.5.2.6 impact_storage_reader_free	33
8.5.2.7 impact_storage_remove	33
8.5.2.8 impact_storage_rename	34
8.5.2.9 impact_storage_writer	34
8.5.3 Enumeration Type Documentation	34
8.5.3.1 impact_PresetChange_Kind	34
8.5.4 Function Documentation	35
8.5.4.1 impact_PresetManager_InformChange()	35
8.5.4.2 impact_SetPresetManager()	35

8.6 Metadata Management	36
8.6.1 Detailed Description	36
8.6.2 Function Documentation	36
8.6.2.1 impact_FreeMetadataInfos()	36
8.6.2.2 impact_GetMetadataInfos()	36
8.6.2.3 impact_SetMetadata()	37
8.6.2.4 impact_SetMetadataMulti()	37
8.7 Update Management	38
8.7.1 Detailed Description	38
8.7.2 Function Documentation	38
8.7.2.1 impact_StartUpdateThread()	38
8.7.2.2 impact_StopUpdateThread()	38
8.7.2.3 impact_WaitUpdateThreadReady()	39
8.8 Direct processing	39
8.8.1 Detailed Description	39
8.8.2 Function Documentation	39
8.8.2.1 impact_GetBufferIn()	39
8.8.2.2 impact_GetBufferOut()	40
8.8.2.3 impact_GetEstimatedDelay()	40
8.8.2.4 impact_ProcessAudio()	41
8.8.2.5 impact_ProcessAudio_Planar()	41
8.9 Audio managing (any audio)	42
8.9.1 Detailed Description	42
8.9.2 Enumeration Type Documentation	42
8.9.2.1 impact_SampleFormat	42
8.9.3 Function Documentation	43
8.9.3.1 impact_AddAudio()	43
8.9.3.2 impact_AddPadAudio()	43
8.9.3.3 impact_GetAudio()	44
8.9.3.4 impact_GetBytesFromFormat()	44
8.9.3.5 impact_GetFormatFromName()	44
8.9.3.6 impact_GetFormatName()	45
8.9.3.7 impact_GetMaxPacketFrame()	45
8.9.3.8 impact_GetOutputCount()	45
8.10 Audio convert helpers	46
8.10.1 Detailed Description	46
8.10.2 Function Documentation	46
8.10.2.1 impact_AudioConvertFrom()	46
8.10.2.2 impact_AudioConvertTo()	47
8.10.2.3 impact_AudioMonoFromLiveStereo()	47
8.10.2.4 impact_AudioMonoToLiveStereo()	47
8.10.2.5 impact_MonoToStereo()	48

8.10.2.6 impact_MonoToStereo_Planar()	48
8.10.2.7 impact_StereoToMono()	48
8.10.2.8 impact_StereoToMono_Planar()	49
8.11 Client's JSon config interfacing	49
8.11.1 Detailed Description	50
8.11.2 Function Documentation	50
8.11.2.1 impact_DeleteClient()	50
8.11.2.2 impact_FreeJsonAnswer()	50
8.11.2.3 impact_NewClient()	50
8.11.2.4 impact_ProcessJson()	51
8.11.2.5 impact_SaveState()	51
8.12 JSON syntax	52
8.12.1 Commands	52
8.12.2 Answers	53
8.12.3 Examples	53
8.12.3.1 Get status	53
8.12.3.2 Get all changes since last request	53
8.12.3.3 Check license	53
8.12.3.4 Set Processor name	53
8.12.3.5 Get presetlist	53
8.12.3.6 Load a preset	54
8.12.3.7 Get information on preset on air	54
8.12.3.8 Set input gain	54
8.12.3.9 Set bypass	54
8.12.3.10 Export a preset	54
8.12.3.11 Import a preset	54
8.13 JSON syntax for internal setup	55
8.13.1 Get process and modules information	55
8.13.2 Authentication management	55
8.13.2.1 Difference between setup and admin account	55
8.13.2.2 Access encoding	55
8.13.2.3 Change setup account	56
8.13.2.4 Get user info	56
8.13.2.5 Add or modify a user	56
8.13.2.6 Removes a user	56
8.13.2.7 List users	56
8.13.2.8 Get/set unlogged access	56
8.13.2.9 Get realm	57
8.13.2.10 Get API version	57
8.13.2.11 Get setup is admin	57
8.13.2.12 Get/set setup account	57
8.13.2.13 Get/set custom storage	57

8.13.3 Get process' interface description	57
8.13.3.1 Get XML decription	57
8.13.3.2 Get Process Image	57
8.13.3.3 Get Process Ico	58
8.14 Web server	58
8.14.1 Detailed Description	58
8.14.2 Macro Definition Documentation	58
8.14.2.1 SOUND4_INVALID_WEBSERVER_ID	58
8.14.2.2 SOUND4_WEBSERVER_HTTP_OK	59
8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK	59
8.14.3 Function Documentation	59
8.14.3.1 impact_Webserver()	59
8.14.3.2 impact_Webserver_FreeString()	60
8.14.3.3 impact_Webserver_GetAppHealth()	60
8.14.3.4 impact_Webserver_SetAppHealth()	60
8.14.3.5 impact_Webserver_Status()	61
8.14.3.6 impact_Webserver_Stop()	61
8.14.3.7 impact_Webserver_tcp()	61
8.14.3.8 impact_Webserver_tcp2()	62
8.15 Sample using C++ interface wrapper	63
8.15.1 Setup and sanity checks	63
8.15.2 Sample to create the processing instance	63
8.15.3 Communicating with the instance	63
8.15.4 Starting a web-server	63
8.15.5 Sample processing loop	64
8.16 Sample using C++ interface wrapper with dynamic library loading	64
8.16.1 Dynamic loading	64
8.16.2 Setup and sanity checks	64
8.16.3 Sample to create the processing instance	64
8.16.4 Communicating with the instance	64
8.16.5 Starting a web-server	64
8.16.6 Sample processing loop	65
8.16.7 Custom preset manager	65
8.16.8 Custom preset manager loading	66
8.17 Sample using Generic C++ interface (multi-process compatible)	66
8.17.1 Loading the library	66
8.17.2 Setup and configure	66
8.17.3 Create the processing instance	66
8.17.4 Processing loop	67
8.17.5 Starting a web-server	67
8.17.6 Communicating with the instance	67
8.17.7 Custom preset manager example	67

8.17.8 Custom preset manager loading	68
9 Namespace Documentation	69
9.1 dyn Namespace Reference	69
9.1.1 Detailed Description	69
9.2 sound4 Namespace Reference	69
9.2.1 Detailed Description	70
9.2.2 Typedef Documentation	70
9.2.2.1 log_cb_t	70
9.2.2.2 LogSeverity	70
9.2.2.3 PresetChange_Kind	70
9.2.3 Variable Documentation	70
9.2.3.1 error	70
9.2.3.2 fatal	71
9.2.3.3 info	71
9.2.3.4 none	71
9.2.3.5 verbose	71
9.2.3.6 verbose2	71
9.2.3.7 verbose3	71
9.2.3.8 verbose4	71
9.2.3.9 verbose5	71
9.2.3.10 warning	72
9.3 sound4::helper Namespace Reference	72
9.3.1 Detailed Description	72
9.3.2 Typedef Documentation	72
9.3.2.1 DynFunc_t	72
9.3.2.2 DynLib_t	72
9.4 sound4::impact Namespace Reference	73
9.4.1 Detailed Description	73
9.4.2 Typedef Documentation	73
9.4.2.1 log_cb_t	73
9.4.2.2 LogSeverity	73
9.4.2.3 PresetChange_Kind	74
9.4.2.4 SampleFormat	74
9.4.3 Variable Documentation	74
9.4.3.1 ChunkFrames	74
9.4.3.2 InputChannels	74
9.4.3.3 InputSampleSize	74
9.4.3.4 OutputChannels	74
9.4.3.5 OutputSampleSize	75
9.4.3.6 SampleRate	75
9.5 sound4::impact::dyn Namespace Reference	75

9.5.1 Typedef Documentation	76
9.5.1.1 log_cb_t	76
9.5.1.2 LogSeverity	76
9.5.1.3 PresetChange_Kind	76
9.5.1.4 SampleFormat	76
9.5.2 Variable Documentation	76
9.5.2.1 ChunkFrames	76
9.5.2.2 InputChannels	76
9.5.2.3 InputSampleSize	77
9.5.2.4 OutputChannels	77
9.5.2.5 OutputSampleSize	77
9.5.2.6 SampleRate	77
9.6 sound4::impact::dyn::helper Namespace Reference	77
9.6.1 Detailed Description	78
9.6.2 Typedef Documentation	78
9.6.2.1 DynFunc_t	78
9.6.2.2 DynLib_t	78
9.7 sound4::impact::helper Namespace Reference	78
9.7.1 Detailed Description	78
9.8 std Namespace Reference	78
10 Class Documentation	79
10.1 sound4::CBus Class Reference	79
10.1.1 Detailed Description	79
10.1.2 Constructor & Destructor Documentation	79
10.1.2.1 CBus()	79
10.1.2.2 ~CBus()	79
10.1.3 Member Function Documentation	80
10.1.3.1 Get()	80
10.2 sound4::CInstance::CClient Class Reference	80
10.2.1 Detailed Description	80
10.2.2 Constructor & Destructor Documentation	80
10.2.2.1 CClient() [1/2]	80
10.2.2.2 CClient() [2/2]	81
10.2.2.3 ~CClient()	81
10.2.3 Member Function Documentation	81
10.2.3.1 operator=()	81
10.2.3.2 ProcessJson()	81
10.3 sound4::impact::CInstance::CClient Class Reference	82
10.3.1 Detailed Description	82
10.3.2 Constructor & Destructor Documentation	82
10.3.2.1 CClient() [1/2]	82

10.3.2.2 CClient() [2/2]	82
10.3.2.3 ~CClient()	82
10.3.3 Member Function Documentation	82
10.3.3.1 operator=()	82
10.3.3.2 ProcessJson()	82
10.4 sound4::impact::dyn::CInstance::CClient Class Reference	83
10.4.1 Detailed Description	83
10.4.2 Constructor & Destructor Documentation	84
10.4.2.1 CClient() [1/2]	84
10.4.2.2 CClient() [2/2]	84
10.4.2.3 ~CClient()	84
10.4.3 Member Function Documentation	84
10.4.3.1 operator=()	84
10.4.3.2 ProcessJson()	84
10.5 sound4::impact::dyn::CDynLib Class Reference	85
10.5.1 Detailed Description	86
10.5.2 Constructor & Destructor Documentation	86
10.5.2.1 CDynLib()	86
10.5.3 Member Function Documentation	87
10.5.3.1 IsOk()	87
10.5.3.2 Load()	87
10.5.4 Member Data Documentation	87
10.5.4.1 AddAudio	87
10.5.4.2 AddPadAudio	87
10.5.4.3 AudioConvertFrom	88
10.5.4.4 AudioConvertTo	88
10.5.4.5 AudioMonoFromLiveStereo	88
10.5.4.6 AudioMonoToLiveStereo	88
10.5.4.7 DeleteClient	88
10.5.4.8 ExitProcess	88
10.5.4.9 FreeJsonAnswer	88
10.5.4.10 FreeMetadataInfos	89
10.5.4.11 FreeParameters	89
10.5.4.12 FreeParameterValue	89
10.5.4.13 GetAudio	89
10.5.4.14 GetAudioInputCount	89
10.5.4.15 GetAudioOutputCount	89
10.5.4.16 GetBufferIn	89
10.5.4.17 GetBufferOut	90
10.5.4.18 GetBytesFromFormat	90
10.5.4.19 GetChannelCount	90
10.5.4.20 GetChunkSizeInFrames	90

10.5.4.21 GetEstimatedDelay	90
10.5.4.22 GetFormatFromName	90
10.5.4.23 GetFormatName	90
10.5.4.24 GetMaxPacketFrame	91
10.5.4.25 GetMetadataInfos	91
10.5.4.26 GetOutputCount	91
10.5.4.27 GetParameter	91
10.5.4.28 GetPossibleChunkSizeInFrames	91
10.5.4.29 GetProcessChunkFrames	91
10.5.4.30 GetSampleRate	91
10.5.4.31 GetVersion	92
10.5.4.32 InitProcess	92
10.5.4.33 InitProcess2	92
10.5.4.34 InitProcess3	92
10.5.4.35 MonoToStereo	92
10.5.4.36 MonoToStereo_Planar	92
10.5.4.37 NewClient	92
10.5.4.38 NewParameters	93
10.5.4.39 PresetManager_InformChange	93
10.5.4.40 ProcessAudio	93
10.5.4.41 ProcessAudio_Planar	93
10.5.4.42 ProcessJson	93
10.5.4.43 SaveState	93
10.5.4.44 SetLoggerCallback	93
10.5.4.45 SetLogSeverity	94
10.5.4.46 SetMetadata	94
10.5.4.47 SetMetadataMulti	94
10.5.4.48 SetParameter	94
10.5.4.49 SetPresetManager	94
10.5.4.50 StartUpdateThread	94
10.5.4.51 StereoToMono	94
10.5.4.52 StereoToMono_Planar	95
10.5.4.53 StopUpdateThread	95
10.5.4.54 TerminateProcess	95
10.5.4.55 WaitUpdateThreadReady	95
10.5.4.56 Webserver	95
10.5.4.57 Webserver_FreeString	95
10.5.4.58 Webserver_GetAppHealth	95
10.5.4.59 Webserver_SetAppHealth	96
10.5.4.60 Webserver_Status	96
10.5.4.61 Webserver_Stop	96
10.5.4.62 Webserver_tcp	96

10.5.4.63 Webserver_tcp2	96
10.6 sound4::helper::CDynLoader Class Reference	96
10.6.1 Detailed Description	97
10.6.2 Constructor & Destructor Documentation	97
10.6.2.1 CDynLoader() [1/3]	97
10.6.2.2 CDynLoader() [2/3]	97
10.6.2.3 CDynLoader() [3/3]	97
10.6.2.4 ~CDynLoader()	97
10.6.3 Member Function Documentation	98
10.6.3.1 Close()	98
10.6.3.2 GetSymbol()	98
10.6.3.3 GetSymbol_safe()	98
10.6.3.4 GetThisLibraryPath()	98
10.6.3.5 IsOk()	98
10.6.3.6 Load()	98
10.6.3.7 operator bool()	99
10.6.3.8 operator=()	99
10.7 sound4::impact::dyn::helper::CDynLoader Class Reference	99
10.7.1 Detailed Description	99
10.7.2 Constructor & Destructor Documentation	100
10.7.2.1 CDynLoader() [1/3]	100
10.7.2.2 CDynLoader() [2/3]	100
10.7.2.3 CDynLoader() [3/3]	100
10.7.2.4 ~CDynLoader()	100
10.7.3 Member Function Documentation	100
10.7.3.1 Close()	100
10.7.3.2 GetSymbol()	100
10.7.3.3 GetSymbol_safe()	100
10.7.3.4 GetThisLibraryPath()	101
10.7.3.5 IsOk()	101
10.7.3.6 Load()	101
10.7.3.7 operator bool()	101
10.7.3.8 operator=()	101
10.8 sound4::CInstance Class Reference	101
10.8.1 Detailed Description	102
10.8.2 Constructor & Destructor Documentation	103
10.8.2.1 CInstance() [1/2]	103
10.8.2.2 CInstance() [2/2]	103
10.8.2.3 ~CInstance()	103
10.8.3 Member Function Documentation	103
10.8.3.1 AddAudio()	103
10.8.3.2 Create()	103

10.8.3.3 GetAudio()	104
10.8.3.4 GetBufferIn()	104
10.8.3.5 GetBufferOut()	104
10.8.3.6 GetChunkFrames()	105
10.8.3.7 GetEstimatedDelay()	105
10.8.3.8 GetMetadataInfos()	105
10.8.3.9 GetParam()	105
10.8.3.10 GetWebServerAppHealth()	106
10.8.3.11 IsOk()	106
10.8.3.12 NewClient()	106
10.8.3.13 OnUpdateThreadStart()	106
10.8.3.14 OnUpdateThreadStop()	106
10.8.3.15 operator=()	106
10.8.3.16 PresetManager_InformChange()	106
10.8.3.17 ProcessAnyAudio()	107
10.8.3.18 ProcessAudio()	107
10.8.3.19 ProcessAudio_Planar()	108
10.8.3.20 SetBus()	108
10.8.3.21 SetMetadata()	108
10.8.3.22 SetMetadataMulti()	109
10.8.3.23 SetParam()	109
10.8.3.24 SetPresetManager()	109
10.8.3.25 SetWebServerAppHealth()	110
10.8.3.26 StartWebServer()	110
10.8.3.27 Stop()	110
10.8.3.28 StopWebServer()	111
10.8.3.29 Terminate()	111
10.9 sound4::impact::CInstance Class Reference	111
10.9.1 Detailed Description	112
10.9.2 Constructor & Destructor Documentation	113
10.9.2.1 CInstance() [1/2]	113
10.9.2.2 CInstance() [2/2]	113
10.9.2.3 ~CInstance()	113
10.9.3 Member Function Documentation	113
10.9.3.1 AddAudio()	113
10.9.3.2 Create()	113
10.9.3.3 GetAudio()	114
10.9.3.4 GetBufferIn()	114
10.9.3.5 GetBufferOut()	114
10.9.3.6 GetChunkFrames()	115
10.9.3.7 GetEstimatedDelay()	115
10.9.3.8 GetMetadataInfos()	115

10.9.3.9	GetParam()	115
10.9.3.10	GetWebServerAppHealth()	116
10.9.3.11	IsOk()	116
10.9.3.12	NewClient()	116
10.9.3.13	OnUpdateThreadStart()	116
10.9.3.14	OnUpdateThreadStop()	116
10.9.3.15	operator=()	116
10.9.3.16	PresetManager_InformChange()	116
10.9.3.17	ProcessAnyAudio()	117
10.9.3.18	ProcessAudio()	117
10.9.3.19	ProcessAudio_Planar()	118
10.9.3.20	SetMetadata()	118
10.9.3.21	SetMetadataMulti()	118
10.9.3.22	SetParam()	119
10.9.3.23	SetPresetManager()	119
10.9.3.24	SetWebServerAppHealth()	119
10.9.3.25	StartWebServer()	120
10.9.3.26	Stop()	120
10.9.3.27	StopWebServer()	120
10.9.3.28	Terminate()	121
10.10	sound4::impact::dyn::CInstance Class Reference	121
10.10.1	Detailed Description	122
10.10.2	Constructor & Destructor Documentation	122
10.10.2.1	CInstance() [1/2]	122
10.10.2.2	CInstance() [2/2]	122
10.10.2.3	~CInstance()	122
10.10.3	Member Function Documentation	122
10.10.3.1	AddAudio()	122
10.10.3.2	Create()	123
10.10.3.3	GetAudio()	123
10.10.3.4	GetBufferIn()	123
10.10.3.5	GetBufferOut()	124
10.10.3.6	GetChunkFrames()	124
10.10.3.7	GetEstimatedDelay()	124
10.10.3.8	GetMetadataInfos()	124
10.10.3.9	GetParam()	125
10.10.3.10	GetWebServerAppHealth()	125
10.10.3.11	IsOk()	125
10.10.3.12	NewClient()	125
10.10.3.13	OnUpdateThreadStart()	126
10.10.3.14	OnUpdateThreadStop()	126
10.10.3.15	operator=()	126

10.10.3.16 PresetManager_InformChange()	126
10.10.3.17 ProcessAnyAudio()	126
10.10.3.18 ProcessAudio()	127
10.10.3.19 ProcessAudio_Planar()	127
10.10.3.20 SetMetadata()	128
10.10.3.21 SetMetadataMulti()	128
10.10.3.22 SetParam()	128
10.10.3.23 SetPresetManager()	129
10.10.3.24 SetWebServerAppHealth()	129
10.10.3.25 StartWebServer()	129
10.10.3.26 Stop()	130
10.10.3.27 StopWebServer()	130
10.10.3.28 Terminate()	130
10.11 sound4::CPresetLoader Class Reference	130
10.11.1 Detailed Description	131
10.11.2 Constructor & Destructor Documentation	131
10.11.2.1 CPresetLoader()	131
10.11.2.2 ~CPresetLoader()	131
10.11.3 Member Function Documentation	131
10.11.3.1 Exists()	131
10.11.3.2 GetAll()	131
10.11.3.3 IsReadOnly()	131
10.11.3.4 Read()	132
10.11.3.5 Remove()	132
10.11.3.6 Rename()	132
10.11.3.7 Write()	132
10.12 sound4::impact::CPresetLoader Class Reference	132
10.12.1 Detailed Description	133
10.12.2 Constructor & Destructor Documentation	133
10.12.2.1 CPresetLoader()	133
10.12.2.2 ~CPresetLoader()	133
10.12.3 Member Function Documentation	133
10.12.3.1 Exists()	133
10.12.3.2 GetAll()	133
10.12.3.3 IsReadOnly()	133
10.12.3.4 Read()	133
10.12.3.5 Remove()	133
10.12.3.6 Rename()	134
10.12.3.7 Write()	134
10.13 sound4::impact::dyn::CPresetLoader Class Reference	134
10.13.1 Detailed Description	134
10.13.2 Constructor & Destructor Documentation	134

10.13.2.1 CPresetLoader()	134
10.13.2.2 ~CPresetLoader()	135
10.13.3 Member Function Documentation	135
10.13.3.1 Exists()	135
10.13.3.2 GetAll()	135
10.13.3.3 IsReadOnly()	135
10.13.3.4 Read()	135
10.13.3.5 Remove()	135
10.13.3.6 Rename()	135
10.13.3.7 Write()	136
10.14 sound4::CProcessor Class Reference	136
10.14.1 Detailed Description	138
10.14.2 Constructor & Destructor Documentation	138
10.14.2.1 CProcessor()	138
10.14.3 Member Function Documentation	138
10.14.3.1 AudioConvertFrom()	138
10.14.3.2 AudioConvertTo()	139
10.14.3.3 AudioMonoFromLiveStereo()	139
10.14.3.4 AudioMonoToLiveStereo()	139
10.14.3.5 GetAudioInputCount()	139
10.14.3.6 GetAudioOutputCount()	139
10.14.3.7 GetBytesFromFormat()	140
10.14.3.8 GetChannelCount()	140
10.14.3.9 GetChunkSizeInFrames()	140
10.14.3.10 GetFormatFromName()	140
10.14.3.11 GetFormatName()	141
10.14.3.12 GetPossibleChunkSizeInFrames()	141
10.14.3.13 GetPrefixSymbol()	142
10.14.3.14 GetSampleRate()	142
10.14.3.15 GetVersion()	142
10.14.3.16 IsOk()	142
10.14.3.17 Load()	142
10.14.3.18 MonoToStereo()	143
10.14.3.19 MonoToStereo_Planar()	143
10.14.3.20 SetLoggerCallback()	143
10.14.3.21 SetLogSeverity()	143
10.14.3.22 StereoToMono()	144
10.14.3.23 StereoToMono_Planar()	144
10.14.4 Friends And Related Symbol Documentation	144
10.14.4.1 CBus	144
10.14.4.2 CInstance	144
10.14.5 Member Data Documentation	144

10.14.5.1 fnAddAudio	144
10.14.5.2 fnAddPadAudio	144
10.14.5.3 fnAudioConvertFrom	145
10.14.5.4 fnAudioConvertTo	145
10.14.5.5 fnAudioMonoFromLiveStereo	145
10.14.5.6 fnAudioMonoToLiveStereo	145
10.14.5.7 fnDeleteClient	145
10.14.5.8 fnExitProcess	145
10.14.5.9 fnFreeBus	145
10.14.5.10 fnFreeJsonAnswer	146
10.14.5.11 fnFreeMetadataInfos	146
10.14.5.12 fnFreeParameters	146
10.14.5.13 fnFreeParameterValue	146
10.14.5.14 fnGetAudio	146
10.14.5.15 fnGetAudioInputCount	146
10.14.5.16 fnGetAudioOutputCount	146
10.14.5.17 fnGetBufferIn	147
10.14.5.18 fnGetBufferOut	147
10.14.5.19 fnGetBytesFromFormat	147
10.14.5.20 fnGetChannelCount	147
10.14.5.21 fnGetChunkSizeInFrames	147
10.14.5.22 fnGetEstimatedDelay	147
10.14.5.23 fnGetFormatFromName	147
10.14.5.24 fnGetFormatName	148
10.14.5.25 fnGetMaxPacketFrame	148
10.14.5.26 fnGetMetadataInfos	148
10.14.5.27 fnGetOutputCount	148
10.14.5.28 fnGetParameter	148
10.14.5.29 fnGetPossibleChunkSizeInFrames	148
10.14.5.30 fnGetProcessChunkFrames	148
10.14.5.31 fnGetSampleRate	149
10.14.5.32 fnGetVersion	149
10.14.5.33 fnInitProcess	149
10.14.5.34 fnInitProcess2	149
10.14.5.35 fnInitProcess3	149
10.14.5.36 fnMonoToStereo	149
10.14.5.37 fnMonoToStereo_Planar	149
10.14.5.38 fnNewBus	150
10.14.5.39 fnNewClient	150
10.14.5.40 fnNewParameters	150
10.14.5.41 fnPresetManager_InformChange	150
10.14.5.42 fnProcessAudio	150

10.14.5.43 fnProcessAudio_Planar	150
10.14.5.44 fnProcessJson	150
10.14.5.45 fnSaveState	151
10.14.5.46 fnSetInstanceBus	151
10.14.5.47 fnSetLoggerCallback	151
10.14.5.48 fnSetLogSeverity	151
10.14.5.49 fnSetMetadata	151
10.14.5.50 fnSetMetadataMulti	151
10.14.5.51 fnSetParameter	151
10.14.5.52 fnSetPresetManager	152
10.14.5.53 fnStartUpdateThread	152
10.14.5.54 fnStereoToMono	152
10.14.5.55 fnStereoToMono_Planar	152
10.14.5.56 fnStopUpdateThread	152
10.14.5.57 fnTerminateProcess	152
10.14.5.58 fnWaitUpdateThreadReady	152
10.14.5.59 fnWebserver	153
10.14.5.60 fnWebserver_FreeString	153
10.14.5.61 fnWebserver_GetAppHealth	153
10.14.5.62 fnWebserver_SetAppHealth	153
10.14.5.63 fnWebserver_Status	153
10.14.5.64 fnWebserver_Stop	153
10.14.5.65 fnWebserver_tcp	153
10.14.5.66 fnWebserver_tcp2	154
10.15 sound4::helper::DynFuncHolder< T > Class Template Reference	154
10.15.1 Detailed Description	154
10.15.2 Constructor & Destructor Documentation	154
10.15.2.1 DynFuncHolder() [1/2]	154
10.15.2.2 DynFuncHolder() [2/2]	155
10.15.3 Member Function Documentation	155
10.15.3.1 IsOk()	155
10.15.3.2 operator bool()	155
10.15.3.3 operator T*()	155
10.16 sound4::impact::dyn::helper::DynFuncHolder< T > Class Template Reference	155
10.16.1 Detailed Description	155
10.16.2 Constructor & Destructor Documentation	156
10.16.2.1 DynFuncHolder() [1/2]	156
10.16.2.2 DynFuncHolder() [2/2]	156
10.16.3 Member Function Documentation	156
10.16.3.1 IsOk()	156
10.16.3.2 operator bool()	156
10.16.3.3 operator T*()	156

10.17 impact_CClientInstance Struct Reference	157
10.17.1 Detailed Description	157
10.18 impact_CInstance Struct Reference	157
10.18.1 Detailed Description	157
10.19 impact_CParameters Struct Reference	157
10.19.1 Detailed Description	157
10.20 MyPresetManager Class Reference	157
10.20.1 Detailed Description	158
10.20.2 Constructor & Destructor Documentation	158
10.20.2.1 MyPresetManager() [1/2]	158
10.20.2.2 ~MyPresetManager() [1/2]	158
10.20.2.3 MyPresetManager() [2/2]	159
10.20.2.4 ~MyPresetManager() [2/2]	159
10.20.3 Member Function Documentation	159
10.20.3.1 Exists() [1/2]	159
10.20.3.2 Exists() [2/2]	159
10.20.3.3 GetAll() [1/2]	159
10.20.3.4 GetAll() [2/2]	159
10.20.3.5 IsReadOnly() [1/2]	160
10.20.3.6 IsReadOnly() [2/2]	160
10.20.3.7 Read() [1/2]	160
10.20.3.8 Read() [2/2]	160
10.20.3.9 Remove() [1/2]	160
10.20.3.10 Remove() [2/2]	160
10.20.3.11 Rename() [1/2]	161
10.20.3.12 Rename() [2/2]	161
10.20.3.13 SetupPath() [1/2]	161
10.20.3.14 SetupPath() [2/2]	161
10.20.3.15 Write() [1/2]	161
10.20.3.16 Write() [2/2]	162
10.21 sound4::helper::SampleFormat< T > Struct Template Reference	162
10.21.1 Detailed Description	162
10.22 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference	162
10.22.1 Detailed Description	162
10.23 sound4::impact::helper::SampleFormat< T > Struct Template Reference	163
10.23.1 Detailed Description	163
10.24 sound4::helper::SampleFormat< float > Struct Reference	163
10.24.1 Detailed Description	163
10.24.2 Member Data Documentation	164
10.24.2.1 format	164
10.25 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference	164
10.25.1 Detailed Description	164

10.25.2 Member Data Documentation	164
10.25.2.1 format	164
10.26 sound4::impact::helper::SampleFormat< float > Struct Reference	164
10.26.1 Detailed Description	165
10.26.2 Member Data Documentation	165
10.26.2.1 format	165
10.27 sound4::helper::SampleFormat< int16_t > Struct Reference	165
10.27.1 Detailed Description	165
10.27.2 Member Data Documentation	165
10.27.2.1 format	165
10.28 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference	165
10.28.1 Detailed Description	166
10.28.2 Member Data Documentation	166
10.28.2.1 format	166
10.29 sound4::impact::helper::SampleFormat< int16_t > Struct Reference	166
10.29.1 Detailed Description	166
10.29.2 Member Data Documentation	166
10.29.2.1 format	166
10.30 sound4::helper::SampleFormat< int32_t > Struct Reference	167
10.30.1 Detailed Description	167
10.30.2 Member Data Documentation	167
10.30.2.1 format	167
10.31 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference	167
10.31.1 Detailed Description	167
10.31.2 Member Data Documentation	168
10.31.2.1 format	168
10.32 sound4::impact::helper::SampleFormat< int32_t > Struct Reference	168
10.32.1 Detailed Description	168
10.32.2 Member Data Documentation	168
10.32.2.1 format	168
10.33 SOUND4_ProcessInfo Struct Reference	168
10.33.1 Detailed Description	169
10.33.2 Member Data Documentation	169
10.33.2.1 name	169
10.33.2.2 prefix	169
10.33.2.3 shortname	169
10.33.2.4 struct_size	169
10.33.2.5 version	170
10.34 sound4cl_SOUND4_ProcessInfo Struct Reference	170
10.34.1 Detailed Description	170
10.34.2 Member Data Documentation	170
10.34.2.1 name	170

10.34.2.2 prefix	170
10.34.2.3 shortname	171
10.34.2.4 struct_size	171
10.34.2.5 version	171
11 File Documentation	173
11.1 sound4.impact.cl-pipe-cxx.cpp File Reference	173
11.1.1 Function Documentation	173
11.1.1.1 GetEnv()	173
11.1.1.2 main()	174
11.1.1.3 MyLogger()	174
11.1.1.4 ProcessLoop()	174
11.1.1.5 Usage()	174
11.2 sound4.impact.cl-pipe-cxx.cpp	175
11.3 sound4.impact.cl-pipe-dyn.cpp File Reference	176
11.3.1 Function Documentation	177
11.3.1.1 GetEnv()	177
11.3.1.2 main()	177
11.3.1.3 MyLogger()	178
11.3.1.4 ProcessLoop()	178
11.3.1.5 Usage()	178
11.4 sound4.impact.cl-pipe-dyn.cpp	178
11.5 sound4.impact.cl-pipe.cpp File Reference	181
11.5.1 Macro Definition Documentation	182
11.5.1.1 CHANNEL_COUNT	182
11.5.1.2 CHUNK_SIZE	182
11.5.1.3 SAMPLE_RATE	182
11.5.2 Function Documentation	182
11.5.2.1 main()	182
11.5.2.2 MyLogger()	183
11.5.2.3 ProcessLoop()	183
11.5.2.4 Usage()	183
11.6 sound4.impact.cl-pipe.cpp	183
11.7 sound4cl-pipe.cpp File Reference	186
11.7.1 Function Documentation	187
11.7.1.1 GetEnv()	187
11.7.1.2 main()	187
11.7.1.3 ProcessLoop()	188
11.8 sound4cl-pipe.cpp	188
11.9 sound4.impact.cl.h File Reference	191
11.9.1 Macro Definition Documentation	194
11.9.1.1 IMPACT_DECL	194

11.9.1.2	IMPACT_HAS_CLOUDBUS	194
11.9.1.3	IMPACT_HAS_WEBSERVER	194
11.10	sound4.impact.cl.h	194
11.11	sound4.impact.cl.hpp File Reference	198
11.11.1	Macro Definition Documentation	199
11.11.1.1	fs_strdup	199
11.12	sound4.impact.cl.hpp	200
11.13	sound4.impact.cl_dyn.hpp File Reference	207
11.13.1	Detailed Description	208
11.13.2	Macro Definition Documentation	209
11.13.2.1	fs_strdup	209
11.13.2.2	UNIXLIKE	209
11.14	sound4.impact.cl_dyn.hpp	209
11.15	sound4cl.hpp File Reference	222
11.15.1	Detailed Description	224
11.15.2	Macro Definition Documentation	224
11.15.2.1	UNIXLIKE	224
11.16	sound4cl.hpp	224
11.17	sound4cl_cdef.h File Reference	237
11.17.1	Macro Definition Documentation	239
11.17.1.1	fs_strdup	239
11.17.1.2	SOUND4_INVALID_WEBSERVER_ID	239
11.17.1.3	SOUND4_WEBSERVER_HTTP_OK	240
11.17.1.4	SOUND4_WEBSERVER_HTTPS_OK	240
11.17.2	Typedef Documentation	240
11.17.2.1	fs_char	240
11.17.2.2	sound4cl_loggerfn	240
11.17.2.3	sound4cl_storage_exists	240
11.17.2.4	sound4cl_storage_getall	240
11.17.2.5	sound4cl_storage_getall_free	240
11.17.2.6	sound4cl_storage_reader	241
11.17.2.7	sound4cl_storage_reader_free	241
11.17.2.8	sound4cl_storage_remove	241
11.17.2.9	sound4cl_storage_rename	241
11.17.2.10	sound4cl_storage_writer	241
11.17.3	Enumeration Type Documentation	241
11.17.3.1	sound4cl_LogSeverity	241
11.17.3.2	sound4cl_PresetChange_Kind	242
11.17.3.3	sound4cl_SampleFormat	242
11.17.4	Function Documentation	242
11.17.4.1	sound4cl_AddAudio()	242
11.17.4.2	sound4cl_AddPadAudio()	243

11.17.4.3 sound4cl_AudioConvertFrom()	243
11.17.4.4 sound4cl_AudioConvertTo()	243
11.17.4.5 sound4cl_AudioMonoFromLiveStereo()	243
11.17.4.6 sound4cl_AudioMonoToLiveStereo()	243
11.17.4.7 sound4cl_DeleteClient()	243
11.17.4.8 sound4cl_ExitProcess()	243
11.17.4.9 sound4cl_FreeBus()	243
11.17.4.10 sound4cl_FreeJsonAnswer()	244
11.17.4.11 sound4cl_FreeMetadataInfos()	244
11.17.4.12 sound4cl_FreeParameters()	244
11.17.4.13 sound4cl_FreeParameterValue()	244
11.17.4.14 sound4cl_GetAudio()	244
11.17.4.15 sound4cl_GetAudioInputCount()	244
11.17.4.16 sound4cl_GetAudioOutputCount()	244
11.17.4.17 sound4cl_GetBufferIn()	244
11.17.4.18 sound4cl_GetBufferOut()	245
11.17.4.19 sound4cl_GetBytesFromFormat()	245
11.17.4.20 sound4cl_GetChannelCount()	245
11.17.4.21 sound4cl_GetChunkSizeInFrames()	245
11.17.4.22 sound4cl_GetEstimatedDelay()	245
11.17.4.23 sound4cl_GetFormatFromName()	245
11.17.4.24 sound4cl_GetFormatName()	245
11.17.4.25 sound4cl_GetMaxPacketFrame()	245
11.17.4.26 sound4cl_GetMetadataInfos()	245
11.17.4.27 sound4cl_GetOutputCount()	246
11.17.4.28 sound4cl_GetParameter()	246
11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames()	246
11.17.4.30 sound4cl_GetProcessChunkFrames()	246
11.17.4.31 sound4cl_GetSampleRate()	246
11.17.4.32 sound4cl_GetVersion()	246
11.17.4.33 sound4cl_InitProcess()	246
11.17.4.34 sound4cl_InitProcess2()	246
11.17.4.35 sound4cl_InitProcess3()	247
11.17.4.36 sound4cl_MonoToStereo()	247
11.17.4.37 sound4cl_MonoToStereo_Planar()	247
11.17.4.38 sound4cl_NewBus()	247
11.17.4.39 sound4cl_NewClient()	247
11.17.4.40 sound4cl_NewParameters()	247
11.17.4.41 sound4cl_PresetManager_InformChange()	247
11.17.4.42 sound4cl_ProcessAudio()	248
11.17.4.43 sound4cl_ProcessAudio_Planar()	248
11.17.4.44 sound4cl_ProcessJson()	248

11.17.4.45 sound4cl_SaveState()	248
11.17.4.46 sound4cl_SetInstanceBus()	248
11.17.4.47 sound4cl_SetLoggerCallback()	248
11.17.4.48 sound4cl_SetLogSeverity()	248
11.17.4.49 sound4cl_SetMetadata()	248
11.17.4.50 sound4cl_SetMetadataMulti()	249
11.17.4.51 sound4cl_SetParameter()	249
11.17.4.52 sound4cl_SetPresetManager()	249
11.17.4.53 sound4cl_SOUND4_GetProcessInfo()	249
11.17.4.54 sound4cl_StartUpdateThread()	249
11.17.4.55 sound4cl_StereoToMono()	249
11.17.4.56 sound4cl_StereoToMono_Planar()	250
11.17.4.57 sound4cl_StopUpdateThread()	250
11.17.4.58 sound4cl_TerminateProcess()	250
11.17.4.59 sound4cl_WaitUpdateThreadReady()	250
11.17.4.60 sound4cl_Webserver()	250
11.17.4.61 sound4cl_Webserver_FreeString()	250
11.17.4.62 sound4cl_Webserver_GetAppHealth()	250
11.17.4.63 sound4cl_Webserver_SetAppHealth()	251
11.17.4.64 sound4cl_Webserver_Status()	251
11.17.4.65 sound4cl_Webserver_Stop()	251
11.17.4.66 sound4cl_Webserver_tcp()	251
11.17.4.67 sound4cl_Webserver_tcp2()	251
11.18 sound4cl_cdef.h	251

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	26
Custom preset managagement	30
Metadata Management	36
Update Management	38
Direct processing	39
Audio managing (any audio)	42
Audio convert helpers	46
Client's JSoN config interfacing	49
JSON syntax	52
JSON syntax for internal setup	55
Web server	58
Sample using C++ interface wrapper	63
Sample using C++ interface wrapper with dynamic library loading	64
Sample using Generic C++ interface (multi-process compatible)	66

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

dyn	69
sound4	69
sound4::helper	72
sound4::impact	73
sound4::impact::dyn	75
sound4::impact::dyn::helper	77
sound4::impact::helper	78
std	78

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

sound4::CBus	79
sound4::CInstance::CClient	80
sound4::impact::CInstance::CClient	82
sound4::impact::dyn::CInstance::CClient	83
sound4::impact::dyn::CDynLib	85
sound4::helper::CDynLoader	96
sound4::impact::dyn::helper::CDynLoader	99
sound4::CInstance	101
sound4::impact::CInstance	111
sound4::impact::dyn::CInstance	121
sound4::CPresetLoader	130
MyPresetManager	157
sound4::impact::CPresetLoader	132
sound4::impact::dyn::CPresetLoader	134
MyPresetManager	157
sound4::CProcessor	136
sound4::helper::DynFuncHolder< T >	154
sound4::impact::dyn::helper::DynFuncHolder< T >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddAudio) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddPadAudio) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertFrom) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertTo) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_DeleteClient) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ExitProcess) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameters) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameterValue) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudio) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioInputCount) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferIn) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferOut) >	155

sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChannelCount) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatFromName) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatName) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMetadataInfos) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetOutputCount) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetParameter) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetSampleRate) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetVersion) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess2) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess3) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewClient) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewParameters) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_PresetManager_InformChange) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessJson) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SaveState) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLoggerCallback) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLogSeverity) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadata) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadataMulti) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetParameter) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetPresetManager) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StartUpdateThread) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StopUpdateThread) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_TerminateProcess) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_FreeString) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_GetAppHealth) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_SetAppHealth) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Status) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Stop) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp) >	155
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp2) >	155
sound4::helper::DynFuncHolder< decltype(sound4cl_AddAudio) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_DeleteClient) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_ExitProcess) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeBus) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameters) >	154

sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudio) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatName) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetParameter) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_GetVersion) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess2) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess3) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_NewBus) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_NewClient) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_NewParameters) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessJson) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SaveState) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadata) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetParameter) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) >	154
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) >	154
impact_CClientInstance	157
impact_CInstance	157
impact_CParameters	157
sound4::helper::SampleFormat< T >	162

sound4::impact::dyn::helper::SampleFormat< T >	162
sound4::impact::helper::SampleFormat< T >	163
sound4::helper::SampleFormat< float >	163
sound4::impact::dyn::helper::SampleFormat< float >	164
sound4::impact::helper::SampleFormat< float >	164
sound4::helper::SampleFormat< int16_t >	165
sound4::impact::dyn::helper::SampleFormat< int16_t >	165
sound4::impact::helper::SampleFormat< int16_t >	166
sound4::helper::SampleFormat< int32_t >	167
sound4::impact::dyn::helper::SampleFormat< int32_t >	167
sound4::impact::helper::SampleFormat< int32_t >	168
SOUND4_ProcessInfo	168
sound4cl_SOUND4_ProcessInfo	170

Chapter 6

Class Index

6.1 Class List

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

sound4::CBus	
Shared Bus	79
sound4::CInstance::CClient	80
sound4::impact::CInstance::CClient	82
sound4::impact::dyn::CInstance::CClient	83
sound4::impact::dyn::CDynLib	
Dynamic library interface	85
sound4::helper::CDynLoader	
Helper to load dynamic library	96
sound4::impact::dyn::helper::CDynLoader	
Helper to load dynamic library	99
sound4::CInstance	
Instance handling class	101
sound4::impact::CInstance	111
sound4::impact::dyn::CInstance	121
sound4::CPresetLoader	
Custom preset handler helper	130
sound4::impact::CPresetLoader	132
sound4::impact::dyn::CPresetLoader	134
sound4::CProcessor	
Dynamic library interface	136
sound4::helper::DynFuncHolder< T >	
Helper to wrap a function pointer	154
sound4::impact::dyn::helper::DynFuncHolder< T >	
Helper to wrap a function pointer	155
impact_CClientInstance	157
impact_CInstance	157
impact_CParameters	157
MyPresetManager	
[Custom preset manager]	157
sound4::helper::SampleFormat< T >	
Helper for Sample format types	162
sound4::impact::dyn::helper::SampleFormat< T >	
Helper for Sample format types	162
sound4::impact::helper::SampleFormat< T >	
Helper for Sample format types	163

sound4::helper::SampleFormat< float >	163
sound4::impact::dyn::helper::SampleFormat< float >	164
sound4::impact::helper::SampleFormat< float >	164
sound4::helper::SampleFormat< int16_t >	165
sound4::impact::dyn::helper::SampleFormat< int16_t >	165
sound4::impact::helper::SampleFormat< int16_t >	166
sound4::helper::SampleFormat< int32_t >	167
sound4::impact::dyn::helper::SampleFormat< int32_t >	167
sound4::impact::helper::SampleFormat< int32_t >	168
SOUND4_ProcessInfo	168
sound4cl_SOUND4_ProcessInfo	170

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	173
sound4.impact.cl-pipe-dyn.cpp	
C++ sample using C++ interface wrapper with dynamic library loading	176
sound4.impact.cl-pipe.cpp	
C++ sample using only C interface	181
sound4cl-pipe.cpp	
C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided	186
sound4.impact.cl.h	
C interface for library	191
sound4.impact.cl.hpp	
C++ wrapper for library	198
sound4.impact.cl_dyn.hpp	
C++ wrapper for dynamic loading library	207
sound4cl.hpp	
C++ wrapper for dynamic loading library	222
sound4cl_cdef.h	
C function signature definition for all sound4.cl libraries	237

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=notempty` : force the storage in read-only mode
- `STATE_RO=notempty` : force the state in read-only mode (but can save presets/auth)
- `WAIT_LICENSE=notempty` : force to wait the license load before starting (always in file mode)
- `LOG_HEADER=string` : use string as header for logs in instance as much as possible

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 173 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 179 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 161 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 167 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 [185](#) 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 [300](#) 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 281 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 managagement

This allows to provide a customized preset management.

Typedefs

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

Enumerations

- enum [impact_PresetChange_Kind](#) { [change_kind_created](#) = 1, [change_kind_modified](#), [change_kind_deleted](#) }

Functions

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

8.5.1 Detailed Description

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

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

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

Known files:

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

Attention

`state.json` name can be 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 568 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 607 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 impact_SetPresetManager
-----------------	--

Returns

A NULL terminated array of file names

Definition at line 616 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 impact_SetPresetManager

Definition at line 624 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 impact_SetPresetManager

Returns

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

Definition at line 580 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 588 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 633 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 643 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 598 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

<code>change_kind_created</code>	Path has been created.
<code>change_kind_modified</code>	Path has been modified.
<code>change_kind_deleted</code>	Path has been deleted.

Definition at line 676 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

Parameters

<i>getall</i>	The function to get all files in storage
<i>getall_free</i>	The function to free the result from getall
<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 899 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": "AABAAUAEBAAAAEAIBo..." }
```

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

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 1575 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 1580 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 1584 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;
        if ( !std::filesystem::exists(m_preset_path) ) {
            return false;
        }
        std::error_code ec;
        if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
            return false;
        }
        return true;
    }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
    {
        std::error_code ec;
        return std::filesystem::exists(GetRealPath(name), ec);
    }
    virtual bool Remove(const std::filesystem::path &name) override
    {
        std::error_code ec;
        return std::filesystem::remove(GetRealPath(name), ec);
    }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
    {
        std::error_code ec;
        std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
        return (!ec);
    }
    virtual std::vector<std::filesystem::path> GetAll() override
    {
        std::vector<std::filesystem::path> list;
        for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
            std::filesystem::directory_options::skip_permission_denied)) {
            if (entry.is_regular_file()) {
                list.push_back(entry.path().lexically_relative(m_preset_path));
            }
        }
        return list;
    }
    virtual std::string Read(const std::filesystem::path &filename) override
    {
        std::ifstream f(GetRealPath(filename));
        if ( f.fail() ) {
            return std::string();
        }
        std::ostringstream ss;
        ss << f.rdbuf();
        return ss.str();
    }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
    {
        std::ofstream f(GetRealPath(filename));
        if ( f.fail() ) {
            return false;
        }
    }
}

```

```

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

```

8.16.8 Custom preset manager loading

```

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

```

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

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

It uses [sound4cl.hpp](#) header.

See full example [cpp/sound4cl-pipe.cpp](#)

8.17.1 Loading the library

```

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

```

8.17.2 Setup and configure

```

processor.SetLogSeverity(sound4::verbose5);
processor.SetLoggerCallback([](sound4::LogSeverity severity, const std::string& msg) {
    std::cerr << msg << std::endl;
});

sound4::CInstance instance(processor);
instance.SetParam("ADMIN_USER", "admin");
instance.SetParam("ADMIN_SECRET", "admin");

```

8.17.3 Create the processing instance

```

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

```

8.17.4 Processing loop

```

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

```

8.17.5 Starting a web-server

```

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

```

8.17.6 Communicating with the instance

See also

JSON syntax

```

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

```

8.17.7 Custom preset manager example

```

class MyPresetManager : public sound4::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;

    bool SetupPath(const std::filesystem::path& a_preset_path)
    {
        m_preset_path=a_preset_path;
        if ( !std::filesystem::exists(m_preset_path) ) {
            return false;
        }
        std::error_code ec;
        if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
            return false;
        }
        return true;
    }

    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
    {
        std::error_code ec;
        return std::filesystem::exists(GetRealPath(name), ec);
    }

    virtual bool Remove(const std::filesystem::path &name) override
    {
        std::error_code ec;
        return std::filesystem::remove(GetRealPath(name), ec);
    }

    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
    {
        std::error_code ec;
        std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
        return (!ec);
    }
}

```

```

virtual std::vector<std::filesystem::path> GetAll() override
{
    std::vector<std::filesystem::path> list;
    for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
std::filesystem::directory_options::skip_permission_denied)) {
        if (entry.is_regular_file()) {
            list.push_back(entry.path().lexically_relative(m_preset_path));
        }
    }
    return list;
}

virtual std::string Read(const std::filesystem::path &filename) override
{
    std::ifstream f(GetRealPath(filename));
    if ( f.fail() ) {
        return std::string();
    }
    std::ostringstream ss;
    ss << f.rdbuf();
    return ss.str();
}

virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
{
    std::ofstream f(GetRealPath(filename));
    if ( f.fail() ) {
        return false;
    }
    f << content;
    if ( f.fail() ) {
        return false;
    }
    return true;
}

private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
    {
        return m_preset_path / name;
    }
};

```

8.17.8 Custom preset manager loading

```

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

```

Chapter 9

Namespace Documentation

9.1 dyn Namespace Reference

9.1.1 Detailed Description

Dynamic loading namespace

9.2 sound4 Namespace Reference

Namespaces

- 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 [749](#) 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 739 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 735 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 807 of file [sound4.impact.cl_dyn.hpp](#).

9.5.1.4 SampleFormat

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

Definition at line 468 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 472 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 476 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 484 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 480 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 488 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 492 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 [709](#) of file [sound4cl.hpp](#).

10.1.2 Constructor & Destructor Documentation

10.1.2.1 CBus()

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

Definition at line [711](#) of file [sound4cl.hpp](#).

10.1.2.2 ~CBus()

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

Definition at line [718](#) 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 725 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 1221 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 1223 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 1233 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 1247 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 1278 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 1280 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 1290 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 1304 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 458 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 756 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 761 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 773 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 1279 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 860 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 1285 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 1134 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 1150 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 906 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 1118 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 995 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 799 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 1072 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 781 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 1271 of file [sound4cl.hpp](#).

10.8.3.13 OnUpdateThreadStart()

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

Definition at line 1017 of file [sound4cl.hpp](#).

10.8.3.14 OnUpdateThreadStop()

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

Definition at line 1019 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 920 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 1197 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 1160 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 1179 of file [sound4cl.hpp](#).

10.8.3.20 SetBus()

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

Set the instance bus

Definition at line 811 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 938 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 959 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 789 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 827 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 1063 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 1026 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 1101 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 1051 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 1090 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 812 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 817 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 829 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 1336 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 917 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 1342 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 1189 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 1206 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 960 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 1173 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 1049 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 855 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 1123 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 837 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 1328 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.13 OnUpdateThreadStart()

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

Definition at line 1071 of file [sound4.impact.cl_dyn.hpp](#).

10.10.3.14 OnUpdateThreadStop()

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

Definition at line 1073 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 974 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 1254 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 1217 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 1236 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 992 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 1013 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 845 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 884 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 1114 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 1081 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 1154 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 1103 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 1143 of file [sound4.impact.cl_dyn.hpp](#).

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

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

10.11 sound4::CPresetLoader Class Reference

Custom preset handler helper.

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

Inherited by [MyPresetManager](#).

Public Member Functions

- [CPresetLoader](#) ()=default
- virtual [~CPresetLoader](#) ()=default
- virtual bool [IsReadOnly](#) ()=0
- virtual bool [Exists](#) (const std::filesystem::path &name)=0
- virtual bool [Remove](#) (const std::filesystem::path &name)=0
- virtual bool [Rename](#) (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > [GetAll](#) ()=0
- virtual std::string [Read](#) (const std::filesystem::path &filename)=0
- virtual bool [Write](#) (const std::filesystem::path &filename, const std::string &content)=0

10.11.1 Detailed Description

Derive from this class and call [CInstance::SetPresetManager](#)

Definition at line 736 of file [sound4cl.hpp](#).

10.11.2 Constructor & Destructor Documentation

10.11.2.1 CPresetLoader()

```
sound4::CPresetLoader::CPresetLoader ( ) [default]
```

10.11.2.2 ~CPresetLoader()

```
virtual sound4::CPresetLoader::~~CPresetLoader ( ) [virtual], [default]
```

10.11.3 Member Function Documentation

10.11.3.1 Exists()

```
virtual bool sound4::CPresetLoader::Exists (
    const std::filesystem::path & name ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.2 GetAll()

```
virtual std::vector< std::filesystem::path > sound4::CPresetLoader::GetAll ( ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.3 IsReadOnly()

```
virtual bool sound4::CPresetLoader::IsReadOnly ( ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.4 Read()

```
virtual std::string sound4::CPresetLoader::Read (
    const std::filesystem::path & filename ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.5 Remove()

```
virtual bool sound4::CPresetLoader::Remove (
    const std::filesystem::path & name ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.6 Rename()

```
virtual bool sound4::CPresetLoader::Rename (
    const std::filesystem::path & from,
    const std::filesystem::path & to ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

10.11.3.7 Write()

```
virtual bool sound4::CPresetLoader::Write (
    const std::filesystem::path & filename,
    const std::string & content ) [pure virtual]
```

Implemented in [MyPresetManager](#), and [MyPresetManager](#).

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

- [sound4cl.hpp](#)

10.12 sound4::impact::CPresetLoader Class Reference

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

Public Member Functions

- [CPresetLoader](#) ()=default
- virtual [~CPresetLoader](#) ()=default
- virtual bool [IsReadOnly](#) ()=0
- virtual bool [Exists](#) (const std::filesystem::path &name)=0
- virtual bool [Remove](#) (const std::filesystem::path &name)=0
- virtual bool [Rename](#) (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > [GetAll](#) ()=0
- virtual std::string [Read](#) (const std::filesystem::path &filename)=0
- virtual bool [Write](#) (const std::filesystem::path &filename, const std::string &content)=0

10.12.1 Detailed Description

Custom preset handler helper Derive from this class and call [CInstance::SetPresetManager](#)

Definition at line 367 of file [sound4.impact.cl.hpp](#).

10.12.2 Constructor & Destructor Documentation

10.12.2.1 CPresetLoader()

```
sound4::impact::CPresetLoader::CPresetLoader ( ) [default]
```

10.12.2.2 ~CPresetLoader()

```
virtual sound4::impact::CPresetLoader::~~CPresetLoader ( ) [virtual], [default]
```

10.12.3 Member Function Documentation

10.12.3.1 Exists()

```
virtual bool sound4::impact::CPresetLoader::Exists (
    const std::filesystem::path & name ) [pure virtual]
```

10.12.3.2 GetAll()

```
virtual std::vector< std::filesystem::path > sound4::impact::CPresetLoader::GetAll ( ) [pure virtual]
```

10.12.3.3 IsReadOnly()

```
virtual bool sound4::impact::CPresetLoader::IsReadOnly ( ) [pure virtual]
```

10.12.3.4 Read()

```
virtual std::string sound4::impact::CPresetLoader::Read (
    const std::filesystem::path & filename ) [pure virtual]
```

10.12.3.5 Remove()

```
virtual bool sound4::impact::CPresetLoader::Remove (
    const std::filesystem::path & name ) [pure virtual]
```

10.12.3.6 Rename()

```
virtual bool sound4::impact::CPresetLoader::Rename (
    const std::filesystem::path & from,
    const std::filesystem::path & to ) [pure virtual]
```

10.12.3.7 Write()

```
virtual bool sound4::impact::CPresetLoader::Write (
    const std::filesystem::path & filename,
    const std::string & content ) [pure virtual]
```

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

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

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

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

Inherited by [MyPresetManager](#).

Public Member Functions

- [CPresetLoader](#) ()=default
- virtual [~CPresetLoader](#) ()=default
- virtual bool [IsReadOnly](#) ()=0
- virtual bool [Exists](#) (const std::filesystem::path &name)=0
- virtual bool [Remove](#) (const std::filesystem::path &name)=0
- virtual bool [Rename](#) (const std::filesystem::path &from, const std::filesystem::path &to)=0
- virtual std::vector< std::filesystem::path > [GetAll](#) ()=0
- virtual std::string [Read](#) (const std::filesystem::path &filename)=0
- virtual bool [Write](#) (const std::filesystem::path &filename, const std::string &content)=0

10.13.1 Detailed Description

Custom preset handler helper Derive from this class and call [CInstance::SetPresetManager](#)

Definition at line [794](#) 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 569 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 571 of file [sound4cl.hpp](#).

10.14.3.3 AudioMonoFromLiveStereo()

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

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

10.14.3.4 AudioMonoToLiveStereo()

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

Definition at line 583 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 641 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 651 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 688 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 631 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 602 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 679 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 670 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 613 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 661 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 591 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 561 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 575 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 579 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 700 of file [sound4cl.hpp](#).

10.14.3.21 SetLogSeverity()

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

Sets the log severity (global)

Definition at line 694 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 573 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 577 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::fnGet↵  
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::fnMonoToStereo_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

<code>T</code>	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]

Inherits [sound4::impact::dyn::CPresetLoader](#), and [sound4::CPresetLoader](#).

Public Member Functions

- [MyPresetManager](#) ()=default
- virtual [~MyPresetManager](#) ()=default
- bool [SetupPath](#) (const std::filesystem::path &a_preset_path)
- virtual bool [IsReadOnly](#) () override
- virtual bool [Exists](#) (const std::filesystem::path &name) override
- virtual bool [Remove](#) (const std::filesystem::path &name) override
- virtual bool [Rename](#) (const std::filesystem::path &from, const std::filesystem::path &to) override
- virtual std::vector< std::filesystem::path > [GetAll](#) () override
- virtual std::string [Read](#) (const std::filesystem::path &filename) override
- virtual bool [Write](#) (const std::filesystem::path &filename, const std::string &content) override
- [MyPresetManager](#) ()=default
- virtual [~MyPresetManager](#) ()=default
- bool [SetupPath](#) (const std::filesystem::path &a_preset_path)
- virtual bool [IsReadOnly](#) () override
- virtual bool [Exists](#) (const std::filesystem::path &name) override
- virtual bool [Remove](#) (const std::filesystem::path &name) override
- virtual bool [Rename](#) (const std::filesystem::path &from, const std::filesystem::path &to) override
- virtual std::vector< std::filesystem::path > [GetAll](#) () override
- virtual std::string [Read](#) (const std::filesystem::path &filename) override
- virtual bool [Write](#) (const std::filesystem::path &filename, const std::string &content) override

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 163 of file [sound4.impact.cl-pipe-dyn.cpp](#).

10.20.3.8 Read() [2/2]

```
virtual std::string MyPresetManager::Read (
    const std::filesystem::path & filename ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 147 of file [sound4cl-pipe.cpp](#).

10.20.3.9 Remove() [1/2]

```
virtual bool MyPresetManager::Remove (
    const std::filesystem::path & name ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 142 of file [sound4.impact.cl-pipe-dyn.cpp](#).

10.20.3.10 Remove() [2/2]

```
virtual bool MyPresetManager::Remove (
    const std::filesystem::path & name ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 126 of file [sound4cl-pipe.cpp](#).

10.20.3.11 Rename() [1/2]

```
virtual bool MyPresetManager::Rename (
    const std::filesystem::path & from,
    const std::filesystem::path & to ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 147 of file [sound4.impact.cl-pipe-dyn.cpp](#).

10.20.3.12 Rename() [2/2]

```
virtual bool MyPresetManager::Rename (
    const std::filesystem::path & from,
    const std::filesystem::path & to ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 131 of file [sound4cl-pipe.cpp](#).

10.20.3.13 SetupPath() [1/2]

```
bool MyPresetManager::SetupPath (
    const std::filesystem::path & a_preset_path ) [inline]
```

Definition at line 124 of file [sound4.impact.cl-pipe-dyn.cpp](#).

10.20.3.14 SetupPath() [2/2]

```
bool MyPresetManager::SetupPath (
    const std::filesystem::path & a_preset_path ) [inline]
```

Definition at line 108 of file [sound4cl-pipe.cpp](#).

10.20.3.15 Write() [1/2]

```
virtual bool MyPresetManager::Write (
    const std::filesystem::path & filename,
    const std::string & content ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 173 of file [sound4.impact.cl-pipe-dyn.cpp](#).

10.20.3.16 Write() [2/2]

```
virtual bool MyPresetManager::Write (
    const std::filesystem::path & filename,
    const std::string & content ) [inline], [override], [virtual]
```

Implements [sound4::impact::dyn::CPresetLoader](#).

Definition at line 157 of file [sound4cl-pipe.cpp](#).

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

- [sound4.impact.cl-pipe-dyn.cpp](#)
- [sound4cl-pipe.cpp](#)

10.21 sound4::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

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

10.21.1 Detailed Description

```
template<typename T>
struct sound4::helper::SampleFormat< T >
```

Template Parameters

<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 531 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 542 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 543 of file [sound4.impact.cl_dyn.hpp](#).

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

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

10.26 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 534 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 535 of file [sound4.impact.cl_dyn.hpp](#).

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

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

10.29 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 538 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 539 of file [sound4.impact.cl_dyn.hpp](#).

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

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

10.32 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 192 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 194 of file [sound4.impact.cl.h](#).

10.33.2.2 prefix

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

Definition at line 195 of file [sound4.impact.cl.h](#).

10.33.2.3 shortname

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

Definition at line 196 of file [sound4.impact.cl.h](#).

10.33.2.4 struct_size

```
size_t SOUND4_ProcessInfo::struct_size
```

Definition at line 193 of file [sound4.impact.cl.h](#).

10.33.2.5 version

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

Definition at line 197 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)]" <<
00085         std::endl;;
00086     std::cerr << "  - S4LOGINKEY           : SOUND4 Login key (or S4USERID)" << std::endl;;
00087     std::cerr << "  - RADIO_NAME           : SOUND4 Radio Name" << std::endl;;
00088     std::cerr << "  - ACCESS_KEY_ID        : SOUND4 Access Key ID" << std::endl;;
00089     std::cerr << "  - ACCESS_KEY_SECRET    : SOUND4 Access Key Secret" << std::endl;;
00090     std::cerr << "Example:" << std::endl;
00091     std::cerr << "ffmpeg -re -i file.wav -ar " << sound4::impact::GetSampleRate() << " -ac " <<
00092         "sound4::impact::GetChannelCount() << " -f f32le - | " << progname << " | ffmpeg -f f32le -ar " <<
00093         "sound4::impact::GetSampleRate() << " -ac " << sound4::impact::GetChannelCount() << " -i -" << std::endl;;
00094     #if defined(_WIN32) || defined(WIN32)
00095         std::cerr << "ffmpeg -f dshow -i audio=\"Microphone\" -ar " << sound4::impact::GetSampleRate() << " " <<
00096         "sound4::impact::GetChannelCount() << " -f f32le - | " << progname << " | ffmpeg -f f32le -ar " <<
00097         "sound4::impact::GetSampleRate() << " -ac " << sound4::impact::GetChannelCount() << " -i -" << std::endl;;
00098     #else
00099         std::cerr << "ffmpeg -f pulse -i default -ar " << sound4::impact::GetSampleRate() << " -ac " <<
00100         "sound4::impact::GetChannelCount() << " -f f32le - | " << progname << " | ffmpeg -f f32le -ar " <<
00101         "sound4::impact::GetSampleRate() << " -ac " << sound4::impact::GetChannelCount() << " -i -" << std::endl;;
00102     #endif
00103 }
00104
00105 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         store_path = std::filesystem::path(store_path_str);
00136         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00137             std::cerr << "Storage folder '"<store_path.c_str()<"' is not a valid directory" <<
std::endl;
00138             Usage(programe);
00139             return 1;
00140         }
00141     }
00142
00143     sound4::impact::SetLogSeverity(verbose5);
00144     sound4::impact::SetLoggerCallback(MyLogger);
00145
00146     sound4::impact::CInstance instance;
00147     instance.SetParam("ADMIN_USER","admin");
00148     instance.SetParam("ADMIN_SECRET","admin");
00149
00150     std::cerr << "Creating" << std::endl;
00151     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
00152         std::cerr << "Failed to create the processing instance" << std::endl;
00153         return 1;
00154     }
00155
00156     #if (IMPACT_HAS_WEBSERVER)
00157         std::cerr << "Starting web server" << std::endl;
00158         if (!instance.StartWebServer(listen_port,0)) {
00159             std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00160             return 1;
00161         }
00162     #endif // (IMPACT_HAS_WEBSERVER)
00163
00164     {
00165         auto client = instance.NewClient();
00166         auto answer = client->ProcessJson(R"JSON({{get":{"processorname":null}}})JSON");
00167         std::cerr << "Request for processor name returned "<answer<std::endl;
00168     }
00169
00170     ProcessLoop(instance);
00171
00172     std::cerr << "Stopping now" << std::endl;
00173     instance.Stop();
00174
00175     std::cerr << "End" << std::endl;
00176     return 0;
00177 }

```

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 *programe, [sound4::impact::dyn::CDynLib](#) &dynlib)
- void [MyLogger](#) ([sound4::impact::dyn::LogSeverity](#) severity, const std::string &msg)
- int [main](#) (int argc, const char **argv)
[Custom preset manager]

11.3.1 Function Documentation

11.3.1.1 GetEnv()

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

Definition at line 51 of file [sound4.impact.cl-pipe-dyn.cpp](#).

11.3.1.2 main()

```
int main (
    int argc,
    const char ** argv )
```

[Dynamic loading]

[Dynamic loading]

[Setup and sanity checks]

[Setup and sanity checks]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 194 of file [sound4.impact.cl-pipe-dyn.cpp](#).

11.3.1.3 MyLogger()

```
void MyLogger (
    sound4::impact::dyn::LogSeverity severity,
    const std::string & msg )
```

Definition at line 113 of file [sound4.impact.cl-pipe-dyn.cpp](#).

11.3.1.4 ProcessLoop()

```
void ProcessLoop (
    sound4::impact::dyn::CInstance & instance )
```

[Processing Loop]

[Processing Loop]

Definition at line 60 of file [sound4.impact.cl-pipe-dyn.cpp](#).

11.3.1.5 Usage()

```
void Usage (
    const char * progname,
    sound4::impact::dyn::CDynLib & dynlib )
```

Definition at line 91 of file [sound4.impact.cl-pipe-dyn.cpp](#).

11.4 sound4.impact.cl-pipe-dyn.cpp

[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
00119 class MyPresetManager : public sound4::impact::dyn::CPresetLoader {
00120 public:
00121     MyPresetManager() = default;
00122     virtual ~MyPresetManager() = default;
00123
00124     bool SetupPath(const std::filesystem::path& a_preset_path)
00125     {
00126         m_preset_path=a_preset_path;
00127         if ( !std::filesystem::exists(m_preset_path) ) {
00128             return false;
00129         }
00130         std::error_code ec;
00131         if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00132             return false;
00133         }
00134         return true;
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         for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
std::filesystem::directory_options::skip_permission_denied)) {
00157             if (entry.is_regular_file()) {
00158                 list.push_back(entry.path().lexically_relative(m_preset_path));
00159             }
00160         }
00161         return list;
00162     }
00163     virtual std::string Read(const std::filesystem::path &filename) override
00164     {
00165         std::ifstream f(GetRealPath(filename));
00166         if ( f.fail() ) {
00167             return std::string();
00168         }
00169         std::ostringstream ss;
00170         ss << f.rdbuf();
00171         return ss.str();
00172     }
00173     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00174     {
00175         std::ofstream f(GetRealPath(filename));
00176         if ( f.fail() ) {
00177             return false;
00178         }
00179         f << content;
00180         if ( f.fail() ) {
00181             return false;
00182         }
00183         return true;
00184     }
00185 private:
00186     std::filesystem::path m_preset_path;
00187     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00188     {
00189         return m_preset_path / name;
00190     }
00191 };
00192
00193
00194 int main(int argc, const char **argv)
00195 {
00196     #if (IMPACT_HAS_WEBSERVER)
00197         int listen_port = 8080;
00198     #endif // (IMPACT_HAS_WEBSERVER)
00199     auto store_path_str=GetEnv("S4STORE_PATH");
00200
00201     // Get credentials from environment
00202     auto LoginKey=GetEnv("S4LOGINKEY");
00203     if (LoginKey.empty()) {
00204         LoginKey=GetEnv("S4USERID");
00205     }
00206     auto RadioName=GetEnv("RADIO_NAME");
00207     auto KeyId=GetEnv("ACCESS_KEY_ID");
00208     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00209
00210     const char *progname=argv[0];
00211
00212
00213     sound4::impact::dyn::CDynLib dynlib;
00214     if (!dynlib.Load()) {
00215         std::cerr << "Failed to load processing library" << std::endl;
00216         return 1;
00217     }
00218
00219     if (argc>1) {
00220         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00221             Usage(progname, dynlib);
00222             return 0;
00223         }
00224     }
00225
00226     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00227         std::cerr << "Missing credentials in environment" << std::endl;
00228         Usage(progname, dynlib);
00229         return 1;
00230     }
00231
00232     std::filesystem::path store_path;
00233     if (!store_path_str.empty()) {
00234         store_path = std::filesystem::path(store_path_str);
00235         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00236             std::cerr << "Storage folder '"<store_path.c_str()<"' is not a valid directory" <<
std::endl;

```

```

00237         Usage(progname, dynlib);
00238         return 1;
00239     }
00240 }
00241
00243     sound4::impact::dyn::SetLogSeverity(dynlib, verbose5);
00244     sound4::impact::dyn::SetLoggerCallback(dynlib, MyLogger);
00245
00246     sound4::impact::dyn::CInstance instance(dynlib);
00247     instance.SetParam("ADMIN_USER", "admin");
00248     instance.SetParam("ADMIN_SECRET", "admin");
00250
00252     MyPresetManager preset_manager;
00253     if (!store_path.empty()) {
00254         if (!preset_manager.SetupPath(store_path)) {
00255             std::cerr << "Failed to setup the custom preset manager on path " << store_path.c_str() <<
std::endl;
00256             return 1;
00257         }
00258         instance.SetPresetManager(&preset_manager);
00259     }
00261
00262     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     }
00269
00270     #if (IMPACT_HAS_WEBSERVER)
00271     std::cerr << "Starting web server" << std::endl;
00273     if (!instance.StartWebServer(listen_port, 0)) {
00274         std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00275         return 1;
00276     }
00278     #endif // (IMPACT_HAS_WEBSERVER)
00279
00280     {
00282         auto client = instance.NewClient();
00283         auto answer = client->ProcessJson(R"JSON({ "get": { "processorname": null } })JSON");
00284         std::cerr << "Request for processor name returned " << answer << std::endl;
00286     }
00287
00288     ProcessLoop(instance);
00289
00290     std::cerr << "Stopping now" << std::endl;
00291     instance.Stop();
00292
00293     std::cerr << "End" << std::endl;
00294     return 0;
00295 }

```

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 }
```

```

00038     #if defined(_WIN32) || defined(WIN32)
00039         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00040         _setmode(_fileno(stdin), _O_BINARY);
00041         _setmode(_fileno(stdout), _O_BINARY);
00042     #endif
00043
00044     while (!stopthread)
00045     {
00046         // Read a chunk from stdin
00047         if (std::fread(&audio_in[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdin) !=
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) !=
CHUNK_SIZE*CHANNEL_COUNT) {
00057             std::cerr << "Failed to write audio output" << std::endl;
00058             stopthread=true;
00059         }
00060     }
00061 }
00062
00063
00064 void Usage(const char *programe)
00065 {
00066     std::cerr << "Usage: " << programe << 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 - |
" << "programe" << " | 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 - | " << "programe" << " | 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 - |
" << "programe" << " | 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 *programe=argv[0];
00104
00105     if (argc>1) {
00106         if ((strcmp(argv[1], "-h")==0) || (strcmp(argv[1], "--help")==0)) {
00107             Usage(programe);
00108             return 0;
00109         }
00110     }
00111     if (!pLoginKey || !pRadioName || !pKeyId || !pAccessKey) {
00112         std::cerr << "Missing credentials in environment" << std::endl;
00113         Usage(programe);
00114         return 1;
00115     }
00116     std::filesystem::path fs_store_path;
00117     if (!store_path_str || store_path_str[0]==0) {
00118         fs_store_path=std::filesystem::current_path() / "state";
00119     } else {

```

```

00120     fs_store_path = std::filesystem::path(store_path_str);
00121 }
00122 if (!std::filesystem::exists(fs_store_path) || !std::filesystem::is_directory(fs_store_path)) {
00123     std::cerr << "Storage folder '" << fs_store_path.c_str() << "' is not a valid directory" << std::endl;
00124     Usage(progname);
00125     return 1;
00126 }
00127 std::string store_path;
00128 #ifdef _WIN32
00129 {
00130     // Convert path to UTF-8
00131     std::wstring wstr(fs_store_path.wstring());
00132     if (wstr.empty()) return {};
00133     int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
00134     std::string str(size, 0);
00135     WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00136     store_path=str;
00137 }
00138 #else // !_WIN32
00139     store_path=fs_store_path;
00140 #endif // !_WIN32
00141
00142 // Optional: Use our own logging, instead of default color output to stderr.
00143 //impact_SetLoggerCallback(&MyLogger);
00144
00145 //impact_SetLogSeverity(impact_LogSeverity::info);
00146 //impact_SetLogSeverity(impact_LogSeverity::verbose5);
00147
00148 std::cerr << "Process version: " << impact_GetVersion() << std::endl;
00149 // Do some sanity checks
00150 if (impact_GetSampleRate() != SAMPLE_RATE) {
00151     std::cerr << "Bad library sampling rate, expecting '" << SAMPLE_RATE << "' but has " <<
00152     impact_GetSampleRate() << std::endl;
00153     return 1;
00154 }
00155 if (impact_GetChunkSizeInFrames() != CHUNK_SIZE) {
00156     std::cerr << "Bad library frame size, expecting '" << CHUNK_SIZE << "' but has " <<
00157     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 " <<
00162     impact_GetChannelCount() << std::endl;
00163     return 1;
00164 }
00165
00166 // Setup parameters
00167 struct impact_CParameters *params = impact_NewParameters();
00168 // Optional: activate web authentication, set admin password
00169 // This creates the authentication only for a new storage path
00170 impact_SetParameter(params, "ADMIN_USER", "admin");
00171 impact_SetParameter(params, "ADMIN_SECRET", "admin");
00172
00173 // Create the processing instance
00174 pInstance=impact_InitProcess2(pLoginKey, pRadioName, pKeyId, pAccessKey, store_path.c_str(), params);
00175 impact_FreeParameters(params);
00176 if (!pInstance) {
00177     std::cerr << "Failed to create the process instance" << std::endl;
00178     return 1;
00179 }
00180
00181 // Start the update thread, without JSON server (low priority)
00182 std::thread updater([pInstance]() {
00183     // will returned when stopped or destroyed
00184     std::cerr << "impact_StartUpdateThread (no json port)" << std::endl;
00185     impact_StartUpdateThread(pInstance, 0);
00186     std::cerr << "impact_StartUpdateThread END" << std::endl;
00187 });
00188
00189 // This is not mandatory, but better be sure we are ready right now
00190 std::cerr << "Waiting update thread" << std::endl;
00191 if (impact_WaitUpdateThreadReady(pInstance, 5000) < 0) {
00192     std::cerr << "Timeout to create the process instance" << std::endl;
00193 }
00194 std::cerr << "Waiting update thread DONE" << std::endl;
00195
00196 #if (IMPACT_HAS_WEBSERVER)
00197 // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
00198 std::cerr << "Starting web server" << std::endl;
00199 int listenport= 8080;
00200 int listenport_secure= 8443;
00201 uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
00202 if (listenport || listenport_secure) {
00203     webserver = impact_Webserver(listenport,listenport_secure,pInstance);
00204     if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00205         std::cerr << "Starting web server FAILED" << std::endl;
00206     } else {
00207

```

```

00208         int web_status=impact_Webserver_Status(webserver);
00209         if (listenport) {
00210             if (web_status & SOUND4_WEBSEVER_HTTP_OK) {
00211                 std::cerr << "Web server listening on port " << listenport;
00212             } else {
00213                 std::cerr << "Web server failed to listen on port " << listenport;
00214             }
00215         }
00216         if (listenport_secure) {
00217             if (web_status & SOUND4_WEBSEVER_HTTPS_OK) {
00218                 std::cerr << "Web server listening on secure port " << listenport_secure;
00219             } else {
00220                 std::cerr << "Web server failed to listen on secure port " << listenport_secure;
00221             }
00222         }
00223         std::cerr << "Starting web server DONE" << std::endl;
00224     }
00225 }
00227 #endif // (IMPACT_HAS_WEBSEVER)
00228
00229     // Force load a factory preset now
00230     // Could be used instead of setting a store_path
00231     /*{
00232         impact_CClientInstance *pClient=impact_NewClient(pInstance);
00233         const char *answer=impact_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }",
00234         NULL);
00235         std::cerr << "Loaded factory preset answer: " << answer << std::endl;
00236         impact_FreeJsonAnswer(answer);
00237         impact_DeleteClient(pClient);
00238     }*/
00239
00240     std::cerr << "Estimated delay " << ( impact_GetEstimatedDelay(pInstance)*1000.0/SAMPLE_RATE ) << "
ms" << std::endl;
00241
00242     ProcessLoop(pInstance);
00243
00245 #if (IMPACT_HAS_WEBSEVER)
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_WEBSEVER)
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 }

```

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 178 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         if ( !std::filesystem::exists(m_preset_path ) ) {
00112             return false;
00113         }
00114         std::error_code ec;
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         for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
std::filesystem::directory_options::skip_permission_denied)) {
00141             if (entry.is_regular_file()) {
00142                 list.push_back(entry.path().lexically_relative(m_preset_path));
00143             }
00144         }
00145         return list;
00146     }
00147     virtual std::string Read(const std::filesystem::path &filename) override
00148     {
00149         std::ifstream f(GetRealPath(filename));
00150         if ( f.fail() ) {
00151             return std::string();
00152         }
00153         std::ostringstream ss;
00154         ss << f.rdbuf();
00155         return ss.str();
00156     }
00157     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00158     {
00159         std::ofstream f(GetRealPath(filename));
00160         if ( f.fail() ) {
00161             return false;
00162         }
00163         f << content;
00164         if ( f.fail() ) {
00165             return false;
00166         }
00167         return true;
00168     }
00169 private:
00170     std::filesystem::path m_preset_path;
00171     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00172     {
00173         return m_preset_path / name;
00174     }
00175 };
00176
00177
00178 int main(int argc, const char **argv)
00179 {
00180     int listen_port = 8080;
00181     auto store_path_str=GetEnv("S4STORE_PATH");
00182     std::string libname = "libsound4.impact.cl.so";
00183
00184     // Get credentials from environment
00185     auto LoginKey=GetEnv("S4LOGINKEY");
00186     if (LoginKey.empty()) {
00187         LoginKey=GetEnv("S4USERID");
00188     }
00189     auto RadioName=GetEnv("RADIO_NAME");
00190     auto KeyId=GetEnv("ACCESS_KEY_ID");

```

```

00194     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00195
00196     const char *programe=argv[0];
00197     for (int n=1;n<argc;n++) {
00198         std::string arg = argv[n];
00199         if (arg=="-p" || arg=="--port") {
00200             if (argc<n+1) {
00201                 std::cerr << "--port needs a number" << std::endl;
00202                 return 1;
00203             }
00204             n++;
00205             listen_port=atoi(argv[n]);
00206         } else if (arg=="-l" || arg=="--lib") {
00207             if (argc<n+1) {
00208                 std::cerr << "--port needs a number" << std::endl;
00209                 return 1;
00210             }
00211             n++;
00212             libname=argv[n];
00213         } else if (arg=="-h" || arg=="--help") {
00214             std::cerr << "Usage: " << programe << " [-p port] [-l libname.so]" << std::endl;;
00215             std::cerr << "  Reads audio from stdin and return processed audio in stdout" << std::endl;
00216             std::cerr << "  -l, --lib    : library to load [" << libname << "]" << std::endl;
00217             std::cerr << "  -p, --port   : port for web server [" << listen_port << "]" << std::endl;
00218             std::cerr << std::endl;
00219             std::cerr << "Environment:" << std::endl;;
00220             std::cerr << "  - S4STORE_PATH          : path to store config [default: (no storage)]" <<
std::endl;;
00221             std::cerr << "  - S4LOGINKEY            : SOUND4 Login key (or S4USERID)" << std::endl;;
00222             std::cerr << "  - RADIO_NAME           : SOUND4 Radio Name" << std::endl;;
00223             std::cerr << "  - ACCESS_KEY_ID        : SOUND4 Access Key ID" << std::endl;;
00224             std::cerr << "  - ACCESS_KEY_SECRET    : SOUND4 Access Key Secret" << std::endl;
00225             std::cerr << std::endl;
00226             std::cerr << "Example:" << std::endl;
00227             std::cerr << "  ffmpeg -re -i file.wav -ar 48000 -ac 2 -f f32le - | " << programe << " | ffplay
-f f32le -ar 48000 -ac 2 -i -" << std::endl;;
00228             #if defined(_WIN32) || defined(WIN32)
00229             std::cerr << "  ffmpeg -f dshow -i audio=\\Microphone\\" -ar 48000 -ac 2 -f f32le - |
" << programe << " | ffplay -f f32le -ar 48000 -ac 2 -i -" << std::endl;;
00230             #else
00231             std::cerr << "  ffmpeg -f pulse -i default -ar 48000 -ac 2 -f f32le - | " << programe << " |
ffplay -f f32le -ar 48000 -ac 2 -i -" << std::endl;;
00232             #endif
00233             return 0;
00234         } else {
00235             std::cerr << "unknown argument '" << arg << "'" << std::endl;
00236             return 1;
00237         }
00238     }
00239
00240
00242     sound4::CProcessor processor;
00243     if (!processor.Load(libname)) {
00244         std::cerr << "Failed to load processing library" << std::endl;
00245         return 1;
00246     }
00247
00249     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00250         std::cerr << "Missing credentials in environment" << std::endl;
00251         return 1;
00252     }
00253     std::filesystem::path store_path;
00254     if (!store_path_str.empty()) {
00255         store_path = std::filesystem::path(store_path_str);
00256         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00257             std::cerr << "Storage folder '" << store_path.c_str() << "' is not a valid directory" <<
std::endl;
00258             return 1;
00259         }
00260     }
00261
00263     processor.SetLogSeverity(sound4::verbose5);
00264     processor.SetLoggerCallback([&](sound4::LogSeverity severity, const std::string& msg) {
00265         std::cerr << msg << std::endl;
00266     });
00267
00268     sound4::CInstance instance(processor);
00269     instance.SetParam("ADMIN_USER", "admin");
00270     instance.SetParam("ADMIN_SECRET", "admin");
00271
00272     MyPresetManager preset_manager;
00273     if (!store_path.empty()) {
00274         if (!preset_manager.SetupPath(store_path)) {
00275             std::cerr << "Failed to setup the custom preset manager on path " << store_path.c_str() <<
std::endl;
00276             return 1;
00277         }
00278     }
00279

```

```

00280         instance.SetPresetManager(&preset_manager);
00281     }
00282
00283
00284     std::cerr << "Creating" << std::endl;
00285     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
00286         std::cerr << "Failed to create the processing instance" << std::endl;
00287         return 1;
00288     }
00289
00290
00291
00292     std::cerr << "Starting web server" << std::endl;
00293     if (!instance.StartWebServer(listen_port, 0)) {
00294         std::cerr << "Failed to create the web server on port " << listen_port << std::endl;
00295         return 1;
00296     }
00297
00298
00299
00300     {
00301         auto client = instance.NewClient();
00302         auto answer = client->ProcessJson(R"JSON({ "get": { "processorname": null } })JSON");
00303         std::cerr << "Request for processor name returned " << answer << std::endl;
00304     }
00305
00306
00307     ProcessLoop(instance, processor);
00308
00309
00310     std::cerr << "Stopping now" << std::endl;
00311     instance.Stop();
00312
00313
00314     std::cerr << "End" << std::endl;
00315     return 0;
00316 }

```

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 // -----
00147 // -----
00148
00149 // -----
00161 #define IMPACT_AUDIOFRAME_COUNT 64
00167 #define IMPACT_CHANNEL_COUNT 2
00173 #define IMPACT_AUDIO_INPUT_COUNT 1
00179 #define IMPACT_AUDIO_OUTPUT_COUNT 1
```

```
00185 #define IMPACT_SAMPLE_RATE          48000
00186
00192 struct SOUND4_ProcessInfo {
00193     size_t struct_size;
00194     const char *name;
00195     const char *prefix;
00196     const char *shortname;
00197     const char *version;
00198 };
00199
00207 IMPACT_DECL const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo();
00208
00214 IMPACT_DECL const char *impact_GetVersion();
00215
00228 IMPACT_DECL unsigned int impact_GetChunkSizeInFrames();
00229
00238 IMPACT_DECL unsigned int* impact_GetPossibleChunkSizeInFrames();
00239
00247 IMPACT_DECL unsigned int impact_GetChannelCount();
00248
00257 IMPACT_DECL unsigned int impact_GetAudioInputCount();
00258
00267 IMPACT_DECL unsigned int impact_GetAudioOutputCount();
00268
00276 IMPACT_DECL unsigned int impact_GetSampleRate();
00277
00281 enum impact_LogSeverity {
00282     none = 0,
00283     fatal = 1,
00284     error = 2,
00285     warning = 3,
00286     info = 4,
00287     verbose = 5,
00288     verbose2 = 6,
00289     verbose3 = 7,
00290     verbose4 = 8,
00291     verbose5 = 9,
00292 };
00293
00300 typedef void (*impact_loggerfn) (enum impact_LogSeverity, const char *);
00301
00311 IMPACT_DECL void impact_SetLoggerCallback( impact_loggerfn logger );
00312
00320 IMPACT_DECL void impact_SetLogSeverity( enum impact_LogSeverity severity );
00321
00324 // -----
00330 // -----
00334 struct impact_CInstance;
00335
00339 struct impact_CParameters;
00340
00348 IMPACT_DECL struct impact_CParameters *impact_NewParameters();
00349
00353 IMPACT_DECL void impact_FreeParameters(struct impact_CParameters *params);
00354
00366 IMPACT_DECL void impact_SetParameter(struct impact_CParameters *params, const char *name, const char
    *value);
00367
00373 IMPACT_DECL const char *impact_GetParameter(struct impact_CParameters *params, const char *name);
00374
00380 IMPACT_DECL void impact_FreeParameterValue(const char *value);
00381
00389 IMPACT_DECL struct impact_CInstance *impact_InitProcess(
00390     const char *LoginKey,
00391     const char *RadioName,
00392     const char *Access_Key_ID,
00393     const char *Access_Key_Secret,
00394     const char *save_path
00395 );
00396
00404 IMPACT_DECL struct impact_CInstance *impact_InitProcess2(
00405     const char *LoginKey,
00406     const char *RadioName,
00407     const char *Access_Key_ID,
00408     const char *Access_Key_Secret,
00409     const char *save_path,
00410     const struct impact_CParameters *parameters
00411 );
00412
00457 IMPACT_DECL struct impact_CInstance *impact_InitProcess3(
00458     const char *LoginKey,
00459     const char *RadioName,
00460     const char *Access_Key_ID,
00461     const char *Access_Key_Secret,
00462     const char *save_path,
00463     const struct impact_CParameters *parameters,
00464     unsigned int frames_per_chunk
```

```

00465     );
00466
00478 IMPACT_DECL unsigned int impact_GetProcessChunkFrames(struct impact_CInstance *instance);
00479
00489 IMPACT_DECL int impact_TerminateProcess(struct impact_CInstance *instance);
00490
00496 IMPACT_DECL void impact_ExitProcess(struct impact_CInstance *instance);
00497
00500 #if IMPACT_HAS_CLOUDBUS
00501 // -----
00510 struct impact_CBus;
00511
00517 IMPACT_DECL struct impact_CBus* impact_NewBus();
00518
00524 IMPACT_DECL void impact_FreeBus(struct impact_CBus *bus);
00525
00535 IMPACT_DECL void impact_SetInstanceBus(struct impact_CParameters *params, struct impact_CBus *bus);
00536
00537
00539 #endif // IMPACT_HAS_CLOUDBUS
00540
00541 // -----
00564 // -----
00565
00566 // OS dependent filesystem path char type
00567 #ifdef _WIN32
00568     typedef wchar_t fs_char;
00569 #else // !_WIN32
00570     typedef char fs_char;
00571 #endif // !_WIN32
00572
00580 typedef char * (*impact_storage_reader) (const fs_char *filename, void* userdata);
00581
00588 typedef void (*impact_storage_reader_free) (char *content, void* userdata);
00589
00598 typedef int (*impact_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00599
00607 typedef int (*impact_storage_exists) (const fs_char *filename, void* userdata);
00608
00616 typedef fs_char** (*impact_storage_getall) (void* userdata);
00617
00624 typedef void (*impact_storage_getall_free) (fs_char**all, void* userdata);
00625
00633 typedef int (*impact_storage_remove) (const fs_char *filename, void* userdata);
00634
00643 typedef int (*impact_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00644
00662 IMPACT_DECL void impact_SetPresetManager(struct impact_CParameters *params
00663     , impact_storage_reader reader, impact_storage_reader_free
00664     , impact_storage_writer writer
00665     , impact_storage_exists exists
00666     , impact_storage_getall getall, impact_storage_getall_free getall_free
00667     , impact_storage_remove remove
00668     , impact_storage_rename rename
00669     , int IsReadOnly
00670     , void* userdata
00671 );
00672
00676 enum impact_PresetChange_Kind {
00677     change_kind_created = 1,
00678     change_kind_modified ,
00679     change_kind_deleted ,
00680 };
00681
00690 IMPACT_DECL void impact_PresetManager_InformChange(struct impact_CInstance *instance, const fs_char
    *relative_path, enum impact_PresetChange_Kind change_kind);
00691
00694 // -----
00702 // -----
00703
00720 IMPACT_DECL void impact_SetMetadata(struct impact_CInstance *instance, const char* key, const char*
    value);
00721
00732 IMPACT_DECL void impact_SetMetadataMulti(struct impact_CInstance *instance, const char** keyvalue);
00733
00741 IMPACT_DECL const char** impact_GetMetadataInfos(struct impact_CInstance *instance);
00742
00749 IMPACT_DECL void impact_FreeMetadataInfos(struct impact_CInstance *instance, const char** infos);
00750
00753 // -----
00758 // -----
00759
00772 IMPACT_DECL void impact_StartUpdateThread(struct impact_CInstance *instance, unsigned int port);
00773
00779 IMPACT_DECL void impact_StopUpdateThread(struct impact_CInstance *instance);
00780
00788 IMPACT_DECL int impact_WaitUpdateThreadReady(struct impact_CInstance *instance, int milliseconds);

```

```

00789
00792 // -----
00797 // -----
00819 IMPACT_DECL void impact_ProcessAudio(struct impact_CInstance *instance, const float *input, float
    *output );
00820
00839 IMPACT_DECL void impact_ProcessAudio_Planar(struct impact_CInstance *instance
00840     , float const * const *input
00841     , float * const *output);
00842
00855 IMPACT_DECL float *impact_GetBufferIn(struct impact_CInstance *instance);
00856
00869 IMPACT_DECL float *impact_GetBufferOut(struct impact_CInstance *instance);
00870
00878 IMPACT_DECL unsigned int impact_GetEstimatedDelay(struct impact_CInstance *instance);
00879
00882 // -----
00889 // -----
00890 #ifdef _MSC_VER
00891     #define __BYTE_ORDER__ REG_DWORD
00892     #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00893     #define __ORDER_BIG_ENDIAN__ REG_DWORD_BIG_ENDIAN
00894 #endif
00895
00899 enum impact_SampleFormat {
00900     INVALID_FORMAT=0,
00901     S16_LE,
00902     S16_BE,
00903     S24_LE,
00904     S24_BE,
00905     S32_LE,
00906     S32_BE,
00907     F32_LE,
00908     F32_BE,
00909
00910     // Native definitions, only links to the physical ones
00911     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00912         S16_NATIVE=S16_LE,
00913         S24_NATIVE=S24_LE,
00914         S32_NATIVE=S32_LE,
00915         F32_NATIVE=F32_LE,
00916     #else
00917         S16_NATIVE=S16_BE,
00918         S24_NATIVE=S24_BE,
00919         S32_NATIVE=S32_BE,
00920         F32_NATIVE=F32_BE,
00921     #endif
00922 };
00923
00930 IMPACT_DECL const char *impact_GetFormatName(const enum impact_SampleFormat fmt);
00931
00938 IMPACT_DECL enum impact_SampleFormat impact_GetFormatFromName(const char *name);
00939
00946 IMPACT_DECL unsigned int impact_GetBytesFromFormat(const enum impact_SampleFormat fmt);
00947
00948
00955 IMPACT_DECL unsigned int impact_GetMaxPacketFrame(struct impact_CInstance *instance);
00956
00967 IMPACT_DECL unsigned int impact_AddAudio(struct impact_CInstance *instance, const uint8_t *payload,
    unsigned int nFrame, enum impact_SampleFormat fmt );
00968
00975 IMPACT_DECL unsigned int impact_AddPadAudio(struct impact_CInstance *instance);
00976
00983 IMPACT_DECL unsigned int impact_GetOutputCount(struct impact_CInstance *instance);
00984
00994 IMPACT_DECL unsigned int impact_GetAudio(struct impact_CInstance *instance, uint8_t *payload, unsigned
    int max_nFrame, enum impact_SampleFormat fmt );
00995
00999 // -----
01003 // -----
01014 IMPACT_DECL void impact_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
    impact_SampleFormat fmt);
01015
01026 IMPACT_DECL void impact_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
    impact_SampleFormat fmt);
01027
01037 IMPACT_DECL void impact_StereoToMono(const float *input, float *output, size_t nFrame);
01038
01048 IMPACT_DECL void impact_MonoToStereo(const float *input, float *output, size_t nFrame);
01049
01060 IMPACT_DECL void impact_StereoToMono_Planar(const float *inputL, const float *inputR, float *output,
    size_t nFrame);
01061
01072 IMPACT_DECL void impact_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t
    nFrame);
01073
01080 IMPACT_DECL void impact_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);

```

```

01081
01088 IMPACT_DECL void impact_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
01089
01092 // -----
01097 // -----
01101 struct impact_CClientInstance;
01102
01111 IMPACT_DECL struct impact_CClientInstance *impact_NewClient(struct impact_CInstance *instance);
01112
01118 IMPACT_DECL void impact_DeleteClient(struct impact_CClientInstance *client);
01119
01133 IMPACT_DECL const char *impact_ProcessJson(struct impact_CClientInstance *client, const char
*json_str, int *need_save);
01134
01140 IMPACT_DECL void impact_FreeJsonAnswer(const char *json_str);
01141
01153 IMPACT_DECL int impact_SaveState(struct impact_CInstance *instance);
01154
01450 #if IMPACT_HAS_WEBSERVER
01451 // -----
01459 // -----
01478 IMPACT_DECL uint64_t impact_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure,
const char *socket_ip, unsigned int socket_port);
01479
01480 // -----
01500 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);
01501
01519 IMPACT_DECL uint64_t impact_Webserver(unsigned int listenport, unsigned int listenport_secure, struct
impact_CInstance *instance);
01520
01530 IMPACT_DECL int impact_Webserver_Stop(uint64_t id, int timeout_ms);
01531
01540 IMPACT_DECL int impact_Webserver_Status(uint64_t id);
01541
01552 IMPACT_DECL void impact_Webserver_SetAppHealth(struct impact_CInstance *instance, int httpcode, const
char *contenttype, const char *content);
01553
01562 IMPACT_DECL void impact_Webserver_GetAppHealth(struct impact_CInstance *instance, int* httpcode,
char** contenttype, char** content);
01563
01569 IMPACT_DECL void impact_Webserver_FreeString(char *str);
01570
01575 #define SOUND4_INVALID_WEBSERVER_ID ((uint64_t)-1)
01576
01580 #define SOUND4_WEBSERVER_HTTP_OK (1<0)
01584 #define SOUND4_WEBSERVER_HTTPS_OK (1<1)
01585
01587 #endif // IMPACT_HAS_WEBSERVER
01588
01589 #ifdef __cplusplus
01590 } // extern "C"
01591 #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_shortcode = "impact";
00053     using SampleFormat = impact_SampleFormat;
00054     const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
00061     const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
00065     const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
00069     const size_t InputSampleSize=ChunkFrames*InputChannels;
00073     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00077     const size_t SampleRate=IMPACT_SAMPLE_RATE;
00078
00082     namespace helper {
00083         #ifdef _WIN32
00089         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00090             if (wstr.empty()) return {};
00091             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
NULL);
00092             std::string str(size, 0);
00093             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00094             return str;
00095         }
00101         static inline std::wstring UTF8ToWString(const std::string& str) {
00102             if (str.empty()) return std::wstring();
00103             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00104             std::wstring wstr(size, 0);
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
00132     static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
00133 impact_SampleFormat fmt) {
00134         impact_AudioConvertFrom(payload, output, nSpl, fmt);
00135     }
00136     static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl,
00137 impact_SampleFormat fmt) {
00138         impact_AudioConvertTo(input, payload, nSpl, fmt);
00139     }
00140
00141     static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
00142         impact_StereoToMono(input, output, nSpl);
00143     }
00144     static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
00145         impact_MonoToStereo(input, output, nSpl);
00146     }
00147     static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float
00148 *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) {
00153         impact_MonoToStereo_Planar(input, outputL, outputR, nSpl);
00154     }
00155     static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
00156         impact_AudioMonoFromLiveStereo(payload, output);
00157     }
00158     static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
00159         impact_AudioMonoToLiveStereo(input, payload);
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) {

```

```

00273         throw std::runtime_error("Bad library input count");
00274     }
00275     if (impact_GetAudioOutputCount() != IMPACT_AUDIO_OUTPUT_COUNT) {
00276         throw std::runtime_error("Bad library output count");
00277     }
00278 }
00279
00280 static inline std::string GetFormatName(const SampleFormat fmt)
00281 {
00282     return std::string(impact_GetFormatName(fmt));
00283 }
00284
00285 static inline SampleFormat GetFormatFromName(const std::string& name)
00286 {
00287     return impact_GetFormatFromName(name.c_str());
00288 }
00289
00290 static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
00291 {
00292     return impact_GetBytesFromFormat(fmt);
00293 }
00294
00295 using LogSeverity = impact_LogSeverity;
00296 using log_cb_t = std::function<void(LogSeverity, const std::string&)>;
00297
00298 static log_cb_t _log_cb;
00299 extern "C" {
00300     static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
00301         _log_cb(severity, std::string(c_msg));
00302     }
00303 }
00304
00305 static inline void SetLogSeverity(LogSeverity severity)
00306 {
00307     impact_SetLogSeverity(severity);
00308 }
00309
00310 static inline void SetLoggerCallback(log_cb_t cb)
00311 {
00312     _log_cb=cb;
00313     impact_SetLoggerCallback(&_log_cb_c);
00314 }
00315
00316 #if IMPACT_HAS_CLOUDBUS
00317 class CBus {
00318 public:
00319     CBus()
00320         : m_bus(impact_NewBus()) {}
00321     ~CBus()
00322     {
00323         impact_FreeBus(m_bus);
00324         m_bus = nullptr;
00325     }
00326     impact_CBus *Get() const { return m_bus; }
00327 private:
00328     impact_CBus *m_bus;
00329 };
00330 #endif // IMPACT_HAS_CLOUDBUS
00331
00332 class CPresetLoader {
00333 public:
00334     CPresetLoader() = default;
00335     virtual ~CPresetLoader() = default;
00336
00337     virtual bool IsReadOnly() = 0;
00338     virtual bool Exists(const std::filesystem::path &name) = 0;
00339     virtual bool Remove(const std::filesystem::path &name) = 0;
00340     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00341     virtual std::vector<std::filesystem::path> GetAll() = 0;
00342     virtual std::string Read(const std::filesystem::path &filename) = 0;
00343     virtual bool Write(const std::filesystem::path &filename, const std::string &content) = 0;
00344 };
00345
00346 using PresetChange_Kind = impact_PresetChange_Kind;
00347
00348 class CInstance {
00349 public:
00350     CInstance()
00351     {
00352         params = impact_NewParameters();
00353         if (!params) throw std::bad_alloc();
00354     }
00355     // non-copyable
00356     CInstance( const CInstance& ) = delete;
00357     CInstance& operator=( const CInstance& ) = delete;
00358     virtual ~CInstance()
00359     {
00360         Stop();
00361         if (params) impact_FreeParameters(params);
00362     }
00363 }

```

```

00409     bool IsOk() const { return instance!=nullptr; }
00417     void SetParam(const std::string& name, const std::string& value)
00418     {
00419         assert(params);
00420         impact_SetParameter(params, name.c_str(), value.c_str());
00421     }
00427     std::string GetParam(const std::string& name)
00428     {
00429         assert(params);
00430         auto c_value=impact_GetParameter(params, name.c_str());
00431         std::string ret(c_value);
00432         impact_FreeParameterValue(c_value);
00433         return ret;
00434     }
00435
00445     void SetPresetManager(CPresetLoader *preset_manager)
00446     {
00447         if (!preset_manager) return;
00448         assert(params);
00449         impact_SetPresetManager(params,
00450             impact_custom_reader,
00451             impact_custom_reader_free,
00452             impact_custom_writer,
00453             impact_custom_exists,
00454             impact_custom_getall,
00455             impact_custom_getall_free,
00456             impact_custom_remove,
00457             impact_custom_rename,
00458             preset_manager->IsReadOnly(),
00459             preset_manager
00460         );
00461     }
00462
00463 #if IMPACT_HAS_CLOUDBUS
00467     void SetBus(const CBus& bus) {
00468         assert(params);
00469         impact_SetInstanceBus(params,bus.Get());
00470     }
00471 #endif // IMPACT_HAS_CLOUDBUS
00472
00487     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)
00488     {
00489         assert(!instance);
00490         std::string l_save_path;
00491         if (!save_path.empty()) {
00492             l_save_path = save_path.u8string();
00493         }
00494         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);
00495         if (!instance) return false;
00496         if (!init_metadata.empty()) {
00497             for (auto&& [key,value]: init_metadata) {
00498                 impact_SetMetadata(instance, key.c_str(), value.c_str());
00499             }
00500             init_metadata.clear();
00501         }
00502         update_thread = std::thread([this, json_port]() {
00503             OnUpdateThreadStart();
00504             impact_StartUpdateThread(instance, json_port);
00505             OnUpdateThreadStop();
00506         });
00507         int timeout=5000;
00508 #ifdef DEBUG
00509             timeout=60*1000; // For debugger pauses
00510 #endif
00511         if (impact_WaitUpdateThreadReady(instance, timeout)<0) {
00512             return false;
00513         }
00514         return true;
00515     }
00516
00524     unsigned int GetChunkFrames() {
00525         if (instance) {
00526             return impact_GetProcessChunkFrames(instance);
00527         }
00528         return 0;
00529     }
00530
00538     void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00539     {
00540         impact_PresetManager_InformChange(instance, relative_path.c_str(),
(impact_PresetChange_Kind) change_kind);
00541     }
00542

```

```

00554 void SetMetadata(const std::string& key, const char* value)
00555 {
00556     if (instance) {
00557         impact_SetMetadata(instance, key.c_str(), value);
00558     } else {
00559         init_metadata.push_back( {key, value} );
00560     }
00561 }
00562
00572 void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00573 {
00574     if (instance) {
00575         typedef const char* pchar;
00576         pchar* keyvalue=new pchar[2*list.size()+1];
00577         size_t n=0;
00578         for (auto&& [key,value]: list) {
00579             keyvalue[2*n+0]=key.c_str();
00580             keyvalue[2*n+1]=value;
00581             n++;
00582         }
00583         keyvalue[2*n+0]=nullptr;
00584
00585         impact_SetMetadataMulti(instance, keyvalue);
00586
00587         delete[] keyvalue;
00588     } else {
00589         for (auto&& [key,value]: list) {
00590             if (value) {
00591                 init_metadata.push_back( {key, value} );
00592             }
00593         }
00594     }
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             impact_FreeMetadataInfos(instance, c_values);
00616         }
00617         return values;
00618     }
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),
impact_GetBufferOut(instance));
00770     }
00771
00784     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
00785     void ProcessAudio_Planar(float const * const *input, float * const *output)
00786     {
00787         assert(instance);
00788         impact_ProcessAudio_Planar(instance, input, output);
00789     }
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
00822     class CClient {
00823     public:
00824         CClient(impact_CInstance* instance)
00825         {
00826             assert(instance);
00827             client=impact_NewClient(instance);
00828             if (!client) throw std::bad_alloc();
00829         }
00830         // non-copyable
00831         CClient( const CInstance& ) = delete;
00832         CClient& operator=( const CInstance& ) = delete;
00833         ~CClient()
00834         {
00835             if (client) {
00836                 impact_DeleteClient(client);
00837                 client=nullptr;
00838             }
00839         }
00840         std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
00841         {
00842             assert(client);
00843             int need_save=0;
00844             const char *canswer = impact_ProcessJson(client, request.c_str(), &need_save);
00845             if (!canswer) return {};
00846             std::string answer(canswer);
00847             impact_FreeJsonAnswer (canswer);
00848             if (NeedSave) {
00849                 *NeedSave=(need_save!=0);
00850             }
00851             return answer;
00852         }
00853     private:
00854         impact_CClientInstance *client = nullptr;
00855     };
00856     std::shared_ptr<CClient> NewClient()
00857     {
00858         assert(instance);
00859         return std::make_shared<CClient>(instance);
00860     }
00861
00862     protected:
00863     template<typename T>
00864     unsigned int AddAudio(const T* payload, unsigned int nFrame)
00865     {
00866         assert(instance);
00867         return impact_AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
00868             helper::SampleFormat<T>::format);
00869     }
00870     template<typename T>
00871     unsigned int GetAudio(T* payload, unsigned int max_nFrame)
00872     {
00873         assert(instance);
00874         return impact_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
00875             helper::SampleFormat<T>::format);
00876     }
00877
00878     private:
00879         impact_CParameters* params = nullptr;
00880         impact_CInstance* instance = nullptr;
00881         std::thread update_thread;
00882         std::vector< std::pair<std::string, std::string> > init_metadata;
00883 #if IMPACT_HAS_WEBSERVER
00884         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
00885 #endif // IMPACT_HAS_WEBSERVER
00886     };
00887
00888     }; // namespace impact
00889     }; // namespace sound4
00890
00891     // bridge C callbacks to CPresetLoader
00892     extern "C" {
00893     static char *impact_custom_reader(const fs_char *filename, void* userdata) {
00894         sound4::impact::CPresetLoader
00895         *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00896         std::string content=preset_loader->Read(filename);
00897         return strdup(content.c_str());
00898     }
00899     static void impact_custom_reader_free(char *content, void* userdata) {
00900         if (!content) return;
00901         free(content);
00902     }
00903     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
00904         sound4::impact::CPresetLoader
00905         *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);

```

```

00921         auto res=preset_loader->Write(filename,content);
00922         return res?0:-1;
00923     }
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     }
00930     static fs_char** impact_custom_getall(void* userdata) {
00931         sound4::impact::CPresetLoader
00932         *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00933         auto res=preset_loader->GetAll();
00934         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
00935         for (size_t n=0;n<res.size();n++) {
00936             all[n]=fs_strdup(res[n].c_str());
00937         }
00938         all[res.size()]=nullptr;
00939         return all;
00940     }
00941     static void impact_custom_getall_free(fs_char** all, void* userdata) {
00942         if (!all) return;
00943         for (fs_char** one=all;*one;one++) {
00944             free(*one);
00945         }
00946         free(all);
00947     }
00948     static int impact_custom_remove(const fs_char *filename, void* userdata) {
00949         sound4::impact::CPresetLoader
00950         *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00951         auto res=preset_loader->Remove(filename);
00952         return res?0:-1;
00953     }
00954     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
00955         sound4::impact::CPresetLoader
00956         *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
00957         auto res=preset_loader->Rename(from,to);
00958         return res?0:-1;
00959     }
00960 }

```

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 #ifndef _WIN32
00039     #include <stringapiset.h>
00040     #include <wchar.h>
00042     #define fs_strdup wcsdup
00043 #else // !_WIN32
00045     #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
00073     namespace helper {
00074         // Storage definition, depending on OS
00075         #if UNIXLIKE
00076             using DynLib_t = void *;
00077             using DynFunc_t = void *;
00078         #else
00079             using DynLib_t = HMODULE;
00080             using DynFunc_t = FARPROC;
00081         #endif
00082
00083         template <typename T>
00084         class DynFuncHolder {
00085         private:
00086             DynFunc_t m_ptr = NULL;
00087         public:
00088             DynFuncHolder() = default;
00089             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00090             operator bool() const { return (m_ptr!=NULL); }
00091             bool IsOk() const { return m_ptr!=NULL; }
00092             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00093         };
00094
00095         class CDynLoader {
00096         private:
00097             DynLib_t m_lib {};
00098         public:
00099             CDynLoader() = default;
00100             CDynLoader(CDynLoader&&) = default;
00101             // No copy allowed
00102             CDynLoader(const CDynLoader&) = delete;
00103             CDynLoader& operator=(CDynLoader const&) = delete;
00104             operator bool() const {
00105                 return (m_lib!=NULL);
00106             }
00107             bool IsOk() const {
00108                 return (m_lib!=NULL);
00109             }
00110             #if UNIXLIKE
00111             void Close() {
00112                 if (IsOk()) {
00113                     #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00114                         dlclose(m_lib);
00115                     #endif // __SANITIZE_ADDRESS__
00116                     m_lib=NULL;
00117                 }
00118             }
00119             bool Load(const std::filesystem::path& dynlib)
00120             {
00121                 Close();
00122                 m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00123                 return IsOk();
00124             }
00125             template <typename T>
00126             DynFuncHolder<T> GetSymbol(const std::string& name) {
00127                 auto ptr=dlsym(m_lib, name.c_str());
00128                 if (!ptr) {
00129                     throw std::runtime_error("Missing function in library");
00130                 }
00131                 return DynFuncHolder<T>(ptr);
00132             }
00133             template <typename T>
00134             DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00135                 auto ptr=dlsym(m_lib, name.c_str());
00136                 return DynFuncHolder<T>(ptr);
00137             }
00138             static std::filesystem::path GetThisLibraryPath(void) {
00139                 static Dl_info info;
00140                 if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00141                 auto rp=realpath(info.dli_fname, NULL);
00142                 std::filesystem::path p(rp);
00143                 free(rp);
00144                 return p;
00145             }
00146             #else
00147             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_
00148             {
00149                 Close();
00150                 m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00151                 return IsOk();
00152             }
00153             void Close() {
00154                 if (IsOk()) {
00155                     // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
FreeLibraryAndExitThread but...

```

```

00168 // So define SOUND4_CALL_FREELIBRARY before including this file if you
really want to unload the library dynamically
00169 #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
00170     FreeLibraryAndExitThread(m_lib, 0);
00171 #elif defined(SOUND4_CALL_FREELIBRARY)
00172     FreeLibrary(m_lib);
00173 #endif // SOUND4_CALL_FREELIBRARY
00174 m_lib=NULL;
00175 }
00176 }
00177 template <typename T>
00178 DynFuncHolder<T> GetSymbol(const std::string& name) {
00179     auto ptr=GetProcAddress(m_lib, name.c_str());
00180     if (!ptr) {
00181         throw std::runtime_error("Missing function in library");
00182     }
00183     return DynFuncHolder<T>(ptr);
00184 }
00185 template <typename T>
00186 DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00187     auto ptr=GetProcAddress(m_lib, name.c_str());
00188     return DynFuncHolder<T>(ptr);
00189 }
00190 static std::filesystem::path GetThisLibraryPath(void) {
00191     static wchar_t path[MAX_PATH]={};
00192     HMODULE hm = NULL;
00193     if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00194         GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
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 ;

```

```

00260     helper::DynFuncHolder< decltype(impact_MonoToStereo) ) > MonoToStereo ;
00261     helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo ;
00262     helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) ) > AudioMonoToLiveStereo ;
00263     helper::DynFuncHolder< decltype(impact_NewClient) ) > NewClient ;
00264     helper::DynFuncHolder< decltype(impact_DeleteClient) ) > DeleteClient ;
00265     helper::DynFuncHolder< decltype(impact_ProcessJson) ) > ProcessJson ;
00266     helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) ) > FreeJsonAnswer ;
00267     helper::DynFuncHolder< decltype(impact_SaveState) ) > SaveState ;
00268 #if IMPACT_HAS_WEBSERVER
00269     helper::DynFuncHolder< decltype(impact_Webserver_tcp) ) > Webserver_tcp ;
00270     helper::DynFuncHolder< decltype(impact_Webserver_tcp2) ) > Webserver_tcp2 ;
00271     helper::DynFuncHolder< decltype(impact_Webserver) ) > Webserver ;
00272     helper::DynFuncHolder< decltype(impact_Webserver_Stop) ) > Webserver_Stop ;
00273     helper::DynFuncHolder< decltype(impact_Webserver_Status) ) > Webserver_Status ;
00274 #endif // IMPACT_HAS_WEBSERVER
00275
00276     // added 2023-03-22
00277     helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) ) > StereoToMono_Planar ;
00278     helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) ) > MonoToStereo_Planar ;
00279
00280     // added 2023-05-22
00281     helper::DynFuncHolder< decltype(impact_SetMetadata) ) > SetMetadata ;
00282     helper::DynFuncHolder< decltype(impact_GetMetadataInfos) ) > GetMetadataInfos ;
00283     helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) ) > FreeMetadataInfos ;
00284
00285     // added 2023-06-20
00286     helper::DynFuncHolder< decltype(impact_SetPresetManager) ) > SetPresetManager ;
00287
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
00307     helper::DynFuncHolder< decltype(impact_Webserver_GetAppHealth) ) > Webserver_GetAppHealth ;
00308
00308     helper::DynFuncHolder< decltype(impact_Webserver_FreeString) ) > Webserver_FreeString ;
00309 #endif // IMPACT_HAS_WEBSERVER
00310
00310     bool Load(const std::filesystem::path& path = {}) {
00311         std::string filename;
00312         #if defined(__APPLE__)
00313             filename="libsound4.impact.cl.dylib";
00314         #elif defined(__unix__)
00315             filename="libsound4.impact.cl.so";
00316         #else
00317             filename="sound4.impact.cl.dll";
00318         #endif
00319         m_lib.Close();
00320         // If a path is given, use it directly and do not try other path
00321         if (!path.empty() && !m_lib.Load(path / filename)) {
00322             return false;
00323         } else if (!m_lib.IsOk()) {
00324             auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00325             #ifndef _WIN32
00326             // Linux: if in a bin directory, try ../lib first
00327             if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00328                 auto libdir = thisdir.parent_path() / "lib";
00329                 m_lib.Load(libdir / filename);
00330             }
00331             #endif // !_WIN32
00332             if (!m_lib.IsOk()) {
00333                 // Search in the same directory this code is in
00334                 m_lib.Load(thisdir / filename);
00335             }
00336             if (!m_lib.IsOk()) {
00337                 // Try current path

```

```

00346         m_lib.Load(std::filesystem::current_path() / filename);
00347     }
00348     if (!m_lib.IsOk()) {
00349         return false;
00350     }
00351 }
00352 // Load all C functions
00353 try {
00354     GetVersion = m_lib.GetSymbol< decltype(impact_GetVersion) >
>("impact_GetVersion" );
00355     GetChunkSizeInFrames = m_lib.GetSymbol< decltype(impact_GetChunkSizeInFrames) >
>("impact_GetChunkSizeInFrames" );
00356     GetChannelCount = m_lib.GetSymbol< decltype(impact_GetChannelCount) >
>("impact_GetChannelCount" );
00357     GetAudioInputCount = m_lib.GetSymbol< decltype(impact_GetAudioInputCount) >
>("impact_GetAudioInputCount" );
00358     GetAudioOutputCount = m_lib.GetSymbol< decltype(impact_GetAudioOutputCount) >
>("impact_GetAudioOutputCount" );
00359     GetSampleRate = m_lib.GetSymbol< decltype(impact_GetSampleRate) >
>("impact_GetSampleRate" );
00360     SetLoggerCallback = m_lib.GetSymbol< decltype(impact_SetLoggerCallback) >
>("impact_SetLoggerCallback" );
00361     SetLogSeverity = m_lib.GetSymbol< decltype(impact_SetLogSeverity) >
>("impact_SetLogSeverity" );
00362     NewParameters = m_lib.GetSymbol< decltype(impact_NewParameters) >
>("impact_NewParameters" );
00363     FreeParameters = m_lib.GetSymbol< decltype(impact_FreeParameters) >
>("impact_FreeParameters" );
00364     SetParameter = m_lib.GetSymbol< decltype(impact_SetParameter) >
>("impact_SetParameter" );
00365     GetParameter = m_lib.GetSymbol< decltype(impact_GetParameter) >
>("impact_GetParameter" );
00366     FreeParameterValue = m_lib.GetSymbol< decltype(impact_FreeParameterValue) >
>("impact_FreeParameterValue" );
00367     InitProcess = m_lib.GetSymbol< decltype(impact_InitProcess) >
>("impact_InitProcess" );
00368     InitProcess2 = m_lib.GetSymbol< decltype(impact_InitProcess2) >
>("impact_InitProcess2" );
00369     TerminateProcess = m_lib.GetSymbol< decltype(impact_TerminateProcess) >
>("impact_TerminateProcess" );
00370     ExitProcess = m_lib.GetSymbol< decltype(impact_ExitProcess) >
>("impact_ExitProcess" );
00371     StartUpdateThread = m_lib.GetSymbol< decltype(impact_StartUpdateThread) >
>("impact_StartUpdateThread" );
00372     StopUpdateThread = m_lib.GetSymbol< decltype(impact_StopUpdateThread) >
>("impact_StopUpdateThread" );
00373     WaitUpdateThreadReady = m_lib.GetSymbol< decltype(impact_WaitUpdateThreadReady) >
>("impact_WaitUpdateThreadReady" );
00374     ProcessAudio = m_lib.GetSymbol< decltype(impact_ProcessAudio) >
>("impact_ProcessAudio" );
00375     ProcessAudio_Planar = m_lib.GetSymbol< decltype(impact_ProcessAudio_Planar) >
>("impact_ProcessAudio_Planar" );
00376     GetBufferIn = m_lib.GetSymbol< decltype(impact_GetBufferIn) >
>("impact_GetBufferIn" );
00377     GetBufferOut = m_lib.GetSymbol< decltype(impact_GetBufferOut) >
>("impact_GetBufferOut" );
00378     GetEstimatedDelay = m_lib.GetSymbol< decltype(impact_GetEstimatedDelay) >
>("impact_GetEstimatedDelay" );
00379     GetFormatName = m_lib.GetSymbol< decltype(impact_GetFormatName) >
>("impact_GetFormatName" );
00380     GetFormatFromName = m_lib.GetSymbol< decltype(impact_GetFormatFromName) >
>("impact_GetFormatFromName" );
00381     GetBytesFromFormat = m_lib.GetSymbol< decltype(impact_GetBytesFromFormat) >
>("impact_GetBytesFromFormat" );
00382     GetMaxPacketFrame = m_lib.GetSymbol< decltype(impact_GetMaxPacketFrame) >
>("impact_GetMaxPacketFrame" );
00383     AddAudio = m_lib.GetSymbol< decltype(impact_AddAudio) >
>("impact_AddAudio" );
00384     AddPadAudio = m_lib.GetSymbol< decltype(impact_AddPadAudio) >
>("impact_AddPadAudio" );
00385     GetOutputCount = m_lib.GetSymbol< decltype(impact_GetOutputCount) >
>("impact_GetOutputCount" );
00386     GetAudio = m_lib.GetSymbol< decltype(impact_GetAudio) >
>("impact_GetAudio" );
00387     AudioConvertFrom = m_lib.GetSymbol< decltype(impact_AudioConvertFrom) >
>("impact_AudioConvertFrom" );
00388     AudioConvertTo = m_lib.GetSymbol< decltype(impact_AudioConvertTo) >
>("impact_AudioConvertTo" );
00389     StereoToMono = m_lib.GetSymbol< decltype(impact_StereoToMono) >
>("impact_StereoToMono" );
00390     MonoToStereo = m_lib.GetSymbol< decltype(impact_MonoToStereo) >
>("impact_MonoToStereo" );
00391     AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoFromLiveStereo) >
>("impact_AudioMonoFromLiveStereo");
00392     AudioMonoToLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoToLiveStereo) >
>("impact_AudioMonoToLiveStereo" );
00393     NewClient = m_lib.GetSymbol< decltype(impact_NewClient) >

```

```

>("impact_NewClient"                );
00394     DeleteClient                = m_lib.GetSymbol< decltype(impact_DeleteClient      )
>("impact_DeleteClient"            );
00395     ProcessJson                 = m_lib.GetSymbol< decltype(impact_ProcessJson    )
>("impact_ProcessJson"             );
00396     FreeJsonAnswer               = m_lib.GetSymbol< decltype(impact_FreeJsonAnswer  )
>("impact_FreeJsonAnswer"          );
00397     SaveState                    = m_lib.GetSymbol< decltype(impact_SaveState      )
>("impact_SaveState"               );
00398 #if IMPACT_HAS_WEBSERVER
00399     Webserver_tcp                 = m_lib.GetSymbol< decltype(impact_Webserver_tcp    )
>("impact_Webserver_tcp"           );
00400     Webserver_tcp2                = m_lib.GetSymbol< decltype(impact_Webserver_tcp2   )
>("impact_Webserver_tcp2"          );
00401     Webserver                     = m_lib.GetSymbol< decltype(impact_Webserver      )
>("impact_Webserver"               );
00402     Webserver_Stop                = m_lib.GetSymbol< decltype(impact_Webserver_Stop   )
>("impact_Webserver_Stop"          );
00403     Webserver_Status              = m_lib.GetSymbol< decltype(impact_Webserver_Status )
>("impact_Webserver_Status"        );
00404 #endif // IMPACT_HAS_WEBSERVER
00405
00406 // added 2023-03-22
00407 StereoToMono_Planar              = m_lib.GetSymbol< decltype(impact_StereoToMono_Planar )
>("impact_StereoToMono_Planar"     );
00408 MonoToStereo_Planar              = m_lib.GetSymbol< decltype(impact_MonoToStereo_Planar )
>("impact_MonoToStereo_Planar"     );
00409
00410     } catch (std::runtime_error& ) {
00411         return false;
00412     }
00413     // C functions allowed to be missing
00414     try {
00415         // added 2023-05-22
00416         SetMetadata                = m_lib.GetSymbol< decltype(impact_SetMetadata      )
>("impact_SetMetadata"            );
00417         GetMetadataInfos           = m_lib.GetSymbol< decltype(impact_GetMetadataInfos   )
>("impact_GetMetadataInfos"       );
00418         FreeMetadataInfos          = m_lib.GetSymbol< decltype(impact_FreeMetadataInfos  )
>("impact_FreeMetadataInfos"      );
00419         // added 2023-06-20
00420         SetPresetManager           = m_lib.GetSymbol< decltype(impact_SetPresetManager  )
>("impact_SetPresetManager"        );
00421         PresetManager_InformChange = m_lib.GetSymbol<
decltype(impact_PresetManager_InformChange ) >("impact_PresetManager_InformChange" );
00422
00423         // added 2023-09-21
00424         GetPossibleChunkSizeInFrames = m_lib.GetSymbol<
decltype(impact_GetPossibleChunkSizeInFrames ) >("impact_GetPossibleChunkSizeInFrames" );
00425         GetProcessChunkFrames      = m_lib.GetSymbol< decltype(impact_GetProcessChunkFrames )
>("impact_GetProcessChunkFrames"   );
00426         InitProcess3                = m_lib.GetSymbol< decltype(impact_InitProcess3     )
>("impact_InitProcess3"            );
00427
00428 #if IMPACT_HAS_CLOUDBUS
00429         // added 2023-06-26
00430         NewBus                      = m_lib.GetSymbol< decltype(impact_NewBus           )
>("impact_NewBus"                  );
00431         FreeBus                     = m_lib.GetSymbol< decltype(impact_FreeBus          )
>("impact_FreeBus"                 );
00432         SetInstanceBus              = m_lib.GetSymbol< decltype(impact_SetInstanceBus   )
>("impact_SetInstanceBus"          );
00433 #endif // IMPACT_HAS_CLOUDBUS
00434
00435         // added 2024-02-19
00436         SetMetadataMulti            = m_lib.GetSymbol< decltype(impact_SetMetadataMulti  )
>("impact_SetMetadataMulti"        );
00437
00438 #if IMPACT_HAS_WEBSERVER
00439         // added 2024-05-23
00440         Webserver_SetAppHealth      = m_lib.GetSymbol< decltype(impact_Webserver_SetAppHealth )
>("Webserver_SetAppHealth"         );
00441         Webserver_GetAppHealth      = m_lib.GetSymbol< decltype(impact_Webserver_GetAppHealth )
>("Webserver_GetAppHealth"         );
00442         Webserver_FreeString        = m_lib.GetSymbol< decltype(impact_Webserver_FreeString )
>("Webserver_FreeString"           );
00443 #endif // IMPACT_HAS_WEBSERVER
00444     } catch (std::runtime_error& ) {
00445         // Ignored, handler will take care
00446     }
00447
00448     m_bOK=true;
00449     return true;
00450 }
00451 bool IsOk() const {
00452     return m_bOK;

```

```

00460     }
00461
00462     };
00463
00464
00465
00466     static constexpr const char *process_name = "SOUND4 IMPACT.CL";
00467     static constexpr const char *process_shortcode = "impact";
00468     using SampleFormat = impact_SampleFormat;
00472     const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
00476     const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
00480     const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
00484     const size_t InputSampleSize=ChunkFrames*InputChannels;
00488     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00492     const size_t SampleRate=IMPACT_SAMPLE_RATE;
00493
00497     namespace helper {
00498         #ifdef _WIN32
00504         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00505             if (wstr.empty()) return {};
00506             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
NULL);
00507             std::string str(size, 0);
00508             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00509             return str;
00510         }
00516         static inline std::wstring UTF8ToWString(const std::string& str) {
00517             if (str.empty()) return std::wstring();
00518             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00519             std::wstring wstr(size, 0);
00520             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00521             return wstr;
00522         }
00523         #endif // _WIN32
00524
00530         template<typename T>
00531         struct SampleFormat {
00532         };
00533         template<>
00534         struct SampleFormat<int16_t> {
00535             const impact_SampleFormat format = S16_NATIVE;
00536         };
00537         template<>
00538         struct SampleFormat<int32_t> {
00539             const impact_SampleFormat format = S32_NATIVE;
00540         };
00541         template<>
00542         struct SampleFormat<float> {
00543             const impact_SampleFormat format = F32_NATIVE;
00544         };
00545
00546
00551         static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output,
size_t nSpl, impact_SampleFormat fmt) {
00552             dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
00553         }
00554         static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload,
size_t nSpl, impact_SampleFormat fmt) {
00555             dynlib.AudioConvertTo(input, payload, nSpl, fmt);
00556         }
00557         static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t
nSpl) {
00558             dynlib.StereoToMono(input, output, nSpl);
00559         }
00560         static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t
nSpl) {
00561             dynlib.MonoToStereo(input, output, nSpl);
00562         }
00563         static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float
*inputR, float *output, size_t nSpl) {
00564             dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
00565         }
00566         static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL,
float *outputR, size_t nSpl) {
00567             dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
00568         }
00569         static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float
*output) {
00570             dynlib.AudioMonoFromLiveStereo(payload, output);
00571         }
00572         static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t
*payload) {
00573             dynlib.AudioMonoToLiveStereo(input, payload);
00574         }
00575
00576     }; // namespace helper
00577

```

```

00583     static inline std::string GetVersion(CDynLib& dynlib)
00584     {
00585         return std::string(dynlib.GetVersion());
00586     }
00587
00598     static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
00599     {
00600         return dynlib.GetChunkSizeInFrames();
00601     }
00602
00611     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
00612     {
00613         std::vector<unsigned int> list;
00614         if (dynlib.GetPossibleChunkSizeInFrames() {
00615             for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
00616                 list.push_back(*src);
00617             }
00618         } else {
00619             list.push_back(64);
00620         }
00621         return list;
00622     }
00623
00631     static inline unsigned int GetChannelCount(CDynLib& dynlib)
00632     {
00633         return dynlib.GetChannelCount();
00634     }
00635
00644     static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
00645     {
00646         return dynlib.GetAudioInputCount();
00647     }
00648
00657     static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
00658     {
00659         return dynlib.GetAudioOutputCount();
00660     }
00661
00669     static inline unsigned int GetSampleRate(CDynLib& dynlib)
00670     {
00671         return dynlib.GetSampleRate();
00672     }
00673
00679     static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
00680     {
00681         // Do some sanity checks
00682         if (dynlib.GetSampleRate() != IMPACT_SAMPLE_RATE) {
00683             throw std::runtime_error("Bad library sampling rate");
00684         }
00685         if (a_bCheckFrames && dynlib.GetChunkSizeInFrames() != ChunkFrames) {
00686             throw std::runtime_error("Bad library frame size");
00687         }
00688         if (dynlib.GetChannelCount() != IMPACT_CHANNEL_COUNT) {
00689             throw std::runtime_error("Bad library channel count");
00690         }
00691         if (dynlib.GetAudioInputCount() != IMPACT_AUDIO_INPUT_COUNT) {
00692             throw std::runtime_error("Bad library input count");
00693         }
00694         if (dynlib.GetAudioOutputCount() != IMPACT_AUDIO_OUTPUT_COUNT) {
00695             throw std::runtime_error("Bad library output count");
00696         }
00697     }
00698
00705     static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
00706     {
00707         return std::string(dynlib.GetFormatName(fmt));
00708     }
00709
00716     static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
00717     {
00718         return dynlib.GetFormatFromName(name.c_str());
00719     }
00720
00727     static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
00728     {
00729         return dynlib.GetBytesFromFormat(fmt);
00730     }
00731
00735     using LogSeverity = impact_LogSeverity;
00739     using log_cb_t = std::function<void(LogSeverity, const std::string&);>;
00740
00744     static log_cb_t _log_cb;
00745     extern "C" {
00746         static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
00747             _log_cb(severity, std::string(c_msg));
00748         }
00749     }

```

```

00753     static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
00754     {
00755         dynlib.SetLogSeverity(severity);
00756     }
00760     static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
00761     {
00762         _log_cb=cb;
00763         dynlib.SetLoggerCallback(&_log_cb_c);
00764     }
00765
00766 #if IMPACT_HAS_CLOUDBUS
00767     class CBus {
00768     public:
00769         CBus(CDynLib& dynlib)
00770             : m_dynlib(dynlib)
00771         {
00772             if (m_dynlib.NewBus()) {
00773                 m_bus=m_dynlib.NewBus();
00774             }
00775         }
00776         ~CBus()
00777         {
00778             if (m_bus && m_dynlib.FreeBus()) {
00779                 m_dynlib.FreeBus(m_bus);
00780             }
00781             m_bus = nullptr;
00782         }
00783         impact_CBus *Get() const { return m_bus; }
00784     private:
00785         CDynLib& m_dynlib;
00786         impact_CBus *m_bus;
00787     };
00788 #endif // IMPACT_HAS_CLOUDBUS
00789
00794     class CPresetLoader {
00795     public:
00796         CPresetLoader() = default;
00797         virtual ~CPresetLoader() = default;
00798
00799         virtual bool IsReadOnly() = 0;
00800         virtual bool Exists(const std::filesystem::path &name) = 0;
00801         virtual bool Remove(const std::filesystem::path &name) = 0;
00802         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00803         virtual std::vector<std::filesystem::path> GetAll() = 0;
00804         virtual std::string Read(const std::filesystem::path &filename) = 0;
00805         virtual bool Write(const std::filesystem::path &filename, const std::string &content) = 0;
00806     };
00807     using PresetChange_Kind = impact_PresetChange_Kind;
00808
00812     class CInstance {
00813     public:
00817         CInstance(CDynLib& dynlib)
00818             : m_dynlib(dynlib)
00819         {
00820             params = m_dynlib.NewParameters();
00821             if (!params) throw std::bad_alloc();
00822         }
00823         // non-copyable
00824         CInstance(const CInstance& ) = delete;
00825         CInstance& operator=( const CInstance& ) = delete;
00829         virtual ~CInstance()
00830         {
00831             Stop();
00832             if (params) m_dynlib.FreeParameters(params);
00833         }
00837         bool IsOk() const { return instance!=nullptr; }
00845         void SetParam(const std::string& name, const std::string& value)
00846         {
00847             assert(params);
00848             m_dynlib.SetParameter(params, name.c_str(), value.c_str());
00849         }
00855         std::string GetParam(const std::string& name)
00856         {
00857             assert(params);
00858             auto c_value=m_dynlib.GetParameter(params, name.c_str());
00859             std::string ret(c_value);
00860             m_dynlib.FreeParameterValue(c_value);
00861             return ret;
00862         }
00863 #if IMPACT_HAS_CLOUDBUS
00867         void SetBus(const CBus& bus) {
00868             assert(params);
00869             if (m_dynlib.SetInstanceBus()) {
00870                 m_dynlib.SetInstanceBus(params,bus.Get());
00871             }
00872         }
00873 #endif // IMPACT_HAS_CLOUDBUS

```

```

00874
00884 void SetPresetManager(CPresetLoader *preset_manager)
00885 {
00886     if (!preset_manager) return;
00887     if (!m_dynlib.SetPresetManager) return;
00888     assert(params);
00889     m_dynlib.SetPresetManager(params,
00890         impact_custom_reader,
00891         impact_custom_reader_free,
00892         impact_custom_writer,
00893         impact_custom_exists,
00894         impact_custom_getall,
00895         impact_custom_getall_free,
00896         impact_custom_remove,
00897         impact_custom_rename,
00898         preset_manager->IsReadOnly(),
00899         preset_manager
00900     );
00901 }
00902
00917 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)
00918 {
00919     assert(!instance);
00920     std::string l_save_path;
00921     if (!save_path.empty()) {
00922         l_save_path = save_path.u8string();
00923     }
00924     if (m_dynlib.InitProcess3) {
00925         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);
00926     } else if (frames_per_chunk!=64) {
00927         return false;
00928     } else {
00929         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);
00930     }
00931     if (!instance) return false;
00932     if (!init_metadata.empty()) {
00933         for (auto&& [key,value]: init_metadata) {
00934             m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
00935         }
00936         init_metadata.clear();
00937     }
00938     update_thread = std::thread([this, json_port]() {
00939         OnUpdateThreadStart();
00940         m_dynlib.StartUpdateThread(instance, json_port);
00941         OnUpdateThreadStop();
00942     });
00943     int timeout=5000;
00944     #ifdef DEBUG
00945         timeout=60*1000; // For debugger pauses
00946     #endif
00947     if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
00948         return false;
00949     }
00950     return true;
00951 }
00952
00960 unsigned int GetChunkFrames() {
00961     if (instance && m_dynlib.GetProcessChunkFrames) {
00962         return m_dynlib.GetProcessChunkFrames(instance);
00963     }
00964     return 64;
00965 }
00966
00974 void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00975 {
00976     if (m_dynlib.PresetManager_InformChange) {
00977         m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(),
(impact_PresetChange_Kind) change_kind);
00978     }
00979 }
00980
00992 void SetMetadata(const std::string& key, const char* value)
00993 {
00994     if (m_dynlib.SetMetadata) {
00995         if (instance) {
00996             m_dynlib.SetMetadata(instance, key.c_str(), value);
00997         } else {
00998             init_metadata.push_back( {key, value} );
00999         }
01000     }
01001 }
01002

```

```

01003
01013 void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
01014 {
01015     if (instance) {
01016         if (m_dynlib.SetMetadataMulti) {
01017             typedef const char* pchar;
01018             pchar* keyvalue=new pchar[2*list.size()+1];
01019             size_t n=0;
01020             for (auto&& [key,value]: list) {
01021                 keyvalue[2*n+0]=key.c_str();
01022                 keyvalue[2*n+1]=value;
01023                 n++;
01024             }
01025             keyvalue[2*n+0]=nullptr;
01026
01027             impact_SetMetadataMulti(instance, keyvalue);
01028
01029             delete[] keyvalue;
01030         } else {
01031             for (auto&& [key,value]: list) {
01032                 SetMetadata(key, value);
01033             }
01034         }
01035     } else {
01036         for (auto&& [key,value]: list) {
01037             if (value) {
01038                 init_metadata.push_back( {key, value} );
01039             }
01040         }
01041     }
01042 }
01043
01049 std::vector< std::tuple<std::string, std::string> > GetMetadataInfos()
01050 {
01051     std::vector< std::tuple<std::string, std::string> > values;
01052     if (m_dynlib.GetMetadataInfos) {
01053         const char** c_values = m_dynlib.GetMetadataInfos(instance);
01054         if (c_values) {
01055             for (const char** c_value=c_values; *c_value; ) {
01056                 std::string key(*c_value);
01057                 c_value++;
01058                 if (*c_value) {
01059                     std::string descr(*c_value);
01060                     c_value++;
01061                     values.push_back( {key, descr} );
01062                 }
01063             }
01064             m_dynlib.FreeMetadataInfos(instance, c_values);
01065         }
01066     }
01067     return values;
01068 }
01069
01070 // Can be overloaded, called at start of UpdateThread in the thread
01071 virtual void OnUpdateThreadStart() {}
01072 // Can be overloaded to call at end of UpdateThread in the thread
01073 virtual void OnUpdateThreadStop() {}
01074
01075 #if IMPACT_HAS_WEBSERVER
01081 bool StartWebServer(int http_port, int https_port=0)
01082 {
01083     assert(instance);
01084     assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01085     webserver = m_dynlib.Webserver(http_port, https_port, instance);
01086     if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01087         return false;
01088     }
01089     int web_status=m_dynlib.Webserver_Status(webserver);
01090     if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01091         return false;
01092     }
01093     if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01094         return false;
01095     }
01096     return true;
01097 }
01103 void StopWebServer()
01104 {
01105     if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01106         m_dynlib.Webserver_Stop(webserver, 1000);
01107         webserver = SOUND4_INVALID_WEBSERVER_ID;
01108     }
01109 }
01110
01114 void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
01115 content)
01116 {

```

```

01116         assert(instance);
01117         if (!m_dynlib.Webserver_SetAppHealth) return;
01118         m_dynlib.Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
content.c_str());
01119     }
01120     void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01121     {
01122         assert(instance);
01123         if (!m_dynlib.Webserver_GetAppHealth) return;
01124         char* c_contenttype=nullptr;
01125         char* c_content=nullptr;
01126         m_dynlib.Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01127         contenttype=c_contenttype;
01128         content=c_content;
01129         m_dynlib.Webserver_FreeString(c_contenttype);
01130         m_dynlib.Webserver_FreeString(c_content);
01131     }
01132 #endif // IMPACT_HAS_WEBSERVER
01133
01134     int Terminate()
01135     {
01136         if (!instance) return 0;
01137         return m_dynlib.TerminateProcess(instance);
01138     }
01139
01140     void Stop()
01141     {
01142         if (!instance) return;
01143         #if IMPACT_HAS_WEBSERVER
01144             StopWebServer();
01145         #endif // IMPACT_HAS_WEBSERVER
01146         if (update_thread.joinable()) {
01147             m_dynlib.StopUpdateThread(instance); // this will exit the update thread
01148             update_thread.join();
01149         }
01150         m_dynlib.ExitProcess(instance);
01151         instance=nullptr;
01152     }
01153
01154     unsigned int GetEstimatedDelay()
01155     {
01156         assert(instance);
01157         return m_dynlib.GetEstimatedDelay(instance);
01158     }
01159
01160     std::array<float,InputSampleSize>& GetBufferIn()
01161     {
01162         assert(instance);
01163         float *buf=m_dynlib.GetBufferIn(instance);
01164         return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
01165     }
01166
01167     std::array<float,OutputSampleSize>& GetBufferOut()
01168     {
01169         assert(instance);
01170         float *buf= m_dynlib.GetBufferOut(instance);
01171         return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
01172     }
01173
01174     void ProcessAudio()
01175     {
01176         assert(instance);
01177         m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance),
m_dynlib.GetBufferOut(instance));
01178     }
01179
01180     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01181     void ProcessAudio_Planar(float const * const *input, float * const *output)
01182     {
01183         assert(instance);
01184         m_dynlib.ProcessAudio_Planar(instance, input, output);
01185     }
01186
01187     template<typename T>
01188     std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01189     {
01190         assert(instance);
01191         std::vector<T> output;
01192         unsigned int out_offset=0;
01193         unsigned int in_offset=0;
01194         unsigned int todo = input.size();
01195         while (todo>0) {
01196             unsigned int left = AddAudio(&input[in_offset], todo);
01197             unsigned int out_avail = m_dynlib.GetOutputCount(instance);
01198             output.resize(out_offset + out_avail);
01199             GetAudio(&output[out_offset], out_avail);
01200             out_offset+=out_avail;
01201             in_offset += todo-left;
01202         }
01203     }

```

```

01268         todo=left;
01269     }
01270     return output;
01271 }
01272
01273 class CClient {
01274 public:
01275     CClient(impact_CInstance* instance, CDynLib& dynlib)
01276     : m_dynlib(dynlib)
01277     {
01278         assert(instance);
01279         client=m_dynlib.NewClient(instance);
01280         if (!client) throw std::bad_alloc();
01281     }
01282     // non-copyable
01283     CClient( const CInstance& ) = delete;
01284     CClient& operator=( const CInstance& ) = delete;
01285     ~CClient()
01286     {
01287         if (client) {
01288             m_dynlib.DeleteClient(client);
01289             client=nullptr;
01290         }
01291     }
01292     std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01293     {
01294         assert(client);
01295         int need_save=0;
01296         const char *canswer = m_dynlib.ProcessJson(client, request.c_str(),
01297 &need_save);
01298         if (!canswer) return {};
01299         std::string answer(canswer);
01300         m_dynlib.FreeJsonAnswer(canswer);
01301         if (NeedSave) {
01302             *NeedSave=(need_save!=0);
01303         }
01304         return answer;
01305     }
01306 private:
01307     CDynLib& m_dynlib;
01308     impact_CClientInstance *client = nullptr;
01309 };
01310 std::shared_ptr<CClient> NewClient()
01311 {
01312     assert(instance);
01313     return std::make_shared<CClient>(instance, m_dynlib);
01314 }
01315
01316 protected:
01317     template<typename T>
01318     unsigned int AddAudio(const T* payload, unsigned int nFrame)
01319     {
01320         assert(instance);
01321         return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
01322 helper::SampleFormat<T>::format);
01323     }
01324     template<typename T>
01325     unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01326     {
01327         assert(instance);
01328         return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
01329 helper::SampleFormat<T>::format);
01330     }
01331 private:
01332     CDynLib& m_dynlib;
01333     impact_CParameters* params = nullptr;
01334     impact_CInstance* instance = nullptr;
01335     std::thread update_thread;
01336     std::vector< std::pair<std::string, std::string> > init_metadata;
01337 #if IMPACT_HAS_WEBSERVER
01338     uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01339 #endif // IMPACT_HAS_WEBSERVER
01340 };
01341
01342 }; // namespace dyn
01343 }; // namespace impact
01344 }; // namespace sound4
01345
01346 // bridge C callbacks to CPresetLoader
01347 extern "C" {
01348     static char *impact_custom_reader(const fs_char *filename, void* userdata) {
01349         sound4::impact::dyn::CPresetLoader
01350 *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01351         std::string content=preset_loader->Read(filename);
01352         return strdup(content.c_str());
01353     }

```

```

01370     }
01371     static void impact_custom_reader_free(char *content, void* userdata) {
01372         if (!content) return;
01373         free(content);
01374     }
01375     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01376         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01377         auto res=preset_loader->Write(filename,content);
01378         return res?0:-1;
01379     }
01380     static int impact_custom_exists(const fs_char *filename, void* userdata) {
01381         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01382         auto res=preset_loader->Exists(filename);
01383         return res?0:-1;
01384     }
01385     static fs_char** impact_custom_getall(void* userdata) {
01386         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01387         auto res=preset_loader->GetAll();
01388         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01389         for (size_t n=0;n<res.size();n++) {
01390             all[n]=fs_strdup(res[n].c_str());
01391         }
01392         all[res.size()]=nullptr;
01393         return all;
01394     }
01395     static void impact_custom_getall_free(fs_char** all, void* userdata) {
01396         if (!all) return;
01397         for (fs_char** one=all;*one;one++) {
01398             free(*one);
01399         }
01400         free(all);
01401     }
01402     static int impact_custom_remove(const fs_char *filename, void* userdata) {
01403         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01404         auto res=preset_loader->Remove(filename);
01405         return res?0:-1;
01406     }
01407     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01408         sound4::impact::dyn::CPresetLoader
*preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
01409         auto res=preset_loader->Rename(from,to);
01410         return res?0:-1;
01411     }
01412
01413 };

```

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 <loaderapi.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     namespace helper {
00054         #ifdef _WIN32
00055         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00056             if (wstr.empty()) return {};
00057             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
NULL);
00058             std::string str(size, 0);
00059             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00060             return str;
00061         }
00062         #endif
00063     }
00064 }
```

```

00070     }
00071     static inline std::wstring UTF8ToWString(const std::string& str) {
00072         if (str.empty()) return std::wstring();
00073         int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00074         std::wstring wstr(size, 0);
00075         MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00076         return wstr;
00077     }
00078 #endif // _WIN32
00079
00080     template<typename T>
00081     struct SampleFormat {
00082     };
00083     template<>
00084     struct SampleFormat<int16_t> {
00085         const sound4cl_SampleFormat format = S16_NATIVE;
00086     };
00087     template<>
00088     struct SampleFormat<int32_t> {
00089         const sound4cl_SampleFormat format = S32_NATIVE;
00090     };
00091     template<>
00092     struct SampleFormat<float> {
00093         const sound4cl_SampleFormat format = F32_NATIVE;
00094     };
00095
00096 // Storage definition, depending on OS
00097 #if UNIXLIKE
00098     using DynLib_t = void *;
00099     using DynFunc_t = void *;
00100 #else
00101     using DynLib_t = HMODULE;
00102     using DynFunc_t = FARPROC;
00103 #endif
00104
00105     template <typename T>
00106     class DynFuncHolder {
00107     private:
00108         DynFunc_t m_ptr = NULL;
00109     public:
00110         DynFuncHolder() = default;
00111         DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00112         operator bool() const { return (m_ptr!=NULL); }
00113         bool IsOk() const { return m_ptr!=NULL; }
00114         operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00115     };
00116
00117     class CDynLoader {
00118     private:
00119         DynLib_t m_lib {};
00120     public:
00121         CDynLoader() = default;
00122         CDynLoader(CDynLoader&&) = default;
00123         // No copy allowed
00124         CDynLoader(const CDynLoader&) = delete;
00125         CDynLoader& operator=(CDynLoader const&) = delete;
00126         operator bool() const {
00127             return (m_lib!=NULL);
00128         }
00129         bool IsOk() const {
00130             return (m_lib!=NULL);
00131         }
00132         #if UNIXLIKE
00133         void Close() {
00134             if (IsOk()) {
00135                 #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00136                     dlclose(m_lib);
00137                 #endif // __SANITIZE_ADDRESS__
00138                 m_lib=NULL;
00139             }
00140         }
00141         bool Load(const std::filesystem::path& dynlib)
00142         {
00143             Close();
00144             m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00145             return IsOk();
00146         }
00147         template <typename T>
00148         DynFuncHolder<T> GetSymbol(const std::string& name) {
00149             auto ptr=dlsym(m_lib, name.c_str());
00150             if (!ptr) {
00151                 throw std::runtime_error("Missing function in library");
00152             }
00153             return DynFuncHolder<T>(ptr);
00154         }
00155         template <typename T>
00156         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_FREELIBRARY before including this file if you
really want to unload the library dynamically
00198             #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
FreeLibraryAndExitThread(m_lib, 0);
00199             #elif defined(SOUND4_CALL_FREELIBRARY)
FreeLibrary(m_lib);
00200             #endif // SOUND4_CALL_FREELIBRARY
m_lib=NULL;
00201         }
00202     }
00203     template <typename T>
00204     DynFuncHolder<T> GetSymbol(const std::string& name) {
00205         auto ptr=GetProcAddress(m_lib, name.c_str());
00206         if (!ptr) {
00207             throw std::runtime_error("Missing function in library");
00208         }
00209         return DynFuncHolder<T>(ptr);
00210     }
00211     template <typename T>
00212     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00213         auto ptr=GetProcAddress(m_lib, name.c_str());
00214         return DynFuncHolder<T>(ptr);
00215     }
00216     static std::filesystem::path GetThisLibraryPath(void) {
00217         static wchar_t path[MAX_PATH]={};
00218         HMODULE hm = NULL;
00219         if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
(LPCWSTR) &GetThisLibraryPath, &hm))
00220         {
00221             if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00222                 return std::filesystem::path(path);
00223             }
00224         }
00225         return {};
00226     }
00227     #endif
00228     virtual ~CDynLoader() {
00229         Close();
00230     }
00231     }; // class CDynLoader
00232 }; // namespace helper
00233
00234 using LogSeverity = sound4cl_LogSeverity;
00235 const LogSeverity none = (LogSeverity) sound4cl_none;
00236 const LogSeverity fatal = (LogSeverity) sound4cl_fatal;
00237 const LogSeverity error = (LogSeverity) sound4cl_error;
00238 const LogSeverity warning = (LogSeverity) sound4cl_warning;
00239 const LogSeverity info = (LogSeverity) sound4cl_info;
00240 const LogSeverity verbose = (LogSeverity) sound4cl_verbose;
00241 const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2;
00242 const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3;
00243 const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4;
00244 const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5;
00245
00246 using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00247
00248 static log_cb_t _log_cb;
00249 extern "C" {
00250     static inline void _log_cb_c(LogSeverity severity, const char *c_msg) {
00251         _log_cb(severity, std::string(c_msg));
00252     }
00253 }
00254

```

```

00269
00276     class CProcessor {
00277     private:
00278         helper::CDynLoader m_lib;
00279         bool m_bOK = false;
00280         std::string m_prefix; // The prefix of functions, depends on the library (cloud_, impact_...)
00281     protected:
00282         friend class CBus;
00283         friend class CInstance;
00284
00285         // All those wrapped functions have the same signature as there C sound4cl_XXX
00286         helper::DynFuncHolder< decltype(sound4cl_GetVersion) ) > fnGetVersion ;
00287         helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) ) > fnGetChunkSizeInFrames ;
00288         helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) ) > fnGetChannelCount ;
00289         helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) ) > fnGetAudioInputCount ;
00290         helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) ) > fnGetAudioOutputCount ;
00291         helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) ) > fnGetSampleRate ;
00292         helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) ) > fnSetLoggerCallback ;
00293         helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) ) > fnSetLogSeverity ;
00294         helper::DynFuncHolder< decltype(sound4cl_NewParameters) ) > fnNewParameters ;
00295         helper::DynFuncHolder< decltype(sound4cl_FreeParameters) ) > fnFreeParameters ;
00296         helper::DynFuncHolder< decltype(sound4cl_SetParameter) ) > fnSetParameter ;
00297         helper::DynFuncHolder< decltype(sound4cl_GetParameter) ) > fnGetParameter ;
00298         helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) ) > fnFreeParameterValue ;
00299         helper::DynFuncHolder< decltype(sound4cl_InitProcess) ) > fnInitProcess ;
00300         helper::DynFuncHolder< decltype(sound4cl_InitProcess2) ) > fnInitProcess2 ;
00301         helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) ) > fnTerminateProcess ;
00302         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 ;
00349         helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) ) >
00350         fnPresetManager_InformChange ;
00351
00352         // added 2023-09-21
00352         helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) ) >
00353         fnGetPossibleChunkSizeInFrames ;
00353         helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) ) >
00354         fnGetProcessChunkFrames ;
00354         helper::DynFuncHolder< decltype(sound4cl_InitProcess3) ) > fnInitProcess3 ;
00355
00356         // added 2023-06-26, only for processes supporting bus

```

```

00357     helper::DynFuncHolder< decltype(sound4cl_NewBus)                ) > fnNewBus        ;
00358     helper::DynFuncHolder< decltype(sound4cl_FreeBus)               ) > fnFreeBus      ;
00359     helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus)        ) > fnSetInstanceBus ;
00360
00361     // added 2024-02-19
00362     helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti)      ) > fnSetMetadataMulti
;
00363
00364     // added 2024-05-23
00365     helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) ) >
fnWebserver_SetAppHealth ;
00366     helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) ) >
fnWebserver_GetAppHealth ;
00367     helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString)  ) > fnWebserver_FreeString
;
00368
00369     template <typename T>
00370     helper::DynFuncHolder<T> GetPrefixSymbol(const std::string& name) {
00371         std::string prefix_name = m_prefix+name;
00372         return m_lib.GetSymbol<T>(prefix_name);
00373     }
00374
00375     public:
00376     CProcessor() = default;
00377
00385     bool Load(const std::filesystem::path& filepath) {
00386         m_lib.Close();
00387         // 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                 m_lib.Load(std::filesystem::current_path() / filename);
00407             }
00408             if (!m_lib.IsOk()) {
00409                 return false;
00410             }
00411         }
00412         // Get information from the library itself
00413         try {
00414             auto GetInfo = m_lib.GetSymbol< decltype(sound4cl_SOUND4_GetProcessInfo) )
>("SOUND4_GetProcessInfo" );
00415             auto info = GetInfo();
00416             m_prefix = info->prefix;
00417         } catch (std::runtime_error& ) {
00418             // Compatibility: search in known libraries without this function
00419             #ifdef _WIN32
00420                 std::string filename = helper::WStringToUTF8(filepath.stem());
00421             #else // !_WIN32
00422                 std::string filename = filepath.stem();
00423                 // remove lib prefix
00424                 filename=filename.substr(3);
00425             #endif // !_WIN32
00426             if (filename=="sound4.xl.cloud") {
00427                 m_prefix="cloudxl_";
00428             } else if (filename=="sound4.impact.cl") {
00429                 m_prefix="impact_";
00430             } else if (filename=="sound4.bigvoice.cl") {
00431                 m_prefix="bigvoice_";
00432             } else {
00433                 return false;
00434             }
00435         }
00436
00437         // Load all C functions
00438         try {
00439             fnGetVersion                = GetPrefixSymbol< decltype(sound4cl_GetVersion)
) >("GetVersion" );
00440             fnGetChunkSizeInFrames      = GetPrefixSymbol< decltype(sound4cl_GetChunkSizeInFrames)
) >("GetChunkSizeInFrames" );
00441             fnGetChannelCount           = GetPrefixSymbol< decltype(sound4cl_GetChannelCount)
) >("GetChannelCount" );
00442             fnGetAudioInputCount        = GetPrefixSymbol< decltype(sound4cl_GetAudioInputCount

```

```

    ) > ("GetAudioInputCount"      );
00443     fnGetAudioOutputCount      = GetPrefixSymbol< decltype(sound4cl_GetAudioOutputCount
    ) > ("GetAudioOutputCount"    );
00444     fnGetSampleRate             = GetPrefixSymbol< decltype(sound4cl_GetSampleRate
    ) > ("GetSampleRate"           );
00445     fnSetLoggerCallback         = GetPrefixSymbol< decltype(sound4cl_SetLoggerCallback
    ) > ("SetLoggerCallback"       );
00446     fnSetLogSeverity            = GetPrefixSymbol< decltype(sound4cl_SetLogSeverity
    ) > ("SetLogSeverity"          );
00447     fnNewParameters             = GetPrefixSymbol< decltype(sound4cl_NewParameters
    ) > ("NewParameters"          );
00448     fnFreeParameters            = GetPrefixSymbol< decltype(sound4cl_FreeParameters
    ) > ("FreeParameters"         );
00449     fnSetParameter              = GetPrefixSymbol< decltype(sound4cl_SetParameter
    ) > ("SetParameter"           );
00450     fnGetParameter              = GetPrefixSymbol< decltype(sound4cl_GetParameter
    ) > ("GetParameter"           );
00451     fnFreeParameterValue        = GetPrefixSymbol< decltype(sound4cl_FreeParameterValue
    ) > ("FreeParameterValue"     );
00452     fnInitProcess               = GetPrefixSymbol< decltype(sound4cl_InitProcess
    ) > ("InitProcess"            );
00453     fnInitProcess2              = GetPrefixSymbol< decltype(sound4cl_InitProcess2
    ) > ("InitProcess2"           );
00454     fnTerminateProcess           = GetPrefixSymbol< decltype(sound4cl_TerminateProcess
    ) > ("TerminateProcess"        );
00455     fnExitProcess               = GetPrefixSymbol< decltype(sound4cl_ExitProcess
    ) > ("ExitProcess"            );
00456     fnStartUpdateThread         = GetPrefixSymbol< decltype(sound4cl_StartUpdateThread
    ) > ("StartUpdateThread"       );
00457     fnStopUpdateThread          = GetPrefixSymbol< decltype(sound4cl_StopUpdateThread
    ) > ("StopUpdateThread"        );
00458     fnWaitUpdateThreadReady     = GetPrefixSymbol< decltype(sound4cl_WaitUpdateThreadReady
    ) > ("WaitUpdateThreadReady"   );
00459     fnProcessAudio              = GetPrefixSymbol< decltype(sound4cl_ProcessAudio
    ) > ("ProcessAudio"           );
00460     fnProcessAudio_Planar       = GetPrefixSymbol< decltype(sound4cl_ProcessAudio_Planar
    ) > ("ProcessAudio_Planar"     );
00461     fnGetBufferIn               = GetPrefixSymbol< decltype(sound4cl_GetBufferIn
    ) > ("GetBufferIn"            );
00462     fnGetBufferOut              = GetPrefixSymbol< decltype(sound4cl_GetBufferOut
    ) > ("GetBufferOut"           );
00463     fnGetEstimatedDelay         = GetPrefixSymbol< decltype(sound4cl_GetEstimatedDelay
    ) > ("GetEstimatedDelay"       );
00464     fnGetFormatName             = GetPrefixSymbol< decltype(sound4cl_GetFormatName
    ) > ("GetFormatName"          );
00465     fnGetFormatFromName         = GetPrefixSymbol< decltype(sound4cl_GetFormatFromName
    ) > ("GetFormatFromName"       );
00466     fnGetBytesFromFormat        = GetPrefixSymbol< decltype(sound4cl_GetBytesFromFormat
    ) > ("GetBytesFromFormat"      );
00467     fnGetMaxPacketFrame         = GetPrefixSymbol< decltype(sound4cl_GetMaxPacketFrame
    ) > ("GetMaxPacketFrame"       );
00468     fnAddAudio                  = GetPrefixSymbol< decltype(sound4cl_AddAudio
    ) > ("AddAudio"               );
00469     fnAddPadAudio               = GetPrefixSymbol< decltype(sound4cl_AddPadAudio
    ) > ("AddPadAudio"            );
00470     fnGetOutputCount            = GetPrefixSymbol< decltype(sound4cl_GetOutputCount
    ) > ("GetOutputCount"         );
00471     fnGetAudio                  = GetPrefixSymbol< decltype(sound4cl_GetAudio
    ) > ("GetAudio"               );
00472     fnAudioConvertFrom          = GetPrefixSymbol< decltype(sound4cl_AudioConvertFrom
    ) > ("AudioConvertFrom"        );
00473     fnAudioConvertTo            = GetPrefixSymbol< decltype(sound4cl_AudioConvertTo
    ) > ("AudioConvertTo"          );
00474     fnStereoToMono              = GetPrefixSymbol< decltype(sound4cl_StereoToMono
    ) > ("StereoToMono"           );
00475     fnMonoToStereo              = GetPrefixSymbol< decltype(sound4cl_MonoToStereo
    ) > ("MonoToStereo"           );
00476     fnAudioMonoFromLiveStereo   = GetPrefixSymbol<
decltype(sound4cl_AudioMonoFromLiveStereo) > ("AudioMonoFromLiveStereo");
00477     fnAudioMonoToLiveStereo     = GetPrefixSymbol< decltype(sound4cl_AudioMonoToLiveStereo
    ) > ("AudioMonoToLiveStereo" );
00478     fnNewClient                 = GetPrefixSymbol< decltype(sound4cl_NewClient
    ) > ("NewClient"              );
00479     fnDeleteClient              = GetPrefixSymbol< decltype(sound4cl_DeleteClient
    ) > ("DeleteClient"           );
00480     fnProcessJson               = GetPrefixSymbol< decltype(sound4cl_ProcessJson
    ) > ("ProcessJson"           );
00481     fnFreeJsonAnswer            = GetPrefixSymbol< decltype(sound4cl_FreeJsonAnswer
    ) > ("FreeJsonAnswer"         );
00482     fnSaveState                 = GetPrefixSymbol< decltype(sound4cl_SaveState
    ) > ("SaveState"              );
00483     } catch (std::runtime_error& ) {
00484         return false;
00485     }
00486     // Webserver : not for all projects
00487     try {
00488         fnWebserver_tcp         = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp

```

```

    ) > ("Webserver_tcp"                );
00489         fnWebserver_tcp2              = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp2
    ) > ("Webserver_tcp2"                );
00490         fnWebserver                    = GetPrefixSymbol< decltype(sound4cl_Webserver
    ) > ("Webserver"                      );
00491         fnWebserver_Stop               = GetPrefixSymbol< decltype(sound4cl_Webserver_Stop
    ) > ("Webserver_Stop"                );
00492         fnWebserver_Status             = GetPrefixSymbol< decltype(sound4cl_Webserver_Status
    ) > ("Webserver_Status"              );
00493         } catch (std::runtime_error& ) {
00494             // Ignored, handler will take care
00495         }
00496         // Stereo/Mono conversion : must be present
00497         try {
00498             // added 2023-03-22
00499             fnStereoToMono_Planar       = GetPrefixSymbol< decltype(sound4cl_StereoToMono_Planar
    ) > ("StereoToMono_Planar"           );
00500             fnMonoToStereo_Planar       = GetPrefixSymbol< decltype(sound4cl_MonoToStereo_Planar
    ) > ("MonoToStereo_Planar"           );
00501         } catch (std::runtime_error& ) {
00502             return false;
00503         }
00504         // C functions allowed to be missing
00505         try {
00506             // added 2023-05-22
00507             fnSetMetadata               = GetPrefixSymbol< decltype(sound4cl_SetMetadata
00508     ) > ("SetMetadata"                  );
00509             fnGetMetadataInfos          = GetPrefixSymbol< decltype(sound4cl_GetMetadataInfos
    ) > ("GetMetadataInfos"              );
00510             fnFreeMetadataInfos         = GetPrefixSymbol< decltype(sound4cl_FreeMetadataInfos
    ) > ("FreeMetadataInfos"            );
00511             // added 2023-06-20
00512             fnSetPresetManager          = GetPrefixSymbol< decltype(sound4cl_SetPresetManager
    ) > ("SetPresetManager"              );
00513             fnPresetManager_InformChange = GetPrefixSymbol<
decltype(sound4cl_PresetManager_InformChange    ) > ("PresetManager_InformChange"    );
00514             // added 2023-09-21
00515             fnGetPossibleChunkSizeInFrames = GetPrefixSymbol<
decltype(sound4cl_GetPossibleChunkSizeInFrames    ) > ("GetPossibleChunkSizeInFrames"    );
00517             fnGetProcessChunkFrames      = GetPrefixSymbol<
decltype(sound4cl_GetProcessChunkFrames            ) > ("GetProcessChunkFrames"            );
00518             fnInitProcess3               = GetPrefixSymbol< decltype(sound4cl_InitProcess3
    ) > ("InitProcess3"                 );
00519         } catch (std::runtime_error& ) {
00520             // Ignored, handler will take care
00521         }
00522         // bus: not for all projects
00523         try {
00524             // added 2023-06-26
00525             fnNewBus                    = GetPrefixSymbol< decltype(sound4cl_NewBus
00526     ) > ("NewBus"                      );
00527             fnFreeBus                   = GetPrefixSymbol< decltype(sound4cl_FreeBus
    ) > ("FreeBus"                      );
00528             fnSetInstanceBus            = GetPrefixSymbol< decltype(sound4cl_SetInstanceBus
    ) > ("SetInstanceBus"                );
00529         } catch (std::runtime_error& ) {
00530             // Ignored, handler will take care
00531         }
00532         // Other optional
00533         try {
00534             // added 2024-02-19
00535             fnSetMetadataMulti          = GetPrefixSymbol< decltype(sound4cl_SetMetadataMulti
00536     ) > ("SetMetadataMulti"            );
00537         } catch (std::runtime_error& ) {
00538             // Ignored, handler will take care
00539         }
00540         // Other optional, web server only
00541         try {
00542             // added 2024-05-23
00543             fnWebserver_SetAppHealth    = GetPrefixSymbol<
decltype(sound4cl_Webserver_SetAppHealth            ) > ("Webserver_SetAppHealth"            );
00546             fnWebserver_GetAppHealth    = GetPrefixSymbol<
decltype(sound4cl_Webserver_GetAppHealth            ) > ("Webserver_GetAppHealth"            );
00547             fnWebserver_FreeString      = GetPrefixSymbol< decltype(sound4cl_Webserver_FreeString
    ) > ("Webserver_FreeString"          );
00548         } catch (std::runtime_error& ) {
00549             // Ignored, handler will take care
00550         }
00551         m_bOK=true;
00552         return true;
00553     }

```

```

00554     }
00561     bool IsOk() const {
00562         return m_bOK;
00563     }
00564
00569     void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
sound4cl_SampleFormat fmt)
00570     { fnAudioConvertFrom(payload, output, nSpl, fmt); }
00571     void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat
fmt)
00572     { fnAudioConvertTo(input, payload, nSpl, fmt); }
00573     void StereoToMono(const float *input, float *output, size_t nSpl)
00574     { fnStereoToMono(input, output, nSpl); }
00575     void MonoToStereo(const float *input, float *output, size_t nSpl)
00576     { fnMonoToStereo(input, output, nSpl); }
00577     void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl)
00578     { fnStereoToMono_Planar(inputL, inputR, output, nSpl); }
00579     void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl)
00580     { fnMonoToStereo_Planar(input, outputL, outputR, nSpl); }
00581     void AudioMonoFromLiveStereo(const uint8_t *payload, float *output)
00582     { fnAudioMonoFromLiveStereo(payload, output); }
00583     void AudioMonoToLiveStereo(const float *input, uint8_t *payload)
00584     { fnAudioMonoToLiveStereo(input, payload); }
00585
00591     std::string GetVersion()
00592     { return std::string(fnGetVersion()); }
00593
00602     unsigned int GetChunkSizeInFrames()
00603     { return fnGetChunkSizeInFrames(); }
00604
00613     std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00614     {
00615         std::vector<unsigned int> list;
00616         if (fnGetPossibleChunkSizeInFrames) {
00617             for (unsigned int* src=fnGetPossibleChunkSizeInFrames();*src;src++) {
00618                 list.push_back(*src);
00619             }
00620         } else {
00621             list.push_back(fnGetChunkSizeInFrames());
00622         }
00623         return list;
00624     }
00625
00631     unsigned int GetChannelCount()
00632     { return fnGetChannelCount(); }
00633
00641     unsigned int GetAudioInputCount()
00642     { return fnGetAudioInputCount(); }
00643
00651     unsigned int GetAudioOutputCount()
00652     { return fnGetAudioOutputCount(); }
00653
00661     unsigned int GetSampleRate()
00662     { return fnGetSampleRate(); }
00663
00670     std::string GetFormatName(const sound4cl_SampleFormat fmt)
00671     { return std::string(fnGetFormatName(fmt)); }
00672
00679     sound4cl_SampleFormat GetFormatFromName(const std::string& name)
00680     { return fnGetFormatFromName(name.c_str()); }
00681
00688     unsigned int GetBytesFromFormat(const sound4cl_SampleFormat fmt)
00689     { return fnGetBytesFromFormat(fmt); }
00690
00694     void SetLogSeverity(LogSeverity severity)
00695     { fnSetLogSeverity(severity); }
00696
00700     void SetLoggerCallback(log_cb_t cb)
00701     { _log_cb=cb; fnSetLoggerCallback(&_log_cb_c); }
00702 };
00703
00709     class CBus {
00710     public:
00711         CBus(CProcessor& dynlib)
00712             : m_dynlib(dynlib)
00713         {
00714             if (m_dynlib.fnNewBus) {
00715                 m_bus=m_dynlib.fnNewBus();
00716             }
00717         }
00718         ~CBus()
00719         {
00720             if (m_bus && m_dynlib.fnFreeBus) {
00721                 m_dynlib.fnFreeBus(m_bus);
00722             }
00723             m_bus = nullptr;
00724         }

```

```

00725         sound4cl_CBus *Get() const { return m_bus; }
00726     private:
00727         CProcessor& m_dynlib;
00728         sound4cl_CBus *m_bus;
00729     };
00730
00731     class CPresetLoader {
00732     public:
00733         CPresetLoader() = default;
00734         virtual ~CPreloader() = default;
00735
00736         virtual bool IsReadOnly() = 0;
00737         virtual bool Exists(const std::filesystem::path &name) = 0;
00738         virtual bool Remove(const std::filesystem::path &name) = 0;
00739         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00740         virtual std::vector<std::filesystem::path> GetAll() = 0;
00741         virtual std::string Read(const std::filesystem::path &filename) = 0;
00742         virtual bool Write(const std::filesystem::path &filename, const std::string &content) = 0;
00743     };
00744     using PresetChange_Kind = sound4cl_PresetChange_Kind;
00745
00746     class CInstance {
00747     public:
00748         CInstance(CProcessor& dynlib)
00749         : m_dynlib(dynlib)
00750         {
00751             params = m_dynlib.fnNewParameters();
00752             if (!params) throw std::bad_alloc();
00753         }
00754         // non-copyable
00755         CInstance( const CInstance& ) = delete;
00756         CInstance& operator=( const CInstance& ) = delete;
00757         virtual ~CInstance()
00758         {
00759             Stop();
00760             if (params) m_dynlib.fnFreeParameters(params);
00761         }
00762     bool IsOk() const { return instance!=nullptr; }
00763     void SetParam(const std::string& name, const std::string& value)
00764     {
00765         assert(params);
00766         m_dynlib.fnSetParameter(params, name.c_str(), value.c_str());
00767     }
00768     std::string GetParam(const std::string& name)
00769     {
00770         assert(params);
00771         auto c_value=m_dynlib.fnGetParameter(params, name.c_str());
00772         std::string ret(c_value);
00773         m_dynlib.fnFreeParameterValue(c_value);
00774         return ret;
00775     }
00776     void SetBus(const CBus& bus) {
00777         assert(params);
00778         if (m_dynlib.fnSetInstanceBus) {
00779             m_dynlib.fnSetInstanceBus(params,bus.Get());
00780         }
00781     }
00782     void SetPresetManager(CPresetLoader *preset_manager)
00783     {
00784         if (!preset_manager) return;
00785         if (!m_dynlib.fnSetPresetManager) return;
00786         assert(params);
00787         m_dynlib.fnSetPresetManager(params,
00788             sound4cl_custom_reader,
00789             sound4cl_custom_reader_free,
00790             sound4cl_custom_writer,
00791             sound4cl_custom_exists,
00792             sound4cl_custom_getall,
00793             sound4cl_custom_getall_free,
00794             sound4cl_custom_remove,
00795             sound4cl_custom_rename,
00796             preset_manager->IsReadOnly(),
00797             preset_manager
00798         );
00799     }
00800
00801     bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
00802     Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
00803     json_port = 0, unsigned int frames_per_chunk=0)
00804     {
00805         assert(!instance);
00806         std::string l_save_path;
00807         if (!save_path.empty()) {
00808             l_save_path = save_path.u8string();
00809         }
00810     }

```

```

00867         if (frames_per_chunk==0) {
00868             frames_per_chunk=m_dynlib.GetChunkSizeInFrames();
00869         }
00870         if (m_dynlib.fnInitProcess3) {
00871             instance = m_dynlib.fnInitProcess3(LoginKey.c_str(), RadioName.c_str(),
Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
00872         } else if (frames_per_chunk!=m_dynlib.GetChunkSizeInFrames()) {
00873             return false;
00874         } else {
00875             instance = m_dynlib.fnInitProcess2(LoginKey.c_str(), RadioName.c_str(),
Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
00876         }
00877         if (!instance) return false;
00878         if (!init_metadata.empty()) {
00879             for (auto&& [key,value]: init_metadata) {
00880                 m_dynlib.fnSetMetadata(instance, key.c_str(), value.c_str());
00881             }
00882             init_metadata.clear();
00883         }
00884         update_thread = std::thread([this, json_port]() {
00885             OnUpdateThreadStart();
00886             m_dynlib.fnStartUpdateThread(instance, json_port);
00887             OnUpdateThreadStop();
00888         });
00889         int timeout=5000;
00890         #ifdef DEBUG
00891             timeout=60*1000; // For debugger pauses
00892         #endif
00893         if (m_dynlib.fnWaitUpdateThreadReady(instance, timeout)<0) {
00894             return false;
00895         }
00896         return true;
00897     }
00898
00906     unsigned int GetChunkFrames() {
00907         if (instance && m_dynlib.fnGetProcessChunkFrames) {
00908             return m_dynlib.fnGetProcessChunkFrames(instance);
00909         }
00910         return m_dynlib.fnGetChunkSizeInFrames();
00911     }
00912
00920     void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
change_kind)
00921     {
00922         if (m_dynlib.fnPresetManager_InformChange) {
00923             m_dynlib.fnPresetManager_InformChange(instance, relative_path.c_str(),
(sound4cl_PresetChange_Kind) change_kind);
00924         }
00925     }
00926
00938     void SetMetadata(const std::string& key, const char* value)
00939     {
00940         if (m_dynlib.fnSetMetadata) {
00941             if (instance) {
00942                 m_dynlib.fnSetMetadata(instance, key.c_str(), value);
00943             } else {
00944                 init_metadata.push_back( {key, value} );
00945             }
00946         }
00947     }
00948
00949
00959     void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00960     {
00961         if (instance) {
00962             if (m_dynlib.fnSetMetadataMulti) {
00963                 typedef const char* pchar;
00964                 pchar* keyvalue=new pchar[2*list.size()+1];
00965                 size_t n=0;
00966                 for (auto&& [key,value]: list) {
00967                     keyvalue[2*n+0]=key.c_str();
00968                     keyvalue[2*n+1]=value;
00969                     n++;
00970                 }
00971                 keyvalue[2*n+0]=nullptr;
00972
00973                 sound4cl_SetMetadataMulti(instance, keyvalue);
00974
00975                 delete[] keyvalue;
00976             } else {
00977                 for (auto&& [key,value]: list) {
00978                     SetMetadata(key, value);
00979                 }
00980             }
00981         } else {
00982             for (auto&& [key,value]: list) {
00983                 if (value) {

```

```

00984         init_metadata.push_back( {key, value} );
00985     }
00986 }
00987 }
00988 }
00989
00995 std::vector< std::tuple<std::string, std::string> > GetMetadataInfos()
00996 {
00997     std::vector< std::tuple<std::string, std::string> > values;
00998     if (m_dynlib.fnGetMetadataInfos) {
00999         const char** c_values = m_dynlib.fnGetMetadataInfos(instance);
01000         if (c_values) {
01001             for (const char** c_value=c_values; *c_value; ) {
01002                 std::string key(*c_value);
01003                 c_value++;
01004                 if (*c_value) {
01005                     std::string descr(*c_value);
01006                     c_value++;
01007                     values.push_back( {key, descr} );
01008                 }
01009             }
01010             m_dynlib.fnFreeMetadataInfos(instance, c_values);
01011         }
01012     }
01013     return values;
01014 }
01015
01016 // Can be overloaded, called at start of UpdateThread in the thread
01017 virtual void OnUpdateThreadStart() {}
01018 // Can be overloaded to call at end of UpdateThread in the thread
01019 virtual void OnUpdateThreadStop() {}
01020
01026 bool StartWebServer(int http_port, int https_port=0)
01027 {
01028     assert(instance);
01029     assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01030     if (!m_dynlib.fnWebserver) {
01031         return false;
01032     }
01033     webserver = m_dynlib.fnWebserver(http_port, https_port, instance);
01034     if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01035         return false;
01036     }
01037     int web_status=m_dynlib.fnWebserver_Status(webserver);
01038     if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01039         return false;
01040     }
01041     if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01042         return false;
01043     }
01044     return true;
01045 }
01051 void StopWebServer()
01052 {
01053     if (!m_dynlib.fnWebserver_Stop) return;
01054     if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01055         m_dynlib.fnWebserver_Stop(webserver, 1000);
01056         webserver = SOUND4_INVALID_WEBSERVER_ID;
01057     }
01058 }
01059
01063 void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
content)
01064 {
01065     assert(instance);
01066     if (!m_dynlib.fnWebserver_SetAppHealth) return;
01067     m_dynlib.fnWebserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
content.c_str());
01068 }
01072 void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01073 {
01074     assert(instance);
01075     if (!m_dynlib.fnWebserver_GetAppHealth) return;
01076     char* c_contenttype=NULLPTR;
01077     char* c_content=NULLPTR;
01078     m_dynlib.fnWebserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01079     contenttype=c_contenttype;
01080     content=c_content;
01081     m_dynlib.fnWebserver_FreeString(c_contenttype);
01082     m_dynlib.fnWebserver_FreeString(c_content);
01083 }
01090 int Terminate()
01091 {
01092     if (!instance) return 0;
01093     return m_dynlib.fnTerminateProcess(instance);
01094 }
01095

```

```

01101     void Stop()
01102     {
01103         if (!instance) return;
01104         StopWebServer();
01105         if (update_thread.joinable()) {
01106             m_dynlib.fnStopUpdateThread(instance); // this will exit the update thread
01107             update_thread.join();
01108         }
01109         m_dynlib.fnExitProcess(instance);
01110         instance=nullptr;
01111     }
01112
01118     unsigned int GetEstimatedDelay()
01119     {
01120         assert(instance);
01121         return m_dynlib.fnGetEstimatedDelay(instance);
01122     }
01123
01134     float* GetBufferIn()
01135     {
01136         assert(instance);
01137         return m_dynlib.fnGetBufferIn(instance);
01138     }
01139
01150     float* GetBufferOut()
01151     {
01152         assert(instance);
01153         return m_dynlib.fnGetBufferOut(instance);
01154     }
01160     void ProcessAudio()
01161     {
01162         assert(instance);
01163         m_dynlib.fnProcessAudio(instance, m_dynlib.fnGetBufferIn(instance),
01164             m_dynlib.fnGetBufferOut(instance));
01165     }
01178     // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
01179     // InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01180     void ProcessAudio_Planar(float const * const *input, float * const *output)
01181     {
01182         assert(instance);
01183         m_dynlib.fnProcessAudio_Planar(instance, input, output);
01184     }
01196     template<typename T>
01197     std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01198     {
01199         assert(instance);
01200         std::vector<T> output;
01201         unsigned int out_offset=0;
01202         unsigned int in_offset=0;
01203         unsigned int todo = input.size();
01204         while (todo>0) {
01205             unsigned int left = AddAudio(&input[in_offset], todo);
01206             unsigned int out_avail = m_dynlib.fnGetOutputCount(instance);
01207             output.resize(out_offset + out_avail);
01208             GetAudio(&output[out_offset], out_avail);
01209             out_offset+=out_avail;
01210             in_offset += todo-left;
01211             todo=left;
01212         }
01213         return output;
01214     }
01215
01221     class CClient {
01222     public:
01223         CClient(sound4cl_CInstance* instance, CProcessor& dynlib)
01224             : m_dynlib(dynlib)
01225         {
01226             assert(instance);
01227             client=m_dynlib.fnNewClient(instance);
01228             if (!client) throw std::bad_alloc();
01229         }
01230         // non-copyable
01231         CClient( const CInstance& ) = delete;
01232         CClient& operator=( const CInstance& ) = delete;
01233         ~CClient()
01234         {
01235             if (client) {
01236                 m_dynlib.fnDeleteClient(client);
01237                 client=nullptr;
01238             }
01239         }
01247     std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01248     {
01249         assert(client);
01250         int need_save=0;
01251         const char *canswer = m_dynlib.fnProcessJson(client, request.c_str()),

```

```

&need_save);
01252         if (!canswer) return {};
01253         std::string answer(canswer);
01254         m_dynlib.fnFreeJsonAnswer (canswer);
01255         if (NeedSave) {
01256             *NeedSave=(need_save!=0);
01257         }
01258         return answer;
01259     }
01260     private:
01261         CProcessor& m_dynlib;
01262         sound4cl_CClientInstance *client = nullptr;
01263     };
01271     std::shared_ptr<CClient> NewClient ()
01272     {
01273         assert(instance);
01274         return std::make_shared<CClient>(instance, m_dynlib);
01275     }
01276
01277     protected:
01278         template<typename T>
01279         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01280         {
01281             assert(instance);
01282             return m_dynlib.fnAddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
01283 helper::SampleFormat<T>::format);
01284         }
01285         template<typename T>
01286         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01287         {
01288             assert(instance);
01289             return m_dynlib.fnGetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
01290 helper::SampleFormat<T>::format);
01291         }
01292     private:
01293         CProcessor& m_dynlib;
01294         sound4cl_CParameters* params = nullptr;
01295         sound4cl_CInstance* instance = nullptr;
01296         std::thread update_thread;
01297         std::vector< std::pair<std::string, std::string> > init_metadata;
01298         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01299     };
01300 }; // namespace sound4
01301
01302 // bridge C callbacks to CPresetLoader
01303 extern "C" {
01304     static char *sound4cl_custom_reader(const fs_char *filename, void* userdata) {
01305         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01306         std::string content=preset_loader->Read(filename);
01307         return strdup(content.c_str());
01308     }
01309     static void sound4cl_custom_reader_free(char *content, void* userdata) {
01310         if (!content) return;
01311         free(content);
01312     }
01313     static int sound4cl_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01314         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01315         auto res=preset_loader->Write(filename,content);
01316         return res?0:-1;
01317     }
01318     static int sound4cl_custom_exists(const fs_char *filename, void* userdata) {
01319         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01320         auto res=preset_loader->Exists(filename);
01321         return res?0:-1;
01322     }
01323     static fs_char** sound4cl_custom_getall(void* userdata) {
01324         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01325         auto res=preset_loader->GetAll();
01326         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01327         for (size_t n=0;n<res.size();n++) {
01328             all[n]=fs_strdup(res[n].c_str());
01329         }
01330         all[res.size()]=nullptr;
01331         return all;
01332     }
01333     static void sound4cl_custom_getall_free(fs_char** all, void* userdata) {
01334         if (!all) return;
01335         for (fs_char** one=all;*one;one++) {
01336             free(*one);
01337         }
01338         free(all);
01339     }
01340     static int sound4cl_custom_remove(const fs_char *filename, void* userdata) {
01341         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01342

```

```

01343         auto res=preset_loader->Remove(filename);
01344         return res?0:-1;
01345     }
01346     static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01347         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01348         auto res=preset_loader->Rename(from,to);
01349         return res?0:-1;
01350     }
01351 }
01352 };

```

11.17 sound4cl_cdef.h File Reference

C function signature definition for all sound4.cl libraries.

```

#include <stdint.h>
#include <stringapiset.h>
#include <wchar.h>

```

Classes

- struct [sound4cl_SOUND4_ProcessInfo](#)

Macros

- #define [fs_strdup](#) wcsdup
helper for OS dependant file name copy
- #define [SOUND4_INVALID_WEBSERVER_ID](#) ((uint64_t)-1)
- #define [SOUND4_WEBSERVER_HTTP_OK](#) (1<<0)
- #define [SOUND4_WEBSERVER_HTTPS_OK](#) (1<<1)

Typedefs

- typedef wchar_t [fs_char](#)
- typedef void(* [sound4cl_loggerfn](#)) (enum [sound4cl_LogSeverity](#), const char *)
- typedef char *(* [sound4cl_storage_reader](#)) (const [fs_char](#) *filename, void *userdata)
- typedef void(* [sound4cl_storage_reader_free](#)) (char *content, void *userdata)
- typedef int(* [sound4cl_storage_writer](#)) (const [fs_char](#) *filename, const char *content, void *userdata)
- typedef int(* [sound4cl_storage_exists](#)) (const [fs_char](#) *filename, void *userdata)
- typedef [fs_char](#) *(* [sound4cl_storage_getall](#)) (void *userdata)
- typedef void(* [sound4cl_storage_getall_free](#)) ([fs_char](#) **all, void *userdata)
- typedef int(* [sound4cl_storage_remove](#)) (const [fs_char](#) *filename, void *userdata)
- typedef int(* [sound4cl_storage_rename](#)) (const [fs_char](#) *from, const [fs_char](#) *to, void *userdata)

Enumerations

- enum [sound4cl_LogSeverity](#) {
 [sound4cl_none](#) = 0 , [sound4cl_fatal](#) = 1 , [sound4cl_error](#) = 2 , [sound4cl_warning](#) = 3 ,
 [sound4cl_info](#) = 4 , [sound4cl_verbose](#) = 5 , [sound4cl_verbose2](#) = 6 , [sound4cl_verbose3](#) = 7 ,
 [sound4cl_verbose4](#) = 8 , [sound4cl_verbose5](#) = 9 }
- enum [sound4cl_PresetChange_Kind](#) { [change_kind_created](#) = 1 , [change_kind_modified](#) , [change_kind_deleted](#) }
- enum [sound4cl_SampleFormat](#) {
 [INVALID_FORMAT](#) =0 , [S16_LE](#) , [S16_BE](#) , [S24_LE](#) ,
 [S24_BE](#) , [S32_LE](#) , [S32_BE](#) , [F32_LE](#) ,
 [F32_BE](#) , [S16_NATIVE](#) =[S16_LE](#) , [S24_NATIVE](#) =[S24_LE](#) , [S32_NATIVE](#) =[S32_LE](#) ,
 [F32_NATIVE](#) =[F32_LE](#) }

Functions

- const struct [sound4cl_SOUND4_ProcessInfo](#) * [sound4cl_SOUND4_GetProcessInfo](#) ()
- const char * [sound4cl_GetVersion](#) ()
- unsigned int [sound4cl_GetChunkSizeInFrames](#) ()
- unsigned int * [sound4cl_GetPossibleChunkSizeInFrames](#) ()
- unsigned int [sound4cl_GetChannelCount](#) ()
- unsigned int [sound4cl_GetAudioInputCount](#) ()
- unsigned int [sound4cl_GetAudioOutputCount](#) ()
- unsigned int [sound4cl_GetSampleRate](#) ()
- void [sound4cl_SetLoggerCallback](#) ([sound4cl_loggerfn](#) logger)
- void [sound4cl_SetLogSeverity](#) (enum [sound4cl_LogSeverity](#) severity)
- struct [sound4cl_CParameters](#) * [sound4cl_NewParameters](#) ()
- void [sound4cl_FreeParameters](#) (struct [sound4cl_CParameters](#) *params)
- void [sound4cl_SetParameter](#) (struct [sound4cl_CParameters](#) *params, const char *name, const char *value)
- const char * [sound4cl_GetParameter](#) (struct [sound4cl_CParameters](#) *params, const char *name)
- void [sound4cl_FreeParameterValue](#) (const char *value)
- struct [sound4cl_CInstance](#) * [sound4cl_InitProcess](#) (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct [sound4cl_CInstance](#) * [sound4cl_InitProcess2](#) (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct [sound4cl_CParameters](#) *parameters)
- struct [sound4cl_CInstance](#) * [sound4cl_InitProcess3](#) (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct [sound4cl_CParameters](#) *parameters, unsigned int frames_per_chunk)
- unsigned int [sound4cl_GetProcessChunkFrames](#) (struct [sound4cl_CInstance](#) *instance)
- int [sound4cl_TerminateProcess](#) (struct [sound4cl_CInstance](#) *instance)
- void [sound4cl_ExitProcess](#) (struct [sound4cl_CInstance](#) *instance)
- struct [sound4cl_CBus](#) * [sound4cl_NewBus](#) ()
- void [sound4cl_FreeBus](#) (struct [sound4cl_CBus](#) *bus)
- void [sound4cl_SetInstanceBus](#) (struct [sound4cl_CParameters](#) *params, struct [sound4cl_CBus](#) *bus)
- void [sound4cl_SetPresetManager](#) (struct [sound4cl_CParameters](#) *params, [sound4cl_storage_reader](#) reader, [sound4cl_storage_reader_free](#), [sound4cl_storage_writer](#) writer, [sound4cl_storage_exists](#) exists, [sound4cl_storage_getall](#) getall, [sound4cl_storage_getall_free](#) getall_free, [sound4cl_storage_remove](#) remove, [sound4cl_storage_rename](#) rename, int IsReadOnly, void *userdata)
- void [sound4cl_PresetManager_InformChange](#) (struct [sound4cl_CInstance](#) *instance, const [fs_char](#) *relative_path, enum [sound4cl_PresetChange_Kind](#) change_kind)
- void [sound4cl_SetMetadata](#) (struct [sound4cl_CInstance](#) *instance, const char *key, const char *value)
- void [sound4cl_SetMetadataMulti](#) (struct [sound4cl_CInstance](#) *instance, const char **keyvalue)
- const char ** [sound4cl_GetMetadataInfos](#) (struct [sound4cl_CInstance](#) *instance)
- void [sound4cl_FreeMetadataInfos](#) (struct [sound4cl_CInstance](#) *instance, const char **infos)
- void [sound4cl_StartUpdateThread](#) (struct [sound4cl_CInstance](#) *instance, unsigned int port)
- void [sound4cl_StopUpdateThread](#) (struct [sound4cl_CInstance](#) *instance)
- int [sound4cl_WaitUpdateThreadReady](#) (struct [sound4cl_CInstance](#) *instance, int milliseconds)
- void [sound4cl_ProcessAudio](#) (struct [sound4cl_CInstance](#) *instance, const float *input, float *output)
- void [sound4cl_ProcessAudio_Planar](#) (struct [sound4cl_CInstance](#) *instance, float const *const *input, float *const *output)
- float * [sound4cl_GetBufferIn](#) (struct [sound4cl_CInstance](#) *instance)
- float * [sound4cl_GetBufferOut](#) (struct [sound4cl_CInstance](#) *instance)
- unsigned int [sound4cl_GetEstimatedDelay](#) (struct [sound4cl_CInstance](#) *instance)
- const char * [sound4cl_GetFormatName](#) (const enum [sound4cl_SampleFormat](#) fmt)
- enum [sound4cl_SampleFormat](#) [sound4cl_GetFormatFromName](#) (const char *name)
- unsigned int [sound4cl_GetBytesFromFormat](#) (const enum [sound4cl_SampleFormat](#) fmt)
- unsigned int [sound4cl_GetMaxPacketFrame](#) (struct [sound4cl_CInstance](#) *instance)

- unsigned int [sound4cl_AddAudio](#) (struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum [sound4cl_SampleFormat](#) fmt)
- unsigned int [sound4cl_AddPadAudio](#) (struct sound4cl_CInstance *instance)
- unsigned int [sound4cl_GetOutputCount](#) (struct sound4cl_CInstance *instance)
- unsigned int [sound4cl_GetAudio](#) (struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int max←_nFrame, enum [sound4cl_SampleFormat](#) fmt)
- void [sound4cl_AudioConvertFrom](#) (const uint8_t *payload, float *output, size_t nSpl, enum [sound4cl_SampleFormat](#) fmt)
- void [sound4cl_AudioConvertTo](#) (const float *input, uint8_t *payload, size_t nSpl, enum [sound4cl_SampleFormat](#) fmt)
- void [sound4cl_StereoToMono](#) (const float *input, float *output, size_t nFrame)
- void [sound4cl_MonoToStereo](#) (const float *input, float *output, size_t nFrame)
- void [sound4cl_StereoToMono_Planar](#) (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void [sound4cl_MonoToStereo_Planar](#) (const float *input, float *outputL, float *outputR, size_t nFrame)
- void [sound4cl_AudioMonoFromLiveStereo](#) (const uint8_t *payload, float *output)
- void [sound4cl_AudioMonoToLiveStereo](#) (const float *input, uint8_t *payload)
- struct sound4cl_CClientInstance * [sound4cl_NewClient](#) (struct sound4cl_CInstance *instance)
- void [sound4cl_DeleteClient](#) (struct sound4cl_CClientInstance *client)
- const char * [sound4cl_ProcessJson](#) (struct sound4cl_CClientInstance *client, const char *json_str, int *need_save)
- void [sound4cl_FreeJsonAnswer](#) (const char *json_str)
- int [sound4cl_SaveState](#) (struct sound4cl_CInstance *instance)
- uint64_t [sound4cl_Webserver_tcp](#) (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t [sound4cl_Webserver_tcp2](#) (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct sound4cl_CParameters *parameters)
- uint64_t [sound4cl_Webserver](#) (unsigned int listenport, unsigned int listenport_secure, struct sound4cl_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 #endif
00092 };
00093
00094 const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo();
00095 const char *sound4cl_GetVersion();
00096 unsigned int sound4cl_GetChunkSizeInFrames();
00097 unsigned int* sound4cl_GetPossibleChunkSizeInFrames();
00098 unsigned int sound4cl_GetChannelCount();
00099 unsigned int sound4cl_GetAudioInputCount();
00100 unsigned int sound4cl_GetAudioOutputCount();
00101 unsigned int sound4cl_GetSampleRate();
00102 typedef void (*sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *);
00103 void sound4cl_SetLoggerCallback( sound4cl_loggerfn logger );
00104 void sound4cl_SetLogSeverity( enum sound4cl_LogSeverity severity );
00105 struct sound4cl_CParameters *sound4cl_NewParameters();
00106 void sound4cl_FreeParameters(struct sound4cl_CParameters *params);
00107 void sound4cl_SetParameter(struct sound4cl_CParameters *params, const char *name, const char *value);
00108 const char *sound4cl_GetParameter(struct sound4cl_CParameters *params, const char *name);
00109 void sound4cl_FreeParameterValue(const char *value);
00110 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, [79](#)
- ~CClient
 - sound4::CInstance::CClient, [81](#)
 - sound4::impact::CInstance::CClient, [82](#)
 - sound4::impact::dyn::CInstance::CClient, [84](#)
- ~CDynLoader
 - sound4::helper::CDynLoader, [97](#)
 - sound4::impact::dyn::helper::CDynLoader, [100](#)
- ~CInstance
 - sound4::CInstance, [103](#)
 - sound4::impact::CInstance, [113](#)
 - sound4::impact::dyn::CInstance, [122](#)
- ~CPresetLoader
 - sound4::CPresetLoader, [131](#)
 - sound4::impact::CPresetLoader, [133](#)
 - sound4::impact::dyn::CPresetLoader, [134](#)
- ~MyPresetManager
 - MyPresetManager, [158](#), [159](#)
- AddAudio
 - sound4::CInstance, [103](#)
 - sound4::impact::CInstance, [113](#)
 - sound4::impact::dyn::CDynLib, [87](#)
 - sound4::impact::dyn::CInstance, [122](#)
- AddPadAudio
 - sound4::impact::dyn::CDynLib, [87](#)
- Audio convert helpers, [46](#)
 - impact_AudioConvertFrom, [46](#)
 - impact_AudioConvertTo, [46](#)
 - impact_AudioMonoFromLiveStereo, [47](#)
 - impact_AudioMonoToLiveStereo, [47](#)
 - impact_MonoToStereo, [47](#)
 - impact_MonoToStereo_Planar, [48](#)
 - impact_StereoToMono, [48](#)
 - impact_StereoToMono_Planar, [49](#)
- Audio managing (any audio), [42](#)
 - F32_BE, [43](#)
 - F32_LE, [43](#)
 - F32_NATIVE, [43](#)
 - impact_AddAudio, [43](#)
 - impact_AddPadAudio, [43](#)
 - impact_GetAudio, [44](#)
 - impact_GetBytesFromFormat, [44](#)
 - impact_GetFormatFromName, [44](#)
 - impact_GetFormatName, [45](#)
 - impact_GetMaxPacketFrame, [45](#)
 - impact_GetOutputCount, [45](#)
 - impact_SampleFormat, [42](#)
 - INVALID_FORMAT, [43](#)
 - S16_BE, [43](#)
 - S16_LE, [43](#)
 - S16_NATIVE, [43](#)
 - S24_BE, [43](#)
 - S24_LE, [43](#)
 - S24_NATIVE, [43](#)
 - S32_BE, [43](#)
 - S32_LE, [43](#)
 - S32_NATIVE, [43](#)
- AudioConvertFrom
 - sound4::CProcessor, [138](#)
 - sound4::impact::dyn::CDynLib, [87](#)
- AudioConvertTo
 - sound4::CProcessor, [138](#)
 - sound4::impact::dyn::CDynLib, [88](#)
- AudioMonoFromLiveStereo
 - sound4::CProcessor, [139](#)
 - sound4::impact::dyn::CDynLib, [88](#)
- AudioMonoToLiveStereo
 - sound4::CProcessor, [139](#)
 - sound4::impact::dyn::CDynLib, [88](#)
- CBus
 - sound4::CBus, [79](#)
 - sound4::CProcessor, [144](#)
- CClient
 - sound4::CInstance::CClient, [80](#)
 - sound4::impact::CInstance::CClient, [82](#)
 - sound4::impact::dyn::CInstance::CClient, [84](#)
- CDynLib
 - sound4::impact::dyn::CDynLib, [86](#)
- CDynLoader
 - sound4::helper::CDynLoader, [97](#)
 - sound4::impact::dyn::helper::CDynLoader, [100](#)
- change_kind_created
 - Custom preset management, [35](#)
 - sound4cl_cdef.h, [242](#)
- change_kind_deleted
 - Custom preset management, [35](#)
 - sound4cl_cdef.h, [242](#)
- change_kind_modified
 - Custom preset management, [35](#)
 - sound4cl_cdef.h, [242](#)
- CHANNEL_COUNT
 - sound4.impact.cl-pipe.cpp, [182](#)
- CHUNK_SIZE
 - sound4.impact.cl-pipe.cpp, [182](#)
- ChunkFrames
 - sound4::impact, [74](#)
 - sound4::impact::dyn, [76](#)

- CInstance
 - sound4::CInstance, [103](#)
 - sound4::CProcessor, [144](#)
 - sound4::impact::CInstance, [113](#)
 - sound4::impact::dyn::CInstance, [122](#)
- Client's JSON config interfacing, [49](#)
 - impact_DeleteClient, [50](#)
 - impact_FreeJsonAnswer, [50](#)
 - impact_NewClient, [50](#)
 - impact_ProcessJson, [50](#)
 - impact_SaveState, [51](#)
- Close
 - sound4::helper::CDynLoader, [98](#)
 - sound4::impact::dyn::helper::CDynLoader, [100](#)
- CPresetLoader
 - sound4::CPresetLoader, [131](#)
 - sound4::impact::CPresetLoader, [133](#)
 - sound4::impact::dyn::CPresetLoader, [134](#)
- CProcessor
 - sound4::CProcessor, [138](#)
- Create
 - sound4::CInstance, [103](#)
 - sound4::impact::CInstance, [113](#)
 - sound4::impact::dyn::CInstance, [122](#)
- Custom preset management, [30](#)
 - change_kind_created, [35](#)
 - change_kind_deleted, [35](#)
 - change_kind_modified, [35](#)
 - fs_char, [32](#)
 - impact_PresetChange_Kind, [34](#)
 - impact_PresetManager_InformChange, [35](#)
 - impact_SetPresetManager, [35](#)
 - impact_storage_exists, [32](#)
 - impact_storage_getall, [32](#)
 - impact_storage_getall_free, [32](#)
 - impact_storage_reader, [33](#)
 - impact_storage_reader_free, [33](#)
 - impact_storage_remove, [33](#)
 - impact_storage_rename, [34](#)
 - impact_storage_writer, [34](#)
- DeleteClient
 - sound4::impact::dyn::CDynLib, [88](#)
- Deprecated List, [3](#)
- Direct processing, [39](#)
 - impact_GetBufferIn, [39](#)
 - impact_GetBufferOut, [40](#)
 - impact_GetEstimatedDelay, [40](#)
 - impact_ProcessAudio, [40](#)
 - impact_ProcessAudio_Planar, [41](#)
- dyn, [69](#)
- DynFunc_t
 - sound4::helper, [72](#)
 - sound4::impact::dyn::helper, [78](#)
- DynFuncHolder
 - sound4::helper::DynFuncHolder< T >, [154](#)
 - sound4::impact::dyn::helper::DynFuncHolder< T >, [156](#)
- DynLib_t
 - sound4::helper, [72](#)
 - sound4::impact::dyn::helper, [78](#)
- Environment variables and parameters, [19](#)
- error
 - Global setup, [22](#)
 - sound4, [70](#)
- Exists
 - MyPresetManager, [159](#)
 - sound4::CPresetLoader, [131](#)
 - sound4::impact::CPresetLoader, [133](#)
 - sound4::impact::dyn::CPresetLoader, [135](#)
- ExitProcess
 - sound4::impact::dyn::CDynLib, [88](#)
- F32_BE
 - Audio managing (any audio), [43](#)
 - sound4cl_cdef.h, [242](#)
- F32_LE
 - Audio managing (any audio), [43](#)
 - sound4cl_cdef.h, [242](#)
- F32_NATIVE
 - Audio managing (any audio), [43](#)
 - sound4cl_cdef.h, [242](#)
- fatal
 - Global setup, [22](#)
 - sound4, [70](#)
- fnAddAudio
 - sound4::CProcessor, [144](#)
- fnAddPadAudio
 - sound4::CProcessor, [144](#)
- fnAudioConvertFrom
 - sound4::CProcessor, [144](#)
- fnAudioConvertTo
 - sound4::CProcessor, [145](#)
- fnAudioMonoFromLiveStereo
 - sound4::CProcessor, [145](#)
- fnAudioMonoToLiveStereo
 - sound4::CProcessor, [145](#)
- fnDeleteClient
 - sound4::CProcessor, [145](#)
- fnExitProcess
 - sound4::CProcessor, [145](#)
- fnFreeBus
 - sound4::CProcessor, [145](#)
- fnFreeJsonAnswer
 - sound4::CProcessor, [145](#)
- fnFreeMetadataInfos
 - sound4::CProcessor, [146](#)
- fnFreeParameters
 - sound4::CProcessor, [146](#)
- fnFreeParameterValue
 - sound4::CProcessor, [146](#)
- fnGetAudio
 - sound4::CProcessor, [146](#)
- fnGetAudioInputCount
 - sound4::CProcessor, [146](#)
- fnGetAudioOutputCount
 - sound4::CProcessor, [146](#)

- fnGetBufferIn
 - sound4::CProcessor, [146](#)
- fnGetBufferOut
 - sound4::CProcessor, [147](#)
- fnGetBytesFromFormat
 - sound4::CProcessor, [147](#)
- fnGetChannelCount
 - sound4::CProcessor, [147](#)
- fnGetChunkSizeInFrames
 - sound4::CProcessor, [147](#)
- fnGetEstimatedDelay
 - sound4::CProcessor, [147](#)
- fnGetFormatFromName
 - sound4::CProcessor, [147](#)
- fnGetFormatName
 - sound4::CProcessor, [147](#)
- fnGetMaxPacketFrame
 - sound4::CProcessor, [148](#)
- fnGetMetadataInfos
 - sound4::CProcessor, [148](#)
- fnGetOutputCount
 - sound4::CProcessor, [148](#)
- fnGetParameter
 - sound4::CProcessor, [148](#)
- fnGetPossibleChunkSizeInFrames
 - sound4::CProcessor, [148](#)
- fnGetProcessChunkFrames
 - sound4::CProcessor, [148](#)
- fnGetSampleRate
 - sound4::CProcessor, [148](#)
- fnGetVersion
 - sound4::CProcessor, [149](#)
- fnInitProcess
 - sound4::CProcessor, [149](#)
- fnInitProcess2
 - sound4::CProcessor, [149](#)
- fnInitProcess3
 - sound4::CProcessor, [149](#)
- fnMonoToStereo
 - sound4::CProcessor, [149](#)
- fnMonoToStereo_Planar
 - sound4::CProcessor, [149](#)
- fnNewBus
 - sound4::CProcessor, [149](#)
- fnNewClient
 - sound4::CProcessor, [150](#)
- fnNewParameters
 - sound4::CProcessor, [150](#)
- fnPresetManager_InformChange
 - sound4::CProcessor, [150](#)
- fnProcessAudio
 - sound4::CProcessor, [150](#)
- fnProcessAudio_Planar
 - sound4::CProcessor, [150](#)
- fnProcessJson
 - sound4::CProcessor, [150](#)
- fnSaveState
 - sound4::CProcessor, [150](#)
- fnSetInstanceBus
 - sound4::CProcessor, [151](#)
- fnSetLoggerCallback
 - sound4::CProcessor, [151](#)
- fnSetLogSeverity
 - sound4::CProcessor, [151](#)
- fnSetMetadata
 - sound4::CProcessor, [151](#)
- fnSetMetadataMulti
 - sound4::CProcessor, [151](#)
- fnSetParameter
 - sound4::CProcessor, [151](#)
- fnSetPresetManager
 - sound4::CProcessor, [151](#)
- fnStartUpdateThread
 - sound4::CProcessor, [152](#)
- fnStereoToMono
 - sound4::CProcessor, [152](#)
- fnStereoToMono_Planar
 - sound4::CProcessor, [152](#)
- fnStopUpdateThread
 - sound4::CProcessor, [152](#)
- fnTerminateProcess
 - sound4::CProcessor, [152](#)
- fnWaitUpdateThreadReady
 - sound4::CProcessor, [152](#)
- fnWebserver
 - sound4::CProcessor, [152](#)
- fnWebserver_FreeString
 - sound4::CProcessor, [153](#)
- fnWebserver_GetAppHealth
 - sound4::CProcessor, [153](#)
- fnWebserver_SetAppHealth
 - sound4::CProcessor, [153](#)
- fnWebserver_Status
 - sound4::CProcessor, [153](#)
- fnWebserver_Stop
 - sound4::CProcessor, [153](#)
- fnWebserver_tcp
 - sound4::CProcessor, [153](#)
- fnWebserver_tcp2
 - sound4::CProcessor, [153](#)
- format
 - sound4::helper::SampleFormat< float >, [164](#)
 - sound4::helper::SampleFormat< int16_t >, [165](#)
 - sound4::helper::SampleFormat< int32_t >, [167](#)
 - sound4::impact::dyn::helper::SampleFormat< float >, [164](#)
 - sound4::impact::dyn::helper::SampleFormat< int16_t >, [166](#)
 - sound4::impact::dyn::helper::SampleFormat< int32_t >, [168](#)
 - sound4::impact::helper::SampleFormat< float >, [165](#)
 - sound4::impact::helper::SampleFormat< int16_t >, [166](#)
 - sound4::impact::helper::SampleFormat< int32_t >, [168](#)

- FreeJsonAnswer
 - sound4::impact::dyn::CDynLib, 88
- FreeMetadataInfos
 - sound4::impact::dyn::CDynLib, 88
- FreeParameters
 - sound4::impact::dyn::CDynLib, 89
- FreeParameterValue
 - sound4::impact::dyn::CDynLib, 89
- fs_char
 - Custom preset managagement, 32
 - sound4cl_cdef.h, 240
- fs_strdup
 - sound4.impact.cl.hpp, 199
 - sound4.impact.cl_dyn.hpp, 209
 - sound4cl_cdef.h, 239
- Get
 - sound4::CBus, 80
- GetAll
 - MyPresetManager, 159
 - sound4::CPresetLoader, 131
 - sound4::impact::CPresetLoader, 133
 - sound4::impact::dyn::CPresetLoader, 135
- GetAudio
 - sound4::CInstance, 104
 - sound4::impact::CInstance, 114
 - sound4::impact::dyn::CDynLib, 89
 - sound4::impact::dyn::CInstance, 123
- GetAudioInputCount
 - sound4::CProcessor, 139
 - sound4::impact::dyn::CDynLib, 89
- GetAudioOutputCount
 - sound4::CProcessor, 139
 - sound4::impact::dyn::CDynLib, 89
- GetBufferIn
 - sound4::CInstance, 104
 - sound4::impact::CInstance, 114
 - sound4::impact::dyn::CDynLib, 89
 - sound4::impact::dyn::CInstance, 123
- GetBufferOut
 - sound4::CInstance, 104
 - sound4::impact::CInstance, 114
 - sound4::impact::dyn::CDynLib, 89
 - sound4::impact::dyn::CInstance, 123
- GetBytesFromFormat
 - sound4::CProcessor, 139
 - sound4::impact::dyn::CDynLib, 90
- GetChannelCount
 - sound4::CProcessor, 140
 - sound4::impact::dyn::CDynLib, 90
- GetChunkFrames
 - sound4::CInstance, 104
 - sound4::impact::CInstance, 114
 - sound4::impact::dyn::CInstance, 124
- GetChunkSizeInFrames
 - sound4::CProcessor, 140
 - sound4::impact::dyn::CDynLib, 90
- GetEnv
 - sound4.impact.cl-pipe-cxx.cpp, 173
 - sound4.impact.cl-pipe-dyn.cpp, 177
 - sound4cl-pipe.cpp, 187
- GetEstimatedDelay
 - sound4::CInstance, 105
 - sound4::impact::CInstance, 115
 - sound4::impact::dyn::CDynLib, 90
 - sound4::impact::dyn::CInstance, 124
- GetFormatFromName
 - sound4::CProcessor, 140
 - sound4::impact::dyn::CDynLib, 90
- GetFormatName
 - sound4::CProcessor, 141
 - sound4::impact::dyn::CDynLib, 90
- GetMaxPacketFrame
 - sound4::impact::dyn::CDynLib, 90
- GetMetadataInfos
 - sound4::CInstance, 105
 - sound4::impact::CInstance, 115
 - sound4::impact::dyn::CDynLib, 91
 - sound4::impact::dyn::CInstance, 124
- GetOutputCount
 - sound4::impact::dyn::CDynLib, 91
- GetParam
 - sound4::CInstance, 105
 - sound4::impact::CInstance, 115
 - sound4::impact::dyn::CInstance, 124
- GetParameter
 - sound4::impact::dyn::CDynLib, 91
- GetPossibleChunkSizeInFrames
 - sound4::CProcessor, 141
 - sound4::impact::dyn::CDynLib, 91
- GetPrefixSymbol
 - sound4::CProcessor, 141
- GetProcessChunkFrames
 - sound4::impact::dyn::CDynLib, 91
- GetSampleRate
 - sound4::CProcessor, 142
 - sound4::impact::dyn::CDynLib, 91
- GetSymbol
 - sound4::helper::CDynLoader, 98
 - sound4::impact::dyn::helper::CDynLoader, 100
- GetSymbol_safe
 - sound4::helper::CDynLoader, 98
 - sound4::impact::dyn::helper::CDynLoader, 100
- GetThisLibraryPath
 - sound4::helper::CDynLoader, 98
 - sound4::impact::dyn::helper::CDynLoader, 100
- GetVersion
 - sound4::CProcessor, 142
 - sound4::impact::dyn::CDynLib, 91
- GetWebServerAppHealth
 - sound4::CInstance, 105
 - sound4::impact::CInstance, 115
 - sound4::impact::dyn::CInstance, 125
- Global setup, 20
 - error, 22
 - fatal, 22
 - IMPACT_AUDIO_INPUT_COUNT, 21

- IMPACT_AUDIO_OUTPUT_COUNT, 21
- IMPACT_AUDIOFRAME_COUNT, 21
- IMPACT_CHANNEL_COUNT, 21
- impact_GetAudioInputCount, 23
- impact_GetAudioOutputCount, 23
- impact_GetChannelCount, 23
- impact_GetChunkSizeInFrames, 23
- impact_GetPossibleChunkSizeInFrames, 24
- impact_GetSampleRate, 24
- impact_GetVersion, 24
- impact_loggerfn, 22
- impact_LogSeverity, 22
- IMPACT_SAMPLE_RATE, 21
- impact_SetLoggerCallback, 25
- impact_SetLogSeverity, 25
- info, 22
- none, 22
- SOUND4_GetProcessInfo, 25
- verbose, 22
- verbose2, 22
- verbose3, 22
- verbose4, 22
- verbose5, 22
- warning, 22
- impact_AddAudio
 - Audio managing (any audio), 43
- impact_AddPadAudio
 - Audio managing (any audio), 43
- IMPACT_AUDIO_INPUT_COUNT
 - Global setup, 21
- IMPACT_AUDIO_OUTPUT_COUNT
 - Global setup, 21
- impact_AudioConvertFrom
 - Audio convert helpers, 46
- impact_AudioConvertTo
 - Audio convert helpers, 46
- IMPACT_AUDIOFRAME_COUNT
 - Global setup, 21
- impact_AudioMonoFromLiveStereo
 - Audio convert helpers, 47
- impact_AudioMonoToLiveStereo
 - Audio convert helpers, 47
- impact_CClientInstance, 157
- IMPACT_CHANNEL_COUNT
 - Global setup, 21
- impact_CInstance, 157
- impact_CParameters, 157
- IMPACT_DECL
 - sound4.impact.cl.h, 194
- impact_DeleteClient
 - Client's JSon config interfacing, 50
- impact_ExitProcess
 - Process creation/destruction, 26
- impact_FreeJsonAnswer
 - Client's JSon config interfacing, 50
- impact_FreeMetadataInfos
 - Metadata Management, 36
- impact_FreeParameters
 - Process creation/destruction, 26
- impact_FreeParameterValue
 - Process creation/destruction, 27
- impact_GetAudio
 - Audio managing (any audio), 44
- impact_GetAudioInputCount
 - Global setup, 23
- impact_GetAudioOutputCount
 - Global setup, 23
- impact_GetBufferIn
 - Direct processing, 39
- impact_GetBufferOut
 - Direct processing, 40
- impact_GetBytesFromFormat
 - Audio managing (any audio), 44
- impact_GetChannelCount
 - Global setup, 23
- impact_GetChunkSizeInFrames
 - Global setup, 23
- impact_GetEstimatedDelay
 - Direct processing, 40
- impact_GetFormatFromName
 - Audio managing (any audio), 44
- impact_GetFormatName
 - Audio managing (any audio), 45
- impact_GetMaxPacketFrame
 - Audio managing (any audio), 45
- impact_GetMetadataInfos
 - Metadata Management, 36
- impact_GetOutputCount
 - Audio managing (any audio), 45
- impact_GetParameter
 - Process creation/destruction, 27
- impact_GetPossibleChunkSizeInFrames
 - Global setup, 24
- impact_GetProcessChunkFrames
 - Process creation/destruction, 27
- impact_GetSampleRate
 - Global setup, 24
- impact_GetVersion
 - Global setup, 24
- IMPACT_HAS_CLOUDBUS
 - sound4.impact.cl.h, 194
- IMPACT_HAS_WEBSERVER
 - sound4.impact.cl.h, 194
- impact_InitProcess
 - Process creation/destruction, 27
- impact_InitProcess2
 - Process creation/destruction, 28
- impact_InitProcess3
 - Process creation/destruction, 28
- impact_loggerfn
 - Global setup, 22
- impact_LogSeverity
 - Global setup, 22
- impact_MonoToStereo
 - Audio convert helpers, 47
- impact_MonoToStereo_Planar

- Audio convert helpers, [48](#)
- impact_NewClient
 - Client's JSon config interfacing, [50](#)
- impact_NewParameters
 - Process creation/destruction, [30](#)
- impact_PresetChange_Kind
 - Custom preset management, [34](#)
- impact_PresetManager_InformChange
 - Custom preset management, [35](#)
- impact_ProcessAudio
 - Direct processing, [40](#)
- impact_ProcessAudio_Planar
 - Direct processing, [41](#)
- impact_ProcessJson
 - Client's JSon config interfacing, [50](#)
- IMPACT_SAMPLE_RATE
 - Global setup, [21](#)
- impact_SampleFormat
 - Audio managing (any audio), [42](#)
- impact_SaveState
 - Client's JSon config interfacing, [51](#)
- impact_SetLoggerCallback
 - Global setup, [25](#)
- impact_SetLogSeverity
 - Global setup, [25](#)
- impact_SetMetadata
 - Metadata Management, [37](#)
- impact_SetMetadataMulti
 - Metadata Management, [37](#)
- impact_SetParameter
 - Process creation/destruction, [30](#)
- impact_SetPresetManager
 - Custom preset management, [35](#)
- impact_StartUpdateThread
 - Update Management, [38](#)
- impact_StereoToMono
 - Audio convert helpers, [48](#)
- impact_StereoToMono_Planar
 - Audio convert helpers, [49](#)
- impact_StopUpdateThread
 - Update Management, [38](#)
- impact_storage_exists
 - Custom preset management, [32](#)
- impact_storage_getall
 - Custom preset management, [32](#)
- impact_storage_getall_free
 - Custom preset management, [32](#)
- impact_storage_reader
 - Custom preset management, [33](#)
- impact_storage_reader_free
 - Custom preset management, [33](#)
- impact_storage_remove
 - Custom preset management, [33](#)
- impact_storage_rename
 - Custom preset management, [34](#)
- impact_storage_writer
 - Custom preset management, [34](#)
- impact_TerminateProcess
 - Process creation/destruction, [30](#)
- impact_WaitUpdateThreadReady
 - Update Management, [39](#)
- impact_Webserver
 - Web server, [59](#)
- impact_Webserver_FreeString
 - Web server, [59](#)
- impact_Webserver_GetAppHealth
 - Web server, [60](#)
- impact_Webserver_SetAppHealth
 - Web server, [60](#)
- impact_Webserver_Status
 - Web server, [60](#)
- impact_Webserver_Stop
 - Web server, [61](#)
- impact_Webserver_tcp
 - Web server, [61](#)
- impact_Webserver_tcp2
 - Web server, [62](#)
- info
 - Global setup, [22](#)
 - sound4, [71](#)
- InitProcess
 - sound4::impact::dyn::CDynLib, [92](#)
- InitProcess2
 - sound4::impact::dyn::CDynLib, [92](#)
- InitProcess3
 - sound4::impact::dyn::CDynLib, [92](#)
- InputChannels
 - sound4::impact, [74](#)
 - sound4::impact::dyn, [76](#)
- InputSampleSize
 - sound4::impact, [74](#)
 - sound4::impact::dyn, [76](#)
- INVALID_FORMAT
 - Audio managing (any audio), [43](#)
 - sound4cl_cdef.h, [242](#)
- IsOk
 - sound4::CInstance, [106](#)
 - sound4::CProcessor, [142](#)
 - sound4::helper::CDynLoader, [98](#)
 - sound4::helper::DynFuncHolder< T >, [155](#)
 - sound4::impact::CInstance, [116](#)
 - sound4::impact::dyn::CDynLib, [87](#)
 - sound4::impact::dyn::CInstance, [125](#)
 - sound4::impact::dyn::helper::CDynLoader, [101](#)
 - sound4::impact::dyn::helper::DynFuncHolder< T >, [156](#)
- IsReadOnly
 - MyPresetManager, [159](#), [160](#)
 - sound4::CPresetLoader, [131](#)
 - sound4::impact::CPresetLoader, [133](#)
 - sound4::impact::dyn::CPresetLoader, [135](#)
- JSON syntax, [52](#)
- JSON syntax for internal setup, [55](#)
- Load
 - sound4::CProcessor, [142](#)

- sound4::helper::CDynLoader, 98
 - sound4::impact::dyn::CDynLib, 87
 - sound4::impact::dyn::helper::CDynLoader, 101
- log_cb_t
 - sound4, 70
 - sound4::impact, 73
 - sound4::impact::dyn, 76
- LogSeverity
 - sound4, 70
 - sound4::impact, 73
 - sound4::impact::dyn, 76
- main
 - sound4.impact.cl-pipe-cxx.cpp, 173
 - sound4.impact.cl-pipe-dyn.cpp, 177
 - sound4.impact.cl-pipe.cpp, 182
 - sound4cl-pipe.cpp, 187
- Metadata Management, 36
 - impact_FreeMetadataInfos, 36
 - impact_GetMetadataInfos, 36
 - impact_SetMetadata, 37
 - impact_SetMetadataMulti, 37
- MonoToStereo
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 92
- MonoToStereo_Planar
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 92
- MyLogger
 - sound4.impact.cl-pipe-cxx.cpp, 174
 - sound4.impact.cl-pipe-dyn.cpp, 177
 - sound4.impact.cl-pipe.cpp, 182
- MyPresetManager, 157
 - ~MyPresetManager, 158, 159
 - Exists, 159
 - GetAll, 159
 - IsReadOnly, 159, 160
 - MyPresetManager, 158
 - Read, 160
 - Remove, 160
 - Rename, 160, 161
 - SetupPath, 161
 - Write, 161
- name
 - SOUND4_ProcessInfo, 169
 - sound4cl_SOUND4_ProcessInfo, 170
- NewClient
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CDynLib, 92
 - sound4::impact::dyn::CInstance, 125
- NewParameters
 - sound4::impact::dyn::CDynLib, 92
- none
 - Global setup, 22
 - sound4, 71
- OnUpdateThreadStart
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CInstance, 125
- OnUpdateThreadStop
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CInstance, 126
- operator bool
 - sound4::helper::CDynLoader, 98
 - sound4::helper::DynFuncHolder< T >, 155
 - sound4::impact::dyn::helper::CDynLoader, 101
 - sound4::impact::dyn::helper::DynFuncHolder< T >, 156
- operator T*
 - sound4::helper::DynFuncHolder< T >, 155
 - sound4::impact::dyn::helper::DynFuncHolder< T >, 156
- operator=
 - sound4::CInstance, 106
 - sound4::CInstance::CClient, 81
 - sound4::helper::CDynLoader, 99
 - sound4::impact::CInstance, 116
 - sound4::impact::CInstance::CClient, 82
 - sound4::impact::dyn::CInstance, 126
 - sound4::impact::dyn::CInstance::CClient, 84
 - sound4::impact::dyn::helper::CDynLoader, 101
- OutputChannels
 - sound4::impact, 74
 - sound4::impact::dyn, 77
- OutputSampleSize
 - sound4::impact, 74
 - sound4::impact::dyn, 77
- prefix
 - SOUND4_ProcessInfo, 169
 - sound4cl_SOUND4_ProcessInfo, 170
- PresetChange_Kind
 - sound4, 70
 - sound4::impact, 73
 - sound4::impact::dyn, 76
- PresetManager_InformChange
 - sound4::CInstance, 106
 - sound4::impact::CInstance, 116
 - sound4::impact::dyn::CDynLib, 93
 - sound4::impact::dyn::CInstance, 126
- Process creation/destruction, 26
 - impact_ExitProcess, 26
 - impact_FreeParameters, 26
 - impact_FreeParameterValue, 27
 - impact_GetParameter, 27
 - impact_GetProcessChunkFrames, 27
 - impact_InitProcess, 27
 - impact_InitProcess2, 28
 - impact_InitProcess3, 28
 - impact_NewParameters, 30
 - impact_SetParameter, 30
 - impact_TerminateProcess, 30
- ProcessAnyAudio
 - sound4::CInstance, 107

- sound4::impact::CInstance, 117
 - sound4::impact::dyn::CInstance, 126
- ProcessAudio
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CDynLib, 93
 - sound4::impact::dyn::CInstance, 127
- ProcessAudio_Planar
 - sound4::CInstance, 107
 - sound4::impact::CInstance, 117
 - sound4::impact::dyn::CDynLib, 93
 - sound4::impact::dyn::CInstance, 127
- ProcessJson
 - sound4::CInstance::CClient, 81
 - sound4::impact::CInstance::CClient, 82
 - sound4::impact::dyn::CDynLib, 93
 - sound4::impact::dyn::CInstance::CClient, 84
- ProcessLoop
 - sound4.impact.cl-pipe-cxx.cpp, 174
 - sound4.impact.cl-pipe-dyn.cpp, 178
 - sound4.impact.cl-pipe.cpp, 183
 - sound4cl-pipe.cpp, 187
- Read
 - MyPresetManager, 160
 - sound4::CPresetLoader, 131
 - sound4::impact::CPresetLoader, 133
 - sound4::impact::dyn::CPresetLoader, 135
- Remove
 - MyPresetManager, 160
 - sound4::CPresetLoader, 132
 - sound4::impact::CPresetLoader, 133
 - sound4::impact::dyn::CPresetLoader, 135
- Rename
 - MyPresetManager, 160, 161
 - sound4::CPresetLoader, 132
 - sound4::impact::CPresetLoader, 133
 - sound4::impact::dyn::CPresetLoader, 135
- S16_BE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S16_LE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S16_NATIVE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S24_BE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S24_LE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S24_NATIVE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S32_BE
 - Audio managing (any audio), 43
- sound4cl_cdef.h, 242
- S32_LE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- S32_NATIVE
 - Audio managing (any audio), 43
 - sound4cl_cdef.h, 242
- Sample using C interface, 17
- Sample using C++ interface wrapper, 63
- Sample using C++ interface wrapper with dynamic library loading, 64
- Sample using Generic C++ interface (multi-process compatible), 66
- SAMPLE_RATE
 - sound4.impact.cl-pipe.cpp, 182
- SampleFormat
 - sound4::impact, 74
 - sound4::impact::dyn, 76
- SampleRate
 - sound4::impact, 75
 - sound4::impact::dyn, 77
- SaveState
 - sound4::impact::dyn::CDynLib, 93
- SetBus
 - sound4::CInstance, 108
- SetLoggerCallback
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 93
- SetLogSeverity
 - sound4::CProcessor, 143
 - sound4::impact::dyn::CDynLib, 93
- SetMetadata
 - sound4::CInstance, 108
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CDynLib, 94
 - sound4::impact::dyn::CInstance, 128
- SetMetadataMulti
 - sound4::CInstance, 109
 - sound4::impact::CInstance, 118
 - sound4::impact::dyn::CDynLib, 94
 - sound4::impact::dyn::CInstance, 128
- SetParam
 - sound4::CInstance, 109
 - sound4::impact::CInstance, 119
 - sound4::impact::dyn::CInstance, 128
- SetParameter
 - sound4::impact::dyn::CDynLib, 94
- SetPresetManager
 - sound4::CInstance, 109
 - sound4::impact::CInstance, 119
 - sound4::impact::dyn::CDynLib, 94
 - sound4::impact::dyn::CInstance, 129
- SetupPath
 - MyPresetManager, 161
- SetWebServerAppHealth
 - sound4::CInstance, 110
 - sound4::impact::CInstance, 119
 - sound4::impact::dyn::CInstance, 129

- shortname
 - SOUND4_ProcessInfo, 169
 - sound4cl_SOUND4_ProcessInfo, 170
- sound4, 69
 - error, 70
 - fatal, 70
 - info, 71
 - log_cb_t, 70
 - LogSeverity, 70
 - none, 71
 - PresetChange_Kind, 70
 - verbose, 71
 - verbose2, 71
 - verbose3, 71
 - verbose4, 71
 - verbose5, 71
 - warning, 71
- SOUND4 IMPACT.CL Library, 1
- sound4.impact.cl-pipe-cxx.cpp, 173, 175
 - GetEnv, 173
 - main, 173
 - MyLogger, 174
 - ProcessLoop, 174
 - Usage, 174
- sound4.impact.cl-pipe-dyn.cpp, 176, 178
 - GetEnv, 177
 - main, 177
 - MyLogger, 177
 - ProcessLoop, 178
 - Usage, 178
- sound4.impact.cl-pipe.cpp, 181, 183
 - CHANNEL_COUNT, 182
 - CHUNK_SIZE, 182
 - main, 182
 - MyLogger, 182
 - ProcessLoop, 183
 - SAMPLE_RATE, 182
 - Usage, 183
- sound4.impact.cl.h, 191, 194
 - IMPACT_DECL, 194
 - IMPACT_HAS_CLOUDBUS, 194
 - IMPACT_HAS_WEBSERVER, 194
- sound4.impact.cl.hpp, 198, 200
 - fs_strdup, 199
- sound4.impact.cl_dyn.hpp, 207, 209
 - fs_strdup, 209
 - UNIXLIKE, 209
- sound4::CBus, 79
 - ~CBus, 79
 - CBus, 79
 - Get, 80
- sound4::CInstance, 101
 - ~CInstance, 103
 - AddAudio, 103
 - CInstance, 103
 - Create, 103
 - GetAudio, 104
 - GetBufferIn, 104
 - GetBufferOut, 104
 - GetChunkFrames, 104
 - GetEstimatedDelay, 105
 - GetMetadataInfos, 105
 - GetParam, 105
 - GetWebServerAppHealth, 105
 - IsOk, 106
 - NewClient, 106
 - OnUpdateThreadStart, 106
 - OnUpdateThreadStop, 106
 - operator=, 106
 - PresetManager_InformChange, 106
 - ProcessAnyAudio, 107
 - ProcessAudio, 107
 - ProcessAudio_Planar, 107
 - SetBus, 108
 - SetMetadata, 108
 - SetMetadataMulti, 109
 - SetParam, 109
 - SetPresetManager, 109
 - SetWebServerAppHealth, 110
 - StartWebServer, 110
 - Stop, 110
 - StopWebServer, 110
 - Terminate, 111
- sound4::CInstance::CClient, 80
 - ~CClient, 81
 - CClient, 80
 - operator=, 81
 - ProcessJson, 81
- sound4::CPresetLoader, 130
 - ~CPresetLoader, 131
 - CPresetLoader, 131
 - Exists, 131
 - GetAll, 131
 - IsReadOnly, 131
 - Read, 131
 - Remove, 132
 - Rename, 132
 - Write, 132
- sound4::CProcessor, 136
 - AudioConvertFrom, 138
 - AudioConvertTo, 138
 - AudioMonoFromLiveStereo, 139
 - AudioMonoToLiveStereo, 139
 - CBus, 144
 - CInstance, 144
 - CProcessor, 138
 - fnAddAudio, 144
 - fnAddPadAudio, 144
 - fnAudioConvertFrom, 144
 - fnAudioConvertTo, 145
 - fnAudioMonoFromLiveStereo, 145
 - fnAudioMonoToLiveStereo, 145
 - fnDeleteClient, 145
 - fnExitProcess, 145
 - fnFreeBus, 145
 - fnFreeJsonAnswer, 145

- fnFreeMetadataInfos, 146
- fnFreeParameters, 146
- fnFreeParameterValue, 146
- fnGetAudio, 146
- fnGetAudioInputCount, 146
- fnGetAudioOutputCount, 146
- fnGetBufferIn, 146
- fnGetBufferOut, 147
- fnGetBytesFromFormat, 147
- fnGetChannelCount, 147
- fnGetChunkSizeInFrames, 147
- fnGetEstimatedDelay, 147
- fnGetFormatFromName, 147
- fnGetFormatName, 147
- fnGetMaxPacketFrame, 148
- fnGetMetadataInfos, 148
- fnGetOutputCount, 148
- fnGetParameter, 148
- fnGetPossibleChunkSizeInFrames, 148
- fnGetProcessChunkFrames, 148
- fnGetSampleRate, 148
- fnGetVersion, 149
- fnInitProcess, 149
- fnInitProcess2, 149
- fnInitProcess3, 149
- fnMonoToStereo, 149
- fnMonoToStereo_Planar, 149
- fnNewBus, 149
- fnNewClient, 150
- fnNewParameters, 150
- fnPresetManager_InformChange, 150
- fnProcessAudio, 150
- fnProcessAudio_Planar, 150
- fnProcessJson, 150
- fnSaveState, 150
- fnSetInstanceBus, 151
- fnSetLoggerCallback, 151
- fnSetLogSeverity, 151
- fnSetMetadata, 151
- fnSetMetadataMulti, 151
- fnSetParameter, 151
- fnSetPresetManager, 151
- fnStartUpdateThread, 152
- fnStereoToMono, 152
- fnStereoToMono_Planar, 152
- fnStopUpdateThread, 152
- fnTerminateProcess, 152
- fnWaitUpdateThreadReady, 152
- fnWebserver, 152
- fnWebserver_FreeString, 153
- fnWebserver_GetAppHealth, 153
- fnWebserver_SetAppHealth, 153
- fnWebserver_Status, 153
- fnWebserver_Stop, 153
- fnWebserver_tcp, 153
- fnWebserver_tcp2, 153
- GetAudioInputCount, 139
- GetAudioOutputCount, 139
- GetBytesFromFormat, 139
- GetChannelCount, 140
- GetChunkSizeInFrames, 140
- GetFormatFromName, 140
- GetFormatName, 141
- GetPossibleChunkSizeInFrames, 141
- GetPrefixSymbol, 141
- GetSampleRate, 142
- GetVersion, 142
- IsOk, 142
- Load, 142
- MonoToStereo, 143
- MonoToStereo_Planar, 143
- SetLoggerCallback, 143
- SetLogSeverity, 143
- StereoToMono, 143
- StereoToMono_Planar, 144
- sound4::helper, 72
 - DynFunc_t, 72
 - DynLib_t, 72
- sound4::helper::CDynLoader, 96
 - ~CDynLoader, 97
 - CDynLoader, 97
 - Close, 98
 - GetSymbol, 98
 - GetSymbol_safe, 98
 - GetThisLibraryPath, 98
 - IsOk, 98
 - Load, 98
 - operator bool, 98
 - operator=, 99
- sound4::helper::DynFuncHolder< T >, 154
 - DynFuncHolder, 154
 - IsOk, 155
 - operator bool, 155
 - operator T*, 155
- sound4::helper::SampleFormat< float >, 163
 - format, 164
- sound4::helper::SampleFormat< int16_t >, 165
 - format, 165
- sound4::helper::SampleFormat< int32_t >, 167
 - format, 167
- sound4::helper::SampleFormat< T >, 162
- sound4::impact, 73
 - ChunkFrames, 74
 - InputChannels, 74
 - InputSampleSize, 74
 - log_cb_t, 73
 - LogSeverity, 73
 - OutputChannels, 74
 - OutputSampleSize, 74
 - PresetChange_Kind, 73
 - SampleFormat, 74
 - SampleRate, 75
- sound4::impact::CInstance, 111
 - ~CInstance, 113
 - AddAudio, 113
 - CInstance, 113

- Create, [113](#)
- GetAudio, [114](#)
- GetBufferIn, [114](#)
- GetBufferOut, [114](#)
- GetChunkFrames, [114](#)
- GetEstimatedDelay, [115](#)
- GetMetadataInfos, [115](#)
- GetParam, [115](#)
- GetWebServerAppHealth, [115](#)
- IsOk, [116](#)
- NewClient, [116](#)
- OnUpdateThreadStart, [116](#)
- OnUpdateThreadStop, [116](#)
- operator=, [116](#)
- PresetManager_InformChange, [116](#)
- ProcessAnyAudio, [117](#)
- ProcessAudio, [117](#)
- ProcessAudio_Planar, [117](#)
- SetMetadata, [118](#)
- SetMetadataMulti, [118](#)
- SetParam, [119](#)
- SetPresetManager, [119](#)
- SetWebServerAppHealth, [119](#)
- StartWebServer, [120](#)
- Stop, [120](#)
- StopWebServer, [120](#)
- Terminate, [120](#)
- sound4::impact::CInstance::CClient, [82](#)
 - ~CClient, [82](#)
 - CClient, [82](#)
 - operator=, [82](#)
 - ProcessJson, [82](#)
- sound4::impact::CPresetLoader, [132](#)
 - ~CPresetLoader, [133](#)
 - CPresetLoader, [133](#)
 - Exists, [133](#)
 - GetAll, [133](#)
 - IsReadOnly, [133](#)
 - Read, [133](#)
 - Remove, [133](#)
 - Rename, [133](#)
 - Write, [134](#)
- sound4::impact::dyn, [75](#)
 - ChunkFrames, [76](#)
 - InputChannels, [76](#)
 - InputSampleSize, [76](#)
 - log_cb_t, [76](#)
 - LogSeverity, [76](#)
 - OutputChannels, [77](#)
 - OutputSampleSize, [77](#)
 - PresetChange_Kind, [76](#)
 - SampleFormat, [76](#)
 - SampleRate, [77](#)
- sound4::impact::dyn::CDynLib, [85](#)
 - AddAudio, [87](#)
 - AddPadAudio, [87](#)
 - AudioConvertFrom, [87](#)
 - AudioConvertTo, [88](#)
 - AudioMonoFromLiveStereo, [88](#)
 - AudioMonoToLiveStereo, [88](#)
 - CDynLib, [86](#)
 - DeleteClient, [88](#)
 - ExitProcess, [88](#)
 - FreeJsonAnswer, [88](#)
 - FreeMetadataInfos, [88](#)
 - FreeParameters, [89](#)
 - FreeParameterValue, [89](#)
 - GetAudio, [89](#)
 - GetAudioInputCount, [89](#)
 - GetAudioOutputCount, [89](#)
 - GetBufferIn, [89](#)
 - GetBufferOut, [89](#)
 - GetBytesFromFormat, [90](#)
 - GetChannelCount, [90](#)
 - GetChunkSizeInFrames, [90](#)
 - GetEstimatedDelay, [90](#)
 - GetFormatFromName, [90](#)
 - GetFormatName, [90](#)
 - GetMaxPacketFrame, [90](#)
 - GetMetadataInfos, [91](#)
 - GetOutputCount, [91](#)
 - GetParameter, [91](#)
 - GetPossibleChunkSizeInFrames, [91](#)
 - GetProcessChunkFrames, [91](#)
 - GetSampleRate, [91](#)
 - GetVersion, [91](#)
 - InitProcess, [92](#)
 - InitProcess2, [92](#)
 - InitProcess3, [92](#)
 - IsOk, [87](#)
 - Load, [87](#)
 - MonoToStereo, [92](#)
 - MonoToStereo_Planar, [92](#)
 - NewClient, [92](#)
 - NewParameters, [92](#)
 - PresetManager_InformChange, [93](#)
 - ProcessAudio, [93](#)
 - ProcessAudio_Planar, [93](#)
 - ProcessJson, [93](#)
 - SaveState, [93](#)
 - SetLoggerCallback, [93](#)
 - SetLogSeverity, [93](#)
 - SetMetadata, [94](#)
 - SetMetadataMulti, [94](#)
 - SetParameter, [94](#)
 - SetPresetManager, [94](#)
 - StartUpdateThread, [94](#)
 - StereoToMono, [94](#)
 - StereoToMono_Planar, [94](#)
 - StopUpdateThread, [95](#)
 - TerminateProcess, [95](#)
 - WaitUpdateThreadReady, [95](#)
 - Webserver, [95](#)
 - Webserver_FreeString, [95](#)
 - Webserver_GetAppHealth, [95](#)
 - Webserver_SetAppHealth, [95](#)

- Webserver_Status, 96
- Webserver_Stop, 96
- Webserver_tcp, 96
- Webserver_tcp2, 96
- sound4::impact::dyn::CInstance, 121
 - ~CInstance, 122
 - AddAudio, 122
 - CInstance, 122
 - Create, 122
 - GetAudio, 123
 - GetBufferIn, 123
 - GetBufferOut, 123
 - GetChunkFrames, 124
 - GetEstimatedDelay, 124
 - GetMetadataInfos, 124
 - GetParam, 124
 - GetWebServerAppHealth, 125
 - IsOk, 125
 - NewClient, 125
 - OnUpdateThreadStart, 125
 - OnUpdateThreadStop, 126
 - operator=, 126
 - PresetManager_InformChange, 126
 - ProcessAnyAudio, 126
 - ProcessAudio, 127
 - ProcessAudio_Planar, 127
 - SetMetadata, 128
 - SetMetadataMulti, 128
 - SetParam, 128
 - SetPresetManager, 129
 - SetWebServerAppHealth, 129
 - StartWebServer, 129
 - Stop, 129
 - StopWebServer, 130
 - Terminate, 130
- sound4::impact::dyn::CInstance::CClient, 83
 - ~CClient, 84
 - CClient, 84
 - operator=, 84
 - ProcessJson, 84
- sound4::impact::dyn::CPresetLoader, 134
 - ~CPresetLoader, 134
 - CPresetLoader, 134
 - Exists, 135
 - GetAll, 135
 - IsReadOnly, 135
 - Read, 135
 - Remove, 135
 - Rename, 135
 - Write, 135
- sound4::impact::dyn::helper, 77
 - DynFunc_t, 78
 - DynLib_t, 78
- sound4::impact::dyn::helper::CDynLoader, 99
 - ~CDynLoader, 100
 - CDynLoader, 100
 - Close, 100
 - GetSymbol, 100
 - GetSymbol_safe, 100
 - GetThisLibraryPath, 100
 - IsOk, 101
 - Load, 101
 - operator bool, 101
 - operator=, 101
- sound4::impact::dyn::helper::DynFuncHolder< T >, 155
 - DynFuncHolder, 156
 - IsOk, 156
 - operator bool, 156
 - operator T*, 156
- sound4::impact::dyn::helper::SampleFormat< float >, 164
 - format, 164
- sound4::impact::dyn::helper::SampleFormat< int16_t >, 165
 - format, 166
- sound4::impact::dyn::helper::SampleFormat< int32_t >, 167
 - format, 168
- sound4::impact::dyn::helper::SampleFormat< T >, 162
- sound4::impact::helper, 78
- sound4::impact::helper::SampleFormat< float >, 164
 - format, 165
- sound4::impact::helper::SampleFormat< int16_t >, 166
 - format, 166
- sound4::impact::helper::SampleFormat< int32_t >, 168
 - format, 168
- sound4::impact::helper::SampleFormat< T >, 163
- SOUND4_GetProcessInfo
 - Global setup, 25
- SOUND4_INVALID_WEBSERVER_ID
 - sound4cl_cdef.h, 239
 - Web server, 58
- SOUND4_ProcessInfo, 168
 - name, 169
 - prefix, 169
 - shortname, 169
 - struct_size, 169
 - version, 169
- SOUND4_WEBSERVER_HTTP_OK
 - sound4cl_cdef.h, 239
 - Web server, 58
- SOUND4_WEBSERVER_HTTPS_OK
 - sound4cl_cdef.h, 240
 - Web server, 59
- sound4cl-pipe.cpp, 186, 188
 - GetEnv, 187
 - main, 187
 - ProcessLoop, 187
- sound4cl.hpp, 222, 224
 - UNIXLIKE, 224
- sound4cl_AddAudio
 - sound4cl_cdef.h, 242
- sound4cl_AddPadAudio
 - sound4cl_cdef.h, 242
- sound4cl_AudioConvertFrom

- sound4cl_cdef.h, 243
- sound4cl_AudioConvertTo
 - sound4cl_cdef.h, 243
- sound4cl_AudioMonoFromLiveStereo
 - sound4cl_cdef.h, 243
- sound4cl_AudioMonoToLiveStereo
 - sound4cl_cdef.h, 243
- sound4cl_cdef.h, 237, 251
 - change_kind_created, 242
 - change_kind_deleted, 242
 - change_kind_modified, 242
 - F32_BE, 242
 - F32_LE, 242
 - F32_NATIVE, 242
 - fs_char, 240
 - fs_strdup, 239
 - INVALID_FORMAT, 242
 - S16_BE, 242
 - S16_LE, 242
 - S16_NATIVE, 242
 - S24_BE, 242
 - S24_LE, 242
 - S24_NATIVE, 242
 - S32_BE, 242
 - S32_LE, 242
 - S32_NATIVE, 242
 - SOUND4_INVALID_WEBSERVER_ID, 239
 - SOUND4_WEBSERVER_HTTP_OK, 239
 - SOUND4_WEBSERVER_HTTPS_OK, 240
- sound4cl_AddAudio, 242
- sound4cl_AddPadAudio, 242
- sound4cl_AudioConvertFrom, 243
- sound4cl_AudioConvertTo, 243
- sound4cl_AudioMonoFromLiveStereo, 243
- sound4cl_AudioMonoToLiveStereo, 243
- sound4cl_DeleteClient, 243
- sound4cl_error, 241
- sound4cl_ExitProcess, 243
- sound4cl_fatal, 241
- sound4cl_FreeBus, 243
- sound4cl_FreeJsonAnswer, 243
- sound4cl_FreeMetadataInfos, 244
- sound4cl_FreeParameters, 244
- sound4cl_FreeParameterValue, 244
- sound4cl_GetAudio, 244
- sound4cl_GetAudioInputCount, 244
- sound4cl_GetAudioOutputCount, 244
- sound4cl_GetBufferIn, 244
- sound4cl_GetBufferOut, 244
- sound4cl_GetBytesFromFormat, 245
- sound4cl_GetChannelCount, 245
- sound4cl_GetChunkSizeInFrames, 245
- sound4cl_GetEstimatedDelay, 245
- sound4cl_GetFormatFromName, 245
- sound4cl_GetFormatName, 245
- sound4cl_GetMaxPacketFrame, 245
- sound4cl_GetMetadataInfos, 245
- sound4cl_GetOutputCount, 245
- sound4cl_GetParameter, 246
- sound4cl_GetPossibleChunkSizeInFrames, 246
- sound4cl_GetProcessChunkFrames, 246
- sound4cl_GetSampleRate, 246
- sound4cl_GetVersion, 246
- sound4cl_info, 241
- sound4cl_InitProcess, 246
- sound4cl_InitProcess2, 246
- sound4cl_InitProcess3, 246
- sound4cl_loggerfn, 240
- sound4cl_LogSeverity, 241
- sound4cl_MonoToStereo, 247
- sound4cl_MonoToStereo_Planar, 247
- sound4cl_NewBus, 247
- sound4cl_NewClient, 247
- sound4cl_NewParameters, 247
- sound4cl_none, 241
- sound4cl_PresetChange_Kind, 242
- sound4cl_PresetManager_InformChange, 247
- sound4cl_ProcessAudio, 247
- sound4cl_ProcessAudio_Planar, 248
- sound4cl_ProcessJson, 248
- sound4cl_SampleFormat, 242
- sound4cl_SaveState, 248
- sound4cl_SetInstanceBus, 248
- sound4cl_SetLoggerCallback, 248
- sound4cl_SetLogSeverity, 248
- sound4cl_SetMetadata, 248
- sound4cl_SetMetadataMulti, 248
- sound4cl_SetParameter, 249
- sound4cl_SetPresetManager, 249
- sound4cl_SOUND4_GetProcessInfo, 249
- sound4cl_StartUpdateThread, 249
- sound4cl_StereoToMono, 249
- sound4cl_StereoToMono_Planar, 249
- sound4cl_StopUpdateThread, 250
- sound4cl_storage_exists, 240
- sound4cl_storage_getall, 240
- sound4cl_storage_getall_free, 240
- sound4cl_storage_reader, 240
- sound4cl_storage_reader_free, 241
- sound4cl_storage_remove, 241
- sound4cl_storage_rename, 241
- sound4cl_storage_writer, 241
- sound4cl_TerminateProcess, 250
- sound4cl_verbose, 241
- sound4cl_verbose2, 241
- sound4cl_verbose3, 241
- sound4cl_verbose4, 241
- sound4cl_verbose5, 241
- sound4cl_WaitUpdateThreadReady, 250
- sound4cl_warning, 241
- sound4cl_Webserver, 250
- sound4cl_Webserver_FreeString, 250
- sound4cl_Webserver_GetAppHealth, 250
- sound4cl_Webserver_SetAppHealth, 250
- sound4cl_Webserver_Status, 251
- sound4cl_Webserver_Stop, 251

- sound4cl_Webserver_tcp, [251](#)
 - sound4cl_Webserver_tcp2, [251](#)
- sound4cl_DeleteClient
 - sound4cl_cdef.h, [243](#)
- sound4cl_error
 - sound4cl_cdef.h, [241](#)
- sound4cl_ExitProcess
 - sound4cl_cdef.h, [243](#)
- sound4cl_fatal
 - sound4cl_cdef.h, [241](#)
- sound4cl_FreeBus
 - sound4cl_cdef.h, [243](#)
- sound4cl_FreeJsonAnswer
 - sound4cl_cdef.h, [243](#)
- sound4cl_FreeMetadataInfos
 - sound4cl_cdef.h, [244](#)
- sound4cl_FreeParameters
 - sound4cl_cdef.h, [244](#)
- sound4cl_FreeParameterValue
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetAudio
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetAudioInputCount
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetAudioOutputCount
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetBufferIn
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetBufferOut
 - sound4cl_cdef.h, [244](#)
- sound4cl_GetBytesFromFormat
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetChannelCount
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetChunkSizeInFrames
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetEstimatedDelay
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetFormatFromName
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetFormatName
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetMaxPacketFrame
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetMetadataInfos
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetOutputCount
 - sound4cl_cdef.h, [245](#)
- sound4cl_GetParameter
 - sound4cl_cdef.h, [246](#)
- sound4cl_GetPossibleChunkSizeInFrames
 - sound4cl_cdef.h, [246](#)
- sound4cl_GetProcessChunkFrames
 - sound4cl_cdef.h, [246](#)
- sound4cl_GetSampleRate
 - sound4cl_cdef.h, [246](#)
- sound4cl_GetVersion
 - sound4cl_cdef.h, [246](#)
- sound4cl_info
 - sound4cl_cdef.h, [241](#)
- sound4cl_InitProcess
 - sound4cl_cdef.h, [246](#)
- sound4cl_InitProcess2
 - sound4cl_cdef.h, [246](#)
- sound4cl_InitProcess3
 - sound4cl_cdef.h, [246](#)
- sound4cl_loggerfn
 - sound4cl_cdef.h, [240](#)
- sound4cl_LogSeverity
 - sound4cl_cdef.h, [241](#)
- sound4cl_MonoToStereo
 - sound4cl_cdef.h, [247](#)
- sound4cl_MonoToStereo_Planar
 - sound4cl_cdef.h, [247](#)
- sound4cl_NewBus
 - sound4cl_cdef.h, [247](#)
- sound4cl_NewClient
 - sound4cl_cdef.h, [247](#)
- sound4cl_NewParameters
 - sound4cl_cdef.h, [247](#)
- sound4cl_none
 - sound4cl_cdef.h, [241](#)
- sound4cl_PresetChange_Kind
 - sound4cl_cdef.h, [242](#)
- sound4cl_PresetManager_InformChange
 - sound4cl_cdef.h, [247](#)
- sound4cl_ProcessAudio
 - sound4cl_cdef.h, [247](#)
- sound4cl_ProcessAudio_Planar
 - sound4cl_cdef.h, [248](#)
- sound4cl_ProcessJson
 - sound4cl_cdef.h, [248](#)
- sound4cl_SampleFormat
 - sound4cl_cdef.h, [242](#)
- sound4cl_SaveState
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetInstanceBus
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetLoggerCallback
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetLogSeverity
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetMetadata
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetMetadataMulti
 - sound4cl_cdef.h, [248](#)
- sound4cl_SetParameter
 - sound4cl_cdef.h, [249](#)
- sound4cl_SetPresetManager
 - sound4cl_cdef.h, [249](#)
- sound4cl_SOUND4_GetProcessInfo
 - sound4cl_cdef.h, [249](#)
- sound4cl_SOUND4_ProcessInfo, [170](#)
 - name, [170](#)
 - prefix, [170](#)
 - shortname, [170](#)

- struct_size, [171](#)
- version, [171](#)
- sound4cl_StartUpdateThread
 - sound4cl_cdef.h, [249](#)
- sound4cl_StereoToMono
 - sound4cl_cdef.h, [249](#)
- sound4cl_StereoToMono_Planar
 - sound4cl_cdef.h, [249](#)
- sound4cl_StopUpdateThread
 - sound4cl_cdef.h, [250](#)
- sound4cl_storage_exists
 - sound4cl_cdef.h, [240](#)
- sound4cl_storage_getall
 - sound4cl_cdef.h, [240](#)
- sound4cl_storage_getall_free
 - sound4cl_cdef.h, [240](#)
- sound4cl_storage_reader
 - sound4cl_cdef.h, [240](#)
- sound4cl_storage_reader_free
 - sound4cl_cdef.h, [241](#)
- sound4cl_storage_remove
 - sound4cl_cdef.h, [241](#)
- sound4cl_storage_rename
 - sound4cl_cdef.h, [241](#)
- sound4cl_storage_writer
 - sound4cl_cdef.h, [241](#)
- sound4cl_TerminateProcess
 - sound4cl_cdef.h, [250](#)
- sound4cl_verbose
 - sound4cl_cdef.h, [241](#)
- sound4cl_verbose2
 - sound4cl_cdef.h, [241](#)
- sound4cl_verbose3
 - sound4cl_cdef.h, [241](#)
- sound4cl_verbose4
 - sound4cl_cdef.h, [241](#)
- sound4cl_verbose5
 - sound4cl_cdef.h, [241](#)
- sound4cl_WaitUpdateThreadReady
 - sound4cl_cdef.h, [250](#)
- sound4cl_warning
 - sound4cl_cdef.h, [241](#)
- sound4cl_Webserver
 - sound4cl_cdef.h, [250](#)
- sound4cl_Webserver_FreeString
 - sound4cl_cdef.h, [250](#)
- sound4cl_Webserver_GetAppHealth
 - sound4cl_cdef.h, [250](#)
- sound4cl_Webserver_SetAppHealth
 - sound4cl_cdef.h, [250](#)
- sound4cl_Webserver_Status
 - sound4cl_cdef.h, [251](#)
- sound4cl_Webserver_Stop
 - sound4cl_cdef.h, [251](#)
- sound4cl_Webserver_tcp
 - sound4cl_cdef.h, [251](#)
- sound4cl_Webserver_tcp2
 - sound4cl_cdef.h, [251](#)
- StartUpdateThread
 - sound4::impact::dyn::CDynLib, [94](#)
- StartWebServer
 - sound4::CInstance, [110](#)
 - sound4::impact::CInstance, [120](#)
 - sound4::impact::dyn::CInstance, [129](#)
- std, [78](#)
- StereoToMono
 - sound4::CProcessor, [143](#)
 - sound4::impact::dyn::CDynLib, [94](#)
- StereoToMono_Planar
 - sound4::CProcessor, [144](#)
 - sound4::impact::dyn::CDynLib, [94](#)
- Stop
 - sound4::CInstance, [110](#)
 - sound4::impact::CInstance, [120](#)
 - sound4::impact::dyn::CInstance, [129](#)
- StopUpdateThread
 - sound4::impact::dyn::CDynLib, [95](#)
- StopWebServer
 - sound4::CInstance, [110](#)
 - sound4::impact::CInstance, [120](#)
 - sound4::impact::dyn::CInstance, [130](#)
- struct_size
 - SOUND4_ProcessInfo, [169](#)
 - sound4cl_SOUND4_ProcessInfo, [171](#)
- Terminate
 - sound4::CInstance, [111](#)
 - sound4::impact::CInstance, [120](#)
 - sound4::impact::dyn::CInstance, [130](#)
- TerminateProcess
 - sound4::impact::dyn::CDynLib, [95](#)
- UNIXLIKE
 - sound4.impact.cl_dyn.hpp, [209](#)
 - sound4cl.hpp, [224](#)
- Update Management, [38](#)
 - impact_StartUpdateThread, [38](#)
 - impact_StopUpdateThread, [38](#)
 - impact_WaitUpdateThreadReady, [39](#)
- Usage
 - sound4.impact.cl-pipe-cxx.cpp, [174](#)
 - sound4.impact.cl-pipe-dyn.cpp, [178](#)
 - sound4.impact.cl-pipe.cpp, [183](#)
- verbose
 - Global setup, [22](#)
 - sound4, [71](#)
- verbose2
 - Global setup, [22](#)
 - sound4, [71](#)
- verbose3
 - Global setup, [22](#)
 - sound4, [71](#)
- verbose4
 - Global setup, [22](#)
 - sound4, [71](#)
- verbose5

- Global setup, [22](#)
- sound4, [71](#)
- version
 - SOUND4_ProcessInfo, [169](#)
 - sound4cl_SOUND4_ProcessInfo, [171](#)
- WaitUpdateThreadReady
 - sound4::impact::dyn::CDynLib, [95](#)
- warning
 - Global setup, [22](#)
 - sound4, [71](#)
- Web server, [58](#)
 - impact_Webserver, [59](#)
 - impact_Webserver_FreeString, [59](#)
 - impact_Webserver_GetAppHealth, [60](#)
 - impact_Webserver_SetAppHealth, [60](#)
 - impact_Webserver_Status, [60](#)
 - impact_Webserver_Stop, [61](#)
 - impact_Webserver_tcp, [61](#)
 - impact_Webserver_tcp2, [62](#)
 - SOUND4_INVALID_WEBSERVER_ID, [58](#)
 - SOUND4_WEBSERVER_HTTP_OK, [58](#)
 - SOUND4_WEBSERVER_HTTPS_OK, [59](#)
- Webserver
 - sound4::impact::dyn::CDynLib, [95](#)
- Webserver_FreeString
 - sound4::impact::dyn::CDynLib, [95](#)
- Webserver_GetAppHealth
 - sound4::impact::dyn::CDynLib, [95](#)
- Webserver_SetAppHealth
 - sound4::impact::dyn::CDynLib, [95](#)
- Webserver_Status
 - sound4::impact::dyn::CDynLib, [96](#)
- Webserver_Stop
 - sound4::impact::dyn::CDynLib, [96](#)
- Webserver_tcp
 - sound4::impact::dyn::CDynLib, [96](#)
- Webserver_tcp2
 - sound4::impact::dyn::CDynLib, [96](#)
- Write
 - MyPresetManager, [161](#)
 - sound4::CPresetLoader, [132](#)
 - sound4::impact::CPresetLoader, [134](#)
 - sound4::impact::dyn::CPresetLoader, [135](#)